Message & SMS

Developer Guide

Issue 1.0

Date 2023-04-20

HUAWEI

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: https://www.huawei.com

Email: support@huawei.com

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to “Vul. Response Process”. For details about the policy, see the following website:https://www.huawei.com/en/psirt/vul-response-process
For enterprise customers who need to obtain vulnerability information, visit:https://securitybulletin.huawei.com/enterprise/en/security-advisory

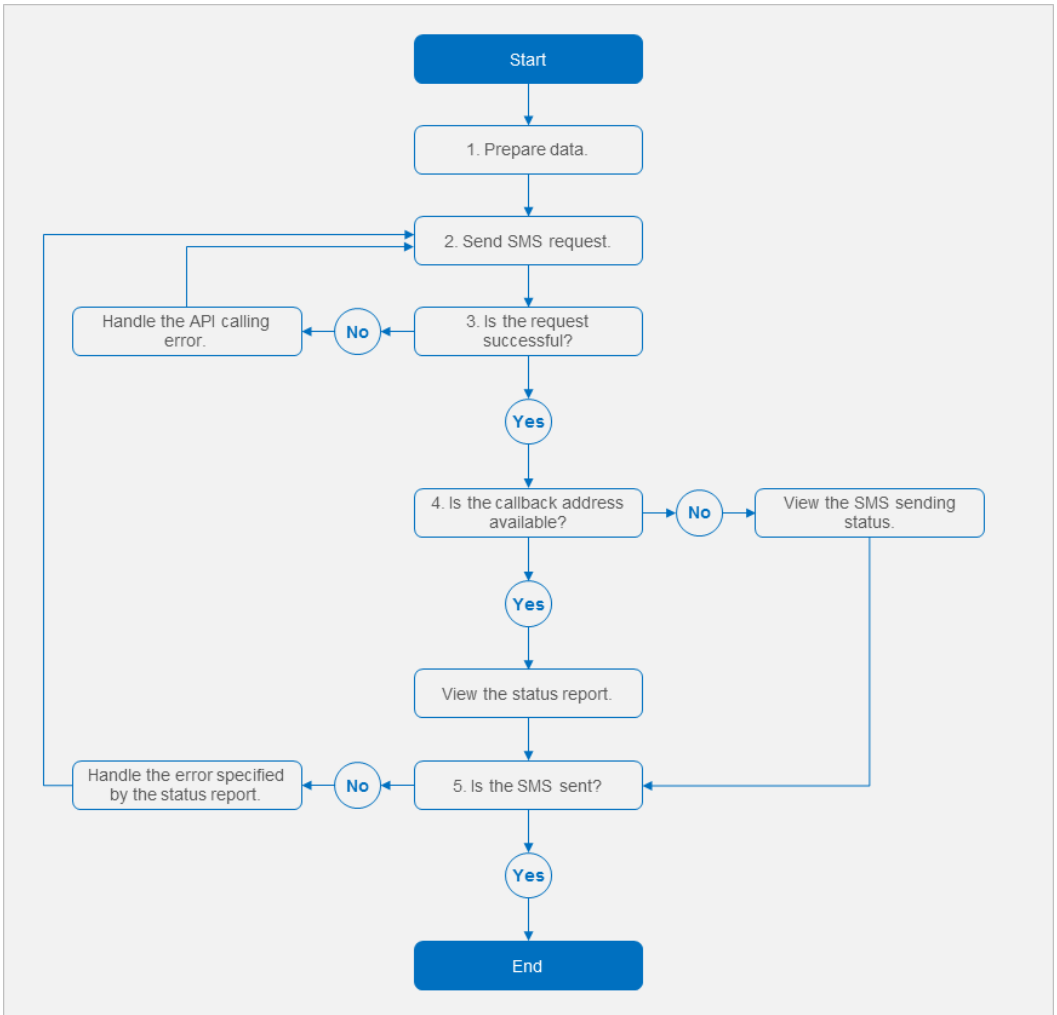
Contents

1 Beginner....1
2 Development Preparation....4
3 Code Examples....7
3.1 AK/SK Authentication (Recommended)....7
3.1.1 Java....7
3.1.2 PHP....14
3.1.3 Python....17
3.1.4 C#....21
3.1.5 Node.js....26
3.1.6 Go....31
3.2 X-WSSE Authentication....36
3.2.1 Java....36
3.2.2 PHP....46
3.2.3 Python....53
3.2.4 C#....59
3.2.5 Node.js....69
3.2.6 Go....73

1 Beginner

Service Process

The following figure shows the general custom-made software development process for the SMS service.



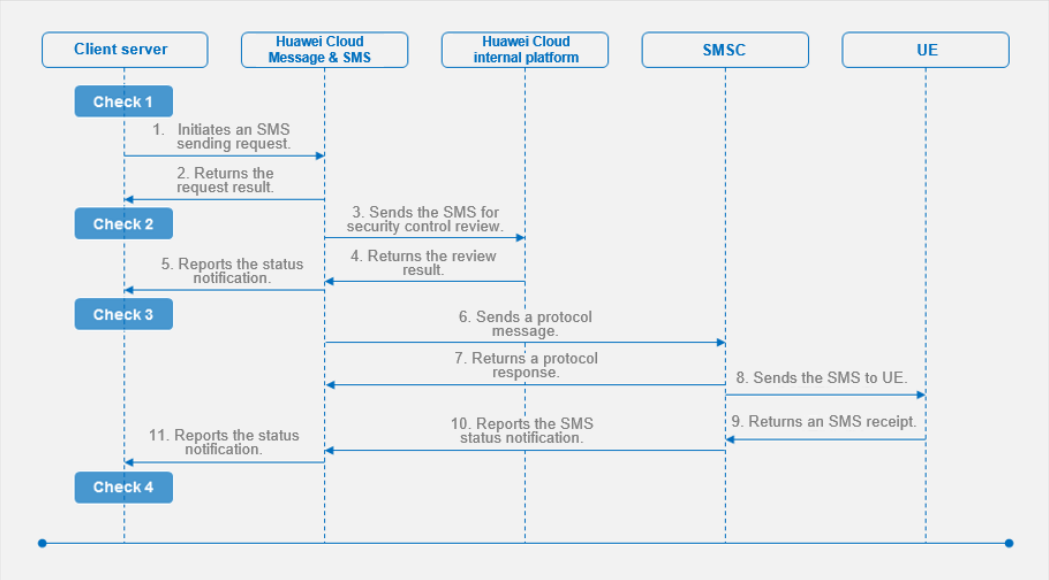
1. Perform steps described in **Development Preparation** to obtain the data required for calling the SMS API.

2. Follow instructs in the **Code Examples** to call the SMS API and initiate an SMS request.
3. View the received response and check whether the request succeeds.
   - Yes: Go to **4**.
   - No: Follow instructions in API Error Codes to rectify the fault and repeat **2**.
4. Check whether the request contains the **statusCallback** parameter. The **statusCallback** parameter specifies the callback address to which the status report is sent.
   - Yes: Check the received status report.
   - No: Log in to the Message & SMS console choose **Send Details**. On the displayed page, view the sending results and status codes.

   The **Send Details** function will be online soon.
5. Check whether the SMS message is successfully sent based on the status report.
   - Yes: No further operation is required.
   - No: Follow instructions in SMS Status Error Codes to rectify the fault and repeat **2**.

Commissioning Guide

During the custom-made software development process for the SMS service, developers need to pay attention to the following service commissioning points:



Note: The SMSC includes China Mobile, China Unicom, China Telecom, China Broadnet, and suppliers (line agents).

- **Check 1**: Before initiating an SMS sending request, the system checks the validity of parameters in the request. For example:
  - The value of the **Content-Type** parameter in the request headers is **application/x-www-form-urlencoded** in the SMS sending API and **application/json** in the batch SMS sending API.

  - The value of the **to** parameter in the request body is a character string in the SMS sending API and a character string array in the batch SMS sending API.
  - If SMS status reports need to be received, the **statusCallback** parameter must be specified and the address is valid and reachable.
  - If the template type specified by **templateId** is a common template, you must set **signature** to the signature added before the SMS content in the common template.

- **Check 2**: When obtaining the request result, parse the response result code and rectify the fault based on the handling suggestions in API Error Codes.

```
HTTP/1.1 200 OK
Date: Fri, 13 Apr 2018 06:29:08 GMT
Server: WebServer
Content-Type: application/json;charset=UTF-8
Content-Length: 220

{"result":
[{"originTo":"+86155****5678","createTime":"2018-05-25T16:34:34Z","from":"1069031221280012","smsMsgId":"d6e3cdd0-522b-4692-8304-a07553cdf591_8539659","status":"000000"}],"code":"000000","description":"Success"}
```

  If **code** is set to **E000510**, the value of **status** needs to be parsed for fault locating and analysis.

- **Check 3**: When the SMS content sent triggers the interception by the Huawei Cloud internal platform, the Huawei Cloud Message & SMS service pushes a status report to the customer. Parse the status code and rectify the fault based on the handling suggestions for the status codes provided by the Huawei Cloud internal platform in SMS Status Error Codes.

  Note: Huawei Cloud Message & SMS pushes the status report notification only when the callback address is specified for **statusCallback** in the **SMS sending request**. Otherwise, log in to the SMS console and go to the sending details page to view the status code.

- **Check 4**: When receiving the SMS status notification from the SMSC, the Huawei Cloud Message & SMS service pushes a status report to the customer. Parse the status code and rectify the fault based on the handling suggestions in SMS Status Error Codes.

  Note: Huawei Cloud Message & SMS pushes the status report notification only when the callback address is specified for **statusCallback** in the **SMS sending request**. Otherwise, log in to the SMS console and go to the sending details page to view the status code.

2 Development Preparation

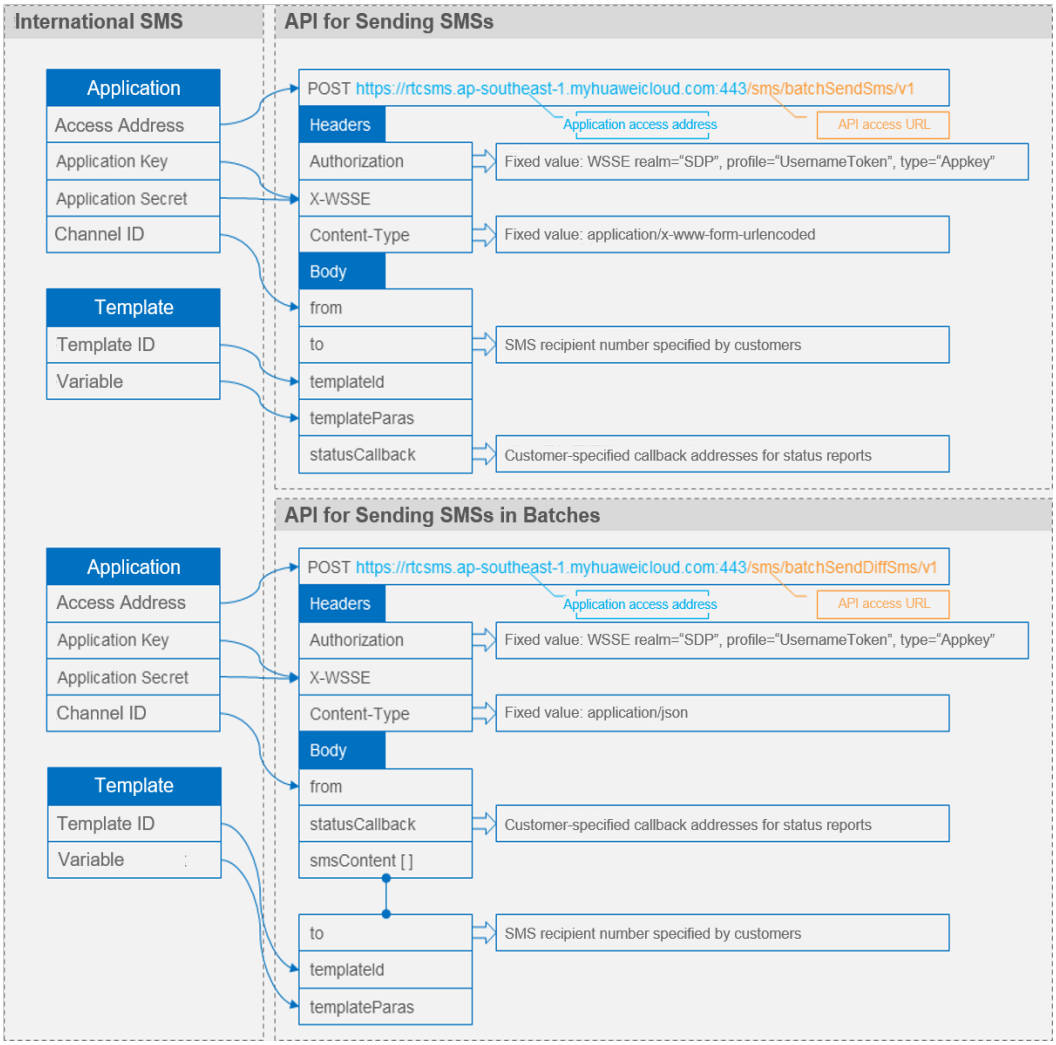
European SMS

The following parameters must be specified before you develop a European SMS application.

Parameter	Example Value	Where to Find the Setting	Document
Application Key	c8RWg3ggEcyd4D3p94bf3Y7x1Ile	Log in to the Message & SMS console, and choose **European SMS** > **Application Management**.	Creating an SMS Application
Application Secret	q4Ii87BhST9vcs8wvrzN80SfD7Al		
Application Access Address	https://smsapi.eu-west-101.myhuaweicloud.eu/msgsms		
Channel No.	isms12345678		

Parameter	Example Value	Where to Find the Setting	Document
Template ID	8ff55eac1d0b478ab3c06c3c6a492300	Log in to the Message & SMS console, and choose **European SMS** > **Template Management**. Note: Obtain the value of this parameter based on the value of **Application**. When sending different SMSs in batches, you can specify multiple template IDs.	Applying for an SMS Template
URI	SMS Sending: **/sms/batchSendSms/v1** Batch SMS sending: **/sms/batchSendDiffSms/v1**	Obtain the URI from the **API Type** described in section SMS Sending API or Batch SMS Sending API.	SMS Sending API Batch SMS Sending API

The relationships between the data to be prepared and the SMS API request parameters are as follows:



3 Code Examples

3.1 AK/SK Authentication (Recommended)

3.1.1 Java

Example	**Example of Sending SMSs**, **Example of Sending SMSs in Batches** **Receiving Status Reports**
Environment	**JDK 1.8** or later

Download required SDKs and demos for **Java**. Copy the following code sample to a new Java file and run it.

NOTICE

- Sending SMSs shows an example of sending group SMSs with a single template. Sending SMSs in batches shows an example of sending group SMSs with multiple templates.
- The examples described in this document may involve the use of personal data. You are advised to comply with relevant laws and regulations and take measures to ensure that personal data is fully protected.
- The examples described in this document are for demonstration purposes only. Commercial use of the examples is prohibited.
- Information in this document is for your reference only. It does not constitute any offer or commitment.
- For details about how to send a video SMS message, see "Sending an SMS Message".

Example of Sending SMSs

```
import com.huawei.apig.sdk.util.Constant;

import com.cloud.apigateway.sdk.utils.Client;
import com.cloud.apigateway.sdk.utils.Request;

import org.apache.http.Header;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpRequestBase;
import org.apache.http.conn.ssl.NoopHostnameVerifier;
import org.apache.http.conn.ssl.SSLConnectionSocketFactory;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.ssl.SSLContextBuilder;
import org.apache.http.util.EntityUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import javax.net.ssl.SSLContext;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.Arrays;

public class SendSms {

    private static final Logger LOGGER = LoggerFactory.getLogger(SendSms.class);

    public static final String UTF_8 = "UTF-8";

    private static CloseableHttpClient client =  null;

    public static void main(String[] args) throws Exception {
            // Ignore the certificate trust issues to prevent API calling failures caused by HTTPS certificate authentication failures.
        client = createIgnoreSSLHttpClient();
        sendSms();
    }

    private static void sendSms() throws IOException {
        // Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
        String url = "https://smsapi.cn-north-4.myhuaweicloud.com:443/sms/batchSendSms/v1"; // Application access address (obtain it from the Application Management page on the console) and API access URI.
        // Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
        String appKey = "c8RWg3gg**********3Y7x1Ile"; //APP_Key
        String appSecret = "q4Ii87Bh**********80SfD7Al"; //APP_Secret
        String sender = "csms12345678"; // Channel number for Chinese mainland SMS or international SMS
        String templateId = "79948563bcba4e49aa6f2ba7f26298ef"; // Template ID

        // Mandatory for Chinese mainland SMS. This parameter is valid and mandatory when Template Type corresponding to templateId is Universal template. The signature name must be approved and be the same type as the template.
        // This parameter is not required for international SMS.
        String signature = "Huawei Cloud SMS test"; // Signature

        // Mandatory. Global number format (including the country code), for example, +86151****6789. Use commas (,) to separate multiple numbers.
        String receiver = "+86151****6788,+86151****6789"; // Recipient numbers

        // Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
        String statusCallBack = "https://your.server.com/rest/callback/statusReport";

        /**
         * Optional. If a non-variable template is used, assign an empty value to this parameter, for example, String templateParas = "";
```



```
        * Example of a single-variable template: If the template content is "Your verification code is ${1}", templateParas can be set to "[\"111111\"]".
        * Example of a dual-variable template: If the template content is "You have ${1} delivered to ${2}", templateParas can be set to "[\"3\",\"main gate of People's Park\"]".
        * Each variable in the template must be assigned a value, and the value cannot be empty.
        * To view more information, choose Service Overview > Template and Variable Specifications.
        */
        String templateParas = "[\"111111\"]"; // Template variable. The following uses a single-variable verification code SMS as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

        // Request body. If the signature name is not required, set signature to null.
        String body = buildRequestBody(sender, receiver, templateId, templateParas, statusCallBack, signature);
        if (null == body || body.isEmpty()) {
            LOGGER.warn("body is null.");
            return;
        }

        Request request = new Request();
        request.setKey(appKey);
        request.setSecret(appSecret);
        request.setMethod("POST");
        request.setUrl(url);
        request.addHeader("Content-Type", "application/x-www-form-urlencoded");
        request.setBody(body);
        LOGGER.info("Print the body: {}", body);

        try {
            HttpRequestBase signedRequest = Client.sign(request, Constant.SIGNATURE_ALGORITHM_SDK_HMAC_SHA256);
            LOGGER.info("Print the authorization: {}", Arrays.toString(signedRequest.getHeaders("Authorization")));
            Header[] requestAllHeaders = signedRequest.getAllHeaders();
            for (Header h : requestAllHeaders) {
                LOGGER.info("req Header with name: {} and value: {}", h.getName(), h.getValue());
            }

            HttpResponse response = client.execute(signedRequest);

            LOGGER.info("Print the status line of the response: {}", response.getStatusLine().toString());
            Header[] resHeaders = response.getAllHeaders();
            for (Header h : resHeaders) {
                LOGGER.info("Processing Header with name: {} and value: {}", h.getName(), h.getValue());
            }
            HttpEntity resEntity = response.getEntity();
            if (resEntity != null) {
                LOGGER.info("Processing Body with name: {} and value: {}", System.getProperty("line.separator"),
                    EntityUtils.toString(resEntity, "UTF-8"));
            }
        } catch (Exception e) {
            LOGGER.error(e.getMessage(), e);
        }
    }

    public static CloseableHttpClient createIgnoreSSLHttpClient() throws Exception {
        SSLContext sslContext = new SSLContextBuilder().loadTrustMaterial(null,  (x509CertChain, authType) -> true).build();
        return HttpClients.custom().setSSLSocketFactory(new SSLConnectionSocketFactory(sslContext, NoopHostnameVerifier.INSTANCE)).build();
    }

    static String buildRequestBody(String sender, String receiver, String templateId, String templateParas,
        String statusCallBack, String signature) throws UnsupportedEncodingException {
        if (null == sender || null == receiver || null == templateId || sender.isEmpty() || receiver.isEmpty()
            || templateId.isEmpty()) {
            System.out.println("buildRequestBody(): sender, receiver or templateId is null.");
            return null;
        }
```



```
        StringBuilder body = new StringBuilder();
        appendToBody(body, "from=", sender);
        appendToBody(body, "&to=", receiver);
        appendToBody(body, "&templateId=", templateId);
        appendToBody(body, "&templateParas=", templateParas);
        appendToBody(body, "&statusCallback=", statusCallBack);
        appendToBody(body, "&signature=", signature);
        return body.toString();
    }

    private static void appendToBody(StringBuilder body, String key, String val) throws
UnsupportedEncodingException {
        if (null != val && !val.isEmpty()) {
            LOGGER.info("Print appendToBody: {}:{}", key, val);
            body.append(key).append(URLEncoder.encode(val, UTF_8));
        }
    }
}
```

Example of Sending SMSs in Batches

Configure the following dependencies in Maven.

```
<dependency>
            <groupId>org.json</groupId>
            <artifactId>json</artifactId>
            <version>20180130</version>
        </dependency>
```

```
import com.huawei.apig.sdk.util.Constant;

import com.cloud.apigateway.sdk.utils.Client;
import com.cloud.apigateway.sdk.utils.Request;

import org.apache.http.Header;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpRequestBase;
import org.apache.http.conn.ssl.NoopHostnameVerifier;
import org.apache.http.conn.ssl.SSLConnectionSocketFactory;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.ssl.SSLContextBuilder;
import org.apache.http.util.EntityUtils;
import org.json.JSONArray;
import org.json.JSONObject;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import javax.net.ssl.SSLContext;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class SendDiffSms {
private static final Logger LOGGER = LoggerFactory.getLogger(SendDiffSms.class);

private static CloseableHttpClient client =  null;

public static void main(String[] args) throws Exception {
        // Ignore the certificate trust issues to prevent API calling failures caused by HTTPS certificate
authentication failures.
client = createIgnoreSSLHttpClient();
sendSms();
}

private static void sendSms() throws IOException {
```



```
// Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
String url
= "https://smsapi.cn-north-4.myhuaweicloud.com:443/sms/batchSendDiffSms/v1"; // Application access address (obtain it from the Application Management page on the console) and API access URI
    // Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
String appKey = "c8RWg3gg**********3Y7x1Ile"; //APP_Key
String appSecret = "q4Ii87Bh**********80SfD7Al"; //APP_Secret
String sender = "csms12345678"; // Channel number for Chinese mainland SMS or international SMS
String templateId1 = "8ff55eac1d0b478ab3c06c3c6a492300"; // Template ID1
String templateId2 = "8ff55eac1d0b478ab3c06c3c6a492300"; // Template ID 2

// Mandatory for Chinese mainland SMS. This parameter is valid and mandatory when Template Type corresponding to templateId is Universal template. The signature name must be approved and be the same type as the template.
// This parameter is not required for international SMS.
String signature1 = "Huawei Cloud SMS test"; // Signature 1
String signature2 = "Huawei Cloud SMS test"; // Signature 2

    // Mandatory. Global number format (including the country code), for example, +86151****6789. Use commas (,) to separate multiple numbers.
String[] receiver1 = {"+86151****6789", "+86152****7890"}; // Recipient number of template 1
String[] receiver2 = {"+86151****6789", "+86152****7890"}; // Recipient number of template 2

    // Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
String statusCallBack = "";

/**
* Optional. If a non-variable template is used, assign an empty value to this parameter, for example, String templateParas = {};
* Example of a single-variable template: If the template content is "Your verification code is ${1}", templateParas can be set to ["111111"].
* Example of a dual-variable template: If the template content is "You have ${1} parcel delivered to ${2}", templateParas can be set to '{"3","main gate of People's Park"}'.
* To view more information, choose Service Overview > Template and Variable Specifications.
*/
String[] templateParas1 = {
"123456"
    }; // Template 1 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).
String[] templateParas2 = {
"234567"
    }; // Template 2 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

    // smsContent. If the signature name is not required, set signature to null.
List<Map<String, Object>> smsContent = new ArrayList<Map<String, Object>>();
Map<String, Object> item1 = initDiffSms(receiver1, templateId1, templateParas1, signature1);
Map<String, Object> item2 = initDiffSms(receiver2, templateId2, templateParas2, signature2);
if (null != item1 && !item1.isEmpty()) {
smsContent.add(item1);
}
if (null != item2 && !item2.isEmpty()) {
smsContent.add(item2);
}

    // Request body
String body = buildRequestBody(sender, smsContent, statusCallBack);
if (null == body || body.isEmpty()) {
System.out.println("body is null.");
return;
}

Request request = new Request();
request.setKey(appKey);
```



```
request.setSecret(appSecret);
request.setMethod("POST");
request.setUrl(url);
request.addHeader("Content-Type", "application/json");
request.setBody(body);
LOGGER.info("Print the body: {}", body);

try {
// Sign the request.
HttpRequestBase signedRequest = Client.sign(request,
Constant.SIGNATURE_ALGORITHM_SDK_HMAC_SHA256);
// Print request header parameters: Authorization
LOGGER.info("Print the allHeaders: {}", Arrays.toString(signedRequest.getAllHeaders()));
HttpResponse response = client.execute(signedRequest);
// Print the status line of the response.
LOGGER.info("Print the status line of the response: {}", response.getStatusLine().toString());
// Print the header fields of the response.
Header[] resHeaders = response.getAllHeaders();
for (Header h : resHeaders) {
LOGGER.info("Processing Header with name: {} and value: {}", h.getName(), h.getValue());
}
// Print the body of the response.
HttpEntity resEntity = response.getEntity();
if (resEntity != null) {
LOGGER.info("Processing Body with name: {} and value: {}", System.getProperty("line.separator"),
EntityUtils.toString(resEntity, "UTF-8"));
}
} catch (Exception e) {
LOGGER.error(e.getMessage(), e);
}
}

public static CloseableHttpClient createIgnoreSSLHttpClient() throws Exception {
SSLContext sslContext = new SSLContextBuilder().loadTrustMaterial(null,  (x509CertChain, authType) ->
true).build();
return HttpClients.custom().setSSLSocketFactory(new SSLConnectionSocketFactory(sslContext,
NoopHostnameVerifier.INSTANCE)).build();
}

static String buildRequestBody(String sender, List<Map<String, Object>> smsContent, String statusCallBack)
{
if (null == sender || null == smsContent || sender.isEmpty() || smsContent.isEmpty()) {
System.out.println("buildRequestBody(): sender or smsContent is null.");
return null;
}
JSONArray jsonArr = new JSONArray();

for (Map<String, Object> it : smsContent) {
jsonArr.put(it);
}

Map<String, Object> data = new HashMap<String, Object>();
data.put("from", sender);
data.put("smsContent", jsonArr);
if (null != statusCallBack && !statusCallBack.isEmpty()) {
data.put("statusCallback", statusCallBack);
}

return new JSONObject(data).toString();
}

/**
    * Construct the value of smsContent.
*/
static Map<String, Object> initDiffSms(String[] receiver, String templateId, String[] templateParas,
String signature) {
if (null == receiver || null == templateId || receiver.length == 0 || templateId.isEmpty()) {
System.out.println("initDiffSms(): receiver or templateId is null.");
return null;
```



```
}
Map<String, Object> map = new HashMap<String, Object>();
map.put("to", receiver);
map.put("templateId", templateId);
if (null != templateParas && templateParas.length > 0) {
map.put("templateParas", templateParas);
}
if (null != signature && !signature.isEmpty()) {
map.put("signature", signature);
}

return map;
}

}
```

Receiving Status Reports

The Maven dependency to be introduced is org.springframework:spring-web:5.3.21 (example version).

```
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class DemoController {
    /**
     * Synchronize SMS receipts.
     */
    @PostMapping("/report")
    public void smsHwReport(@RequestParam String smsMsgId,  // Unique SMS identifier returned after an SMS is successfully sent.
                            @RequestParam(required = false) String total,     // Number of SMS segments split from a long SMS message. If the SMS message is not split, set this parameter to 1.
                            @RequestParam(required = false) String sequence,  // Sequence number after a long SMS is split. This parameter is valid only when the value of total is greater than 1. If the SMS is not split, set this parameter to 1.
                            @RequestParam String status,    // Enumerated values of an SMS status report. For details about the values, see API Reference.
                            @RequestParam(required = false) String source,    // Source of the SMS status report. 1: generated by the SMS platform. 2: returned by the SMSC. 3: generated by the cloud platform.
                            @RequestParam(required = false) String updateTime,// SMS resource update time, which is generally the UTC time when the Message & SMS platform receives the SMS status report. The value is in the format of yyyy-MM-dd'T'HH:mm:ss'Z'. The time is converted to %3a using urlencode. // When the Message & SMS platform does not receive the SMS status report from the SMSC, the platform constructs a status report that does not contain the updateTime parameter.
                            @RequestParam(required = false) String orgCode,   // Status codes of southbound NEs are transparently transmitted. // When the status code is not returned, the parameter is not used.
                            @RequestParam(required = false) String extend,    // Extended field in the request sent by a user. If the SMS sent by a user does not carry the extend parameter, the status report does not contain the extend parameter.
                            @RequestParam(required = false) String to) {      // Recipient number of the SMS corresponding to the status report. This parameter is carried only when the status report contains the extend parameter.
        System.out.println("  ================receive smsStatusReport ======================");
        System.out.println("smsMsgId: " + smsMsgId);
        System.out.println("total: " + total);
        System.out.println("sequence: " + sequence);
        System.out.println("status: " + status);
        System.out.println("source: " + source);
        System.out.println("updateTime: " + updateTime);
        System.out.println("orgCode: " + orgCode);
        System.out.println("extend: " + extend);
        System.out.println("to: " + to);
    }
}
```

3.1.2 PHP

Example	**Example of Sending SMSs**, **Example of Sending SMSs in Batches** **Receiving Status Reports**
Environment	**php-7.4.33-Win32-vc15-x64** Install the PHP plug-in (Oro PHPStorm Plugin) on IntelliJ IDEA (2021.1.3 version).

Download required SDKs and demos for **PHP**. Replace the content of the **index.php** file with the following code example.

NOTICE

- Sending SMSs shows an example of sending group SMSs with a single template. Sending SMSs in batches shows an example of sending group SMSs with multiple templates.
- The examples described in this document may involve the use of personal data. You are advised to comply with relevant laws and regulations and take measures to ensure that personal data is fully protected.
- The examples described in this document are for demonstration purposes only. Commercial use of the examples is prohibited.
- Information in this document is for your reference only. It does not constitute any offer or commitment.

Example of Sending SMSs

```
<html>
<head>
   <title>Send SMS test</title>
</head>
<body>
<pre>
   <?php
   require 'signer.php';
   $signer = new Signer();
   // Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
   $signer->Key = 'c8RWg3ggEcyd4D3p94bf3Y7x1Ile';      // app key
   $signer->Secret = "q4Ii87Bh**********80SfD7Al";   // app secret

   $req = new Request('POST', 'https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendSms/v1');
   //Add header parameters
   $req->headers = array(
       'content-type' => 'application/x-www-form-urlencoded',
   );
   $req->body = 'from=csms12345678&to=%2B8615112346788&templateId=8ff55eac1d0b478ab3c06c3c6a492300&templateParas=["1"]&statusCallback=https://your.server.com/rest/callback/statusReport';
   $curl = $signer->Sign($req);
   curl_setopt($curl, CURLOPT_SSL_VERIFYPEER, FALSE);
   curl_setopt($curl, CURLOPT_SSL_VERIFYHOST, FALSE);
```



```
    curl_setopt($curl, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_1);
    var_dump($req->headers);
    var_dump($req);
    echo "--------------\n";
    $response = curl_exec($curl);
    $status = curl_getinfo($curl, CURLINFO_HTTP_CODE);
    if ($status == 0) {
        echo curl_error($curl);
    } else {
        echo $status . "\n";
        echo $response;
    }
    curl_close($curl);
    ?>
</pre>
</body>
</html>
```

Example of Sending SMSs in Batches

```
<html>
<head>
    <title>Send Diff SMS test</title>
</head>
<body>
<pre>
    <?php
    require 'signer.php';

    // Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
    $url = 'https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendDiffSms/v1'; // Application access address (obtain it from the Application Management page on the console) and API access URI.
    // Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
    $APP_KEY = 'c8RWg3ggEcyd4D3p94bf3Y7x1Ile'; //APP_Key
    $APP_SECRET = 'q4Ii87Bh**********80SfD7Al'; //APP_Secret
    $sender = 'csms12345678'; // Channel number for Chinese mainland SMS or international SMS
    $TEMPLATE_ID_1 = '8ff55eac1d0b478ab3c06c3c6a492300'; // Template ID 1
    $TEMPLATE_ID_2 = '8ff55eac1d0b478ab3c06c3c6a492300'; // Template ID 2

    // Mandatory for Chinese mainland SMS. This parameter is valid and mandatory when Template Type corresponding to templateId is Universal template. The signature name must be approved and be the same type as the template.
    // This parameter is not required for international SMS.
    $signature_1 = "Huawei Cloud SMS test"; // Signature 1
    $signature_2 = "Huawei Cloud SMS test"; // Signature 2

    // Mandatory. Global number format (including the country code), for example, +86151****6789. Use commas (,) to separate multiple numbers.
    $receiver_1 = ['+86151****6789', '+86152****7890']; // Recipient number of template 1
   $receiver_2 = ['+86151****6789', '+86152****7890']; // Recipient number of template 2

    // Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
    $statusCallback = '';

    /**
     * Optional. If a template with no variable is used, assign an empty value to this parameter, for example, $TEMPLATE_PARAS = [];
     * Example of a single-variable template: If the template content is "Your verification code is ${NUM_6}", $TEMPLATE_PARAS can be set to ['369751'].
     * Example of a dual-variable template: If the template content is "You have ${NUM_2} parcels delivered to ${TXT_20}", $TEMPLATE_PARAS can be set to ['3','main gate of People's Park'].
          * To view more information, choose Service Overview > Template and Variable Specifications.
     */
    $TEMPLATE_PARAS_1 = ['123456']; // Template 1 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to
```



```
2569).
    $TEMPLATE_PARAS_2 = ['234567']; // Template 2 variable. The following uses a single-variable
verification code SMS message as an example. The customer needs to generate a 6-digit verification code
and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to
2569).

    // Request body
    $data = json_encode([
        'from' => $sender,
        'statusCallback' => $statusCallback,
        'smsContent' => [// Mandatory. Set the parameters based on the number of template IDs.
            // smsContent. If the signature name is not required, set signature to ''.
            initDiffSms($receiver_1, $TEMPLATE_ID_1, $TEMPLATE_PARAS_1, $signature_1),
            initDiffSms($receiver_2, $TEMPLATE_ID_2, $TEMPLATE_PARAS_2, $signature_2)
        ]
    ]);

    $signer = new Signer();
    $signer->Key = $APP_KEY;      // app key
    $signer->Secret = $APP_SECRET;   // app secret

    $req = new Request('POST', $url);
    //Add header parameters
    $req->headers = array(
        'content-type' => 'application/json',
    );
    $req->body = $data;
    $curl = $signer->Sign($req);
    curl_setopt($curl, CURLOPT_SSL_VERIFYPEER, FALSE);
    curl_setopt($curl, CURLOPT_SSL_VERIFYHOST, FALSE);
    curl_setopt($curl, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_1);
    var_dump($req->headers);
    var_dump($req);
    echo "--------------\n";
    $response = curl_exec($curl);
    $status = curl_getinfo($curl, CURLINFO_HTTP_CODE);
    if ($status == 0) {
        echo curl_error($curl);
    } else {
        echo $status . "\n";
        echo $response;
    }
    curl_close($curl);

    /**
     * Construct the value of smsContent.
     * @param array $receiver
     * @param string $templateId
     * @param array $templateParas
     * @param string $signature | Signature name, which must be specified when the universal template for
Chinese mainland SMS is used.
     * @return string[]
     */
    function initDiffSms(array $receiver, string $templateId, array $templateParas, string $signature)
    {
        if (null != $signature && strlen($signature) > 0) {
            return ['to' => $receiver, 'templateId' => $templateId, 'templateParas' => $templateParas, 'signature'
=> $signature];
        }
        return ['to' => $receiver, 'templateId' => $templateId, 'templateParas' => $templateParas];
    }

    ?>
</pre>
</body>
</html>
```


Receiving Status Reports

```
<?php
// Data example (urlencode) of the status report reported by the SMS platform
// $success_body =
'sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId=2ea20735-
f856-4376-afbf-570bd70a46ee_11840135&status=DELIVRD';
$failed_body =
'orgCode=E200027&sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId
=2ea20735-f856-4376-afbf-570bd70a46ee_11840135&status=RTE_ERR';

// onSmsStatusReport($success_body);
onSmsStatusReport($failed_body);

/**
 * Parse the status report data.
 *
 * @param string $data: Status report data reported by the SMS platform
 */
function onSmsStatusReport(string $data)
{
    $keyValues = [];
    parse_str(urldecode($data), $keyValues); // Parse the status report data.

    /**
            * Example: Parsing status is used as an example.
     *
     * 'smsMsgId': Unique ID of an SMS
     * 'total': Number of SMS segments
     * 'sequence': Sequence number of an SMS after splitting
     * 'source': Status report source
     * 'updateTime': Resource update time
     * 'status': Enumeration values of the status report
     * 'orgCode': Status code
     */
    $status = $keyValues['status']; // Enumerated values of the status report
    // Check whether the SMS is successfully sent based on the value of status.
    if ('DELIVRD' === strtoupper($status)) {
        print 'Send sms success. smsMsgId: ' . $keyValues['smsMsgId'] . PHP_EOL;
    } else {
        // The SMS fails to be sent. The values of status and orgCode are recorded.
        print 'Send sms failed. smsMsgId: ' . $keyValues['smsMsgId'] . PHP_EOL;
        print 'Failed status: ' . $status . PHP_EOL;
        print 'Failed orgCode: ' . $keyValues['orgCode'] . PHP_EOL;
    }
}
?>
```

3.1.3 Python

Example	Example of Sending SMSs, Example of Sending SMSs in Batches Receiving Status Reports
Environment	Python 3.7

Download required SDKs and demos for Python. Replace the content of the **main.py** file with the following code example.

NOTICE

- Sending SMSs shows an example of sending group SMSs with a single template. Sending SMSs in batches shows an example of sending group SMSs with multiple templates.
- The examples described in this document may involve the use of personal data. You are advised to comply with relevant laws and regulations and take measures to ensure that personal data is fully protected.
- The examples described in this document are for demonstration purposes only. Commercial use of the examples is prohibited.
- Information in this document is for your reference only. It does not constitute any offer or commitment.

Example of Sending SMSs

```
import urllib.parse
import urllib.request
import requests
from apig_sdk import signer

if __name__ == '__main__':

    # Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
    url = 'https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendSms/v1' # Application access address and API access URI
    # Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
    APP_KEY = "c8RWg3ggEcyd4D3p94bf3Y7x1Ile" #APP_Key
    APP_SECRET = "q4Ii87Bh**********80SfD7Al" #APP_Secret
sender = "csms12345678" # Channel number for Chinese mainland SMS or international SMS
    TEMPLATE_ID = "8ff55eac1d0b478ab3c06c3c6a492300" # Template ID

    # Mandatory for Chinese mainland SMS. This parameter is valid and mandatory when Template Type corresponding to templateId is Universal template. The signature name must be approved and be the same type as the template.
    # This parameter is not required for international SMS.
    signature := "Huawei Cloud SMS test" # Signature

    # Mandatory. Global number format (including the country code), for example, +86151****6789. Use commas (,) to separate multiple numbers.
    receiver = "+86151****6789,+86152****7890" # Recipient numbers

    # Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
    statusCallBack = ""

    '''
    Optional. If a template with no variable is used, assign an empty value to this parameter, for example, TEMPLATE_PARAM = '';
    Example of a single-variable template: If the template content is "Your verification code is ${NUM_6}", TEMPLATE_PARAM can be set to '["369751"]'.
    Example of a dual-variable template: If the template content is "You have ${NUM_2} parcels delivered to ${TXT_20}", TEMPLATE_PARAM can be set to '["3","main gate of People's Park"]'.
    Each variable in the template must be assigned a value, and the value cannot be empty.
    To view more information, choose Service Overview > Template and Variable Specifications.
    '''
    TEMPLATE_PARAM = '["369751"]' # Template variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

    formData = urllib.parse.urlencode({
        'from': sender,
```



```
        'to': receiver,
        'templateId': TEMPLATE_ID,
        'templateParas': TEMPLATE_PARAM,
        'statusCallback': statusCallBack,
        # 'signature': signature # Uncomment this line if the universal template for Chinese mainland SMS is used.
    }).encode('ascii')
    print(formData)

    sig = signer.Signer()
    sig.Key = APP_KEY
    sig.Secret = APP_SECRET

    r = signer.HttpRequest("POST", url)
    r.headers = {"content-type": "application/x-www-form-urlencoded"}
    r.body = formData

    sig.Sign(r)
    print(r.headers["X-Sdk-Date"])
    print(r.headers["Authorization"])
    resp = requests.request(r.method, r.scheme + "://" + r.host + r.uri, headers=r.headers, data=r.body, verify=False)
    print(resp.status_code, resp.reason)
    print(resp.content)
```

Example of Sending SMSs in Batches

```
import json

import requests

from apig_sdk import signer

if __name__ == '__main__':
    # Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
    url = 'https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendDiffSms/v1'  # Application access address (obtain it from the Application Management page on the console) and API access URI.
    // Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
    APP_KEY = "c8RWg3ggEcyd4D3p94bf3Y7x1Ile"  # APP_Key
    APP_SECRET = "q4Ii87Bh**********80SfD7Al"  # APP_Secret
    sender = "csms12345678" # Channel number for Chinese mainland SMS or international SMS
    TEMPLATE_ID_1 = "8ff55eac1d0b478ab3c06c3c6a492300"  # Template ID 1
    TEMPLATE_ID_2 = "8ff55eac1d0b478ab3c06c3c6a492300"  # Template ID 2

    # Mandatory for Chinese mainland SMS. This parameter is valid and mandatory when Template Type corresponding to templateId is Universal template. The signature name must be approved and be the same type as the template.
    # This parameter is not required for international SMS.
    signature_1 = "Huawei Cloud SMS test"  # Signature 1
    signature_2 = "Huawei Cloud SMS test"  # Signature 2

    # Mandatory. Global number format (including the country code), for example, +86151****6789. Use commas (,) to separate multiple numbers.
    receiver_1 = ["+86151****6789", "+86152****7890"]  # Recipient number of template 1
    receiver_2 = ["+86151****6789", "+86152****7890"]  # Recipient number of template 2

    # Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
    statusCallBack = ""

    '''
    Optional. If a template with no variable is used, assign an empty value to this parameter, for example, TEMPLATE_PARAM = [];
    Example of a single-variable template: If the template content is "Your verification code is ${1}", TEMPLATE_PARAM can be set to ["369751"].
    Example of a dual-variable template: If the template content is "You have ${1} parcel delivered to ${2}", TEMPLATE_PARAM can be set to ["3","main gate of People's Park"].
```



```
    Each variable in the template must be assigned a value, and the value cannot be empty.
    To view more information, choose Service Overview > Template and Variable Specifications.
    '''
    TEMPLATE_PARAM_1 = ["123456"]  # Template 1 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).
    TEMPLATE_PARAM_2 = ["234567"]  # Template 2 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

    '''
    Optional. If a template with no variable is used, assign an empty value to this parameter, for example, TEMPLATE_PARAM = '';
    Example of a single-variable template: If the template content is "Your verification code is ${NUM_6}", TEMPLATE_PARAM can be set to ["369751"].
    Example of a dual-variable template: If the template content is "You have ${NUM_2} parcels delivered to ${TXT_20}", TEMPLATE_PARAM can be set to ["3","main gate of People's Park"].
    To view more information, choose Service Overview > Template and Variable Specifications.
    '''
    TEMPLATE_PARAM = '["369751"]'  # Template variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

    # Request body
    jsonData = json.dumps({'from': sender,
                           'statusCallback': statusCallBack,
                           'smsContent': [
                               {'to': receiver_1,
                                'templateId': TEMPLATE_ID_1,
                                'templateParas': TEMPLATE_PARAM_1,
                                # 'signature':signature_1 # Uncomment this line if the universal template for Chinese mainland SMS is used.
                                },
                               {'to': receiver_2,
                                'templateId': TEMPLATE_ID_2,
                                'templateParas': TEMPLATE_PARAM_2,
                                # 'signature':signature_2 # Uncomment this line if the universal template for Chinese mainland SMS is used.
                                }]
                           }).encode('ascii')

    sig = signer.Signer()
    sig.Key = APP_KEY
    sig.Secret = APP_SECRET

    r = signer.HttpRequest("POST", url)
    r.headers = {"content-type": "application/json"}
    r.body = jsonData

    sig.Sign(r)
    print(r.headers["X-Sdk-Date"])
    print(r.headers["Authorization"])
    resp = requests.request(r.method, r.scheme + "://" + r.host + r.uri, headers=r.headers, data=r.body, verify=False)
    print(resp.status_code, resp.reason)
    print(resp.content)
```

Receiving Status Reports

```
import urllib.parse

# Data example (urlencode) of the status report reported by the SMS platform
#success_body =
"sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId=2ea20735-f856-4376-afbf-570bd70a46ee_11840135&status=DELIVRD";
failed_body =
"orgCode=E200027&sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId
```

```
=2ea20735-f856-4376-afbf-570bd70a46ee_11840135&status=RTE_ERR";

'''
Parse the status report data.
@param data: Status report data reported by the SMS platform.
@return:
'''
def onSmsStatusReport(data):
    keyValues = urllib.parse.parse_qs(data); # Parse the status report data.

    '''
    Example: Parsing status is used as an example.

    'smsMsgId': Unique ID of an SMS
    'total': Number of SMS segments
    'sequence': Sequence number of an SMS after splitting
    'source': Status report source
    'updateTime': Resource update time
    'status': Enumeration values of the status report
     'orgCode': Status code
    '''
    status = keyValues.get('status'); #Enumerated values of the status report
    # Check whether the SMS is successfully sent based on the value of status.
    if 'DELIVRD' == str.upper(status[0]):
        print('Send sms success. smsMsgId: ', keyValues.get('smsMsgId')[0]);
    else:
        # The SMS fails to be sent. The values of status and orgCode are recorded.
        print('Send sms failed. smsMsgId: ', keyValues.get('smsMsgId')[0]);
        print('Failed status: ', status[0]);
        print('Failed orgCode: ', keyValues.get('orgCode')[0]);

if __name__ == '__main__':
#     onSmsStatusReport(success_body)
    onSmsStatusReport(failed_body)
```

3.1.4 C#

Example	**Example of Sending SMSs**, **Example of Sending SMSs in Batches** **Receiving Status Reports**
Environment	**.Net 7.0**, **Visual Studio Code 1.75.1**, and plug-ins of C# v1.25.7 and Code Runner v0.12.0 (optional)

NOTICE

- Sending SMSs shows an example of sending group SMSs with a single template. Sending SMSs in batches shows an example of sending group SMSs with multiple templates.
- The examples described in this document may involve the use of personal data. You are advised to comply with relevant laws and regulations and take measures to ensure that personal data is fully protected.
- The examples described in this document are for demonstration purposes only. Commercial use of the examples is prohibited.
- Information in this document is for your reference only. It does not constitute any offer or commitment.

Download required SDKs and demos for **C#**. Take Visual Studio Code as an example. Select a directory in the local resource manager, for example, **D:/sms**, and use a terminal to access the directory.

1. Run the **dotnet new console** command to generate a project.
2. Copy **Signer.cs** and **HttpEncoder.cs** to **D:/sms**, create an empty **SendSms.cs** file in the **sms** folder, and copy the content in the following example to **SendSms.cs**.
3. Use Visual Studio Code to open the project and run **SendSms.cs**.

Example of Sending SMSs

```
using System;
using System.Net;
using System.IO;
using APIGATEWAY_SDK;
using System.Text;

namespace DEMO
{
    class Program
    {
        static void Main(string[] args)
        {
            ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12 | SecurityProtocolType.Tls11 |
SecurityProtocolType.Tls;

            // Mandatory. The values here are example values only. Obtain the actual values based on
Development Preparation.
            string apiAddress = "https://smsapi.cn-north-4.myhuaweicloud.com:443/sms/batchSendSms/v1"; //
Application access address (obtain it from the Application Management page on the console) and API
access URI.
            // Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret
and store them in the configuration file or environment variables.
            string appKey = "c8RWg3ggEcyd4D3p94bf3Y7x1Ile"; //APP_Key
            string appSecret = "q4Ii87Bh**********80SfD7AI"; //APP_Secret
            string sender = "csms12345678"; // Channel number for Chinese mainland SMS or international
SMS
            string templateId = "8ff55eac1d0b478ab3c06c3c6a492300"; // Template ID

            // Mandatory for Chinese mainland SMS. This parameter is valid and mandatory when Template
Type corresponding to templateId is Universal template. The signature name must be approved and be
the same type as the template.
            // This parameter is not required for international SMS.
            //string signature = "Huawei Cloud SMS test"; // Signature

            // Mandatory. Global number format (including the country code), for example, +86151****6789.
Use commas (,) to separate multiple numbers.
            string receiver = "+86137****6781,+86137****6782"; // Recipient numbers

            // Optional. Address for receiving SMS status reports. The domain name is recommended. If this
parameter is set to an empty value or left unspecified, customers do not receive status reports.
            string statusCallBack = "";

            /*
             * Optional. If a non-variable template is used, assign an empty value to this parameter, for
example, string templateParas = "";
             * Example of a single-variable template: If the template content is "Your verification code is ${1}",
templateParas can be set to "[\"369751\"]".
             * Example of a dual-variable template: If the template content is "You have ${1} delivered to
${2}", templateParas can be set to "[\"3\",\"main gate of People's Park\"]".
             * Each variable in the template must be assigned a value, and the value cannot be empty.
             * To view more information, choose Service Overview > Template and Variable Specifications.
             */
            string templateParas = "[\"369751\"]"; // Template variable. The following uses a single-variable
verification code SMS message as an example. The customer needs to generate a 6-digit verification code
```



```
and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to
2569).

        Signer signer = new Signer();
        signer.Key = appKey;
        signer.Secret = appSecret;

        HttpRequest r = new HttpRequest("POST", new Uri(apiAddress)); // Application access address
(obtain it from the Application Management page on the console) and API access URI.

        // Request body
        var body = new Dictionary<string, string>() {
            {"from", sender},
            {"to", receiver},
            {"templateId", templateId},
            {"templateParas", templateParas},
            {"statusCallback", statusCallBack},
            //{"signature", signature} // The signature must be entered when the universal template for
Chinese mainland SMS is used.
        };

        r.body =new FormUrlEncodedContent(body).ReadAsStringAsync().Result;
        r.headers.Add("Content-Type", "application/x-www-form-urlencoded");

        HttpWebRequest req = signer.Sign(r);
        Console.WriteLine(req.Headers.GetValues("x-sdk-date")[0]);
        Console.WriteLine(string.Join(", ", req.Headers.GetValues("authorization")));
        Console.WriteLine("body: " + r.body);

        // Do not verify the certificate.
        ServicePointManager.ServerCertificateValidationCallback = (sender, certificate, chain,
sslPolicyErrors) => true;
        try
        {
            var writer = new StreamWriter(req.GetRequestStream());
            writer.Write(r.body);
            writer.Flush();
            HttpWebResponse resp = (HttpWebResponse)req.GetResponse();
            var reader = new StreamReader(resp.GetResponseStream());
            Console.WriteLine(reader.ReadToEnd());
        }
        catch (WebException e)
        {
            HttpWebResponse resp = (HttpWebResponse)e.Response;
            if (resp != null)
            {
                Console.WriteLine((int)resp.StatusCode + " " + resp.StatusDescription);
                var reader = new StreamReader(resp.GetResponseStream());
                Console.WriteLine(reader.ReadToEnd());
            }
            else
            {
                Console.WriteLine(e.Message);
            }

        }
    }

  }
}
```

Example of Sending SMSs in Batches

Reference library: For Newtonsoft.Json 11.0.2 and later versions, see **https://www.newtonsoft.com/json**.

```
using System;
using System.Net;
using System.IO;
```



```
using APIGATEWAY_SDK;
using System.Text;
using System.Collections;
using System.Collections.Generic;

using Newtonsoft.Json;

namespace DEMODiffSMS
{
    class Program
    {
        static void Main(string[] args)
        {
            ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12 | SecurityProtocolType.Tls11 |
SecurityProtocolType.Tls;

            // Mandatory. The values here are example values only. Obtain the actual values based on
Development Preparation.
            string apiAddress = "https://smsapi.cn-north-4.myhuaweicloud.com:443/sms/batchSendDiffSms/
v1"; // Application access address (obtain it from the Application Management page on the console) and
API access URI.
           // Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret
and store them in the configuration file or environment variables.
            string appKey = "c8RWg3ggEcyd4D3p94bf3Y7x1Ile"; //APP_Key
            string appSecret = "q4Ii87Bh**********80SfD7Al"; //APP_Secret
            string sender = "csms12345678"; // Channel number for Chinese mainland SMS or international
SMS
            string templateId_1 = "8ff55eac1d0b478ab3c06c3c6a492300"; // Template ID 1
            string templateId_2 = "8ff55eac1d0b478ab3c06c3c6a492301"; // Template ID 2

            // Mandatory for Chinese mainland SMS. This parameter is valid and mandatory when Template
Type corresponding to templateId is Universal template. The signature name must be approved and be
the same type as the template.
            // This parameter is not required for international SMS.
            string signature_1 = "Huawei Cloud SMS test 1"; // Signature 1
            string signature_2 = "Huawei Cloud SMS test 2"; // Signature 2

            // Mandatory. Global number format (including the country code), for example, +86151****6789.
Use commas (,) to separate multiple numbers.
            string[] receiver_1 = { "+86151****6781", "+86151****6783" }; // Recipient number of template 1
            string[] receiver_2 = { "+86151****6782", "+86151****6784" }; // Recipient number of template 2

            // Optional. Address for receiving SMS status reports. The domain name is recommended. If this
parameter is set to an empty value or left unspecified, customers do not receive status reports.
            string statusCallBack = "https://your.server.com/rest/callback/statusReport";

            /*
             * Optional. If a non-variable template is used, assign an empty value to this parameter, for
example, string[] templateParas = {};
            * Example of a single-variable template: If the template content is "Your verification code is ${1}",
templateParas can be set to ["369751"].
             * Example of a dual-variable template: If the template content is "You have ${1} delivered to ${2}",
templateParas can be set to {"3","main gate of People's Park"}.
             * Each variable in the template must be assigned a value, and the value cannot be empty.
             * To view more information, choose Service Overview > Template and Variable Specifications.
             */
            string[] templateParas_1 = {"123456"}; // Template 1 variable. The following uses a single-variable
verification code SMS message as an example. The customer needs to generate a 6-digit verification code
and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to
2569).
            string[] templateParas_2 = {"234567"}; // Template 2 variable. The following uses a single-variable
verification code SMS message as an example. The customer needs to generate a 6-digit verification code
and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to
2569).

            ArrayList smsContent = new ArrayList
            {
                // smsContent. If the signature name is not required, set signature to null.
                InitDiffSms(receiver_1, templateId_1, templateParas_1, signature_1),
```



```
            InitDiffSms(receiver_2, templateId_2, templateParas_2, signature_2)
        };

        // Request body
        var body = new Dictionary<string, object>{
            {"from", sender},
            {"statusCallback", statusCallBack},
            {"smsContent", smsContent}
        };

        Signer signer = new Signer();
        signer.Key = appKey;
        signer.Secret = appSecret;

        HttpRequest r = new HttpRequest("POST", new Uri(apiAddress));
        r.body = JsonConvert.SerializeObject(body);
        r.headers.Add("Content-Type", "application/json");

        HttpWebRequest req = signer.Sign(r);
        Console.WriteLine(req.Headers.GetValues("x-sdk-date")[0]);
        Console.WriteLine(string.Join(", ", req.Headers.GetValues("authorization")));
        Console.WriteLine("body: " + r.body);

        // Do not verify the certificate.
        ServicePointManager.ServerCertificateValidationCallback = (sender, certificate, chain,
sslPolicyErrors) => true;
        try
        {
            var writer = new StreamWriter(req.GetRequestStream());
            writer.Write(r.body);
            writer.Flush();
            HttpWebResponse resp = (HttpWebResponse)req.GetResponse();
            var reader = new StreamReader(resp.GetResponseStream());
            Console.WriteLine(reader.ReadToEnd());
        }
        catch (WebException e)
        {
            HttpWebResponse resp = (HttpWebResponse)e.Response;
            if (resp != null)
            {
                Console.WriteLine((int)resp.StatusCode + " " + resp.StatusDescription);
                var reader = new StreamReader(resp.GetResponseStream());
                Console.WriteLine(reader.ReadToEnd());
            }
            else
            {
                Console.WriteLine(e.Message);
            }

        }
    }

    static Dictionary<string, object> InitDiffSms(string[] receiver, string templateId, string[] templateParas,
string signature)
    {
        Dictionary<string, object> dic = new Dictionary<string, object>
        {
            {"to", receiver},
            {"templateId", templateId},
            {"templateParas", templateParas}
        };
        if (!signature.Equals(null) && signature.Length > 0)
        {
            dic.Add("signature", signature);
        }

        return dic;
    }
```



```
    }
}
```

Receiving Status Reports

```
using System;
using System.Web;

namespace msgsms_csharp_demo
{
    class Report
    {
        static void Main(string[] args)
        {
            //string success_body =
"sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId=2ea20735-
f856-4376-afbf-570bd70a46ee_11840135&status=DELIVRD";
            string failed_body =
"orgCode=E200027&sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId
=2ea20735-f856-4376-afbf-570bd70a46ee_11840135&status=RTE_ERR";

            //onSmsStatusReport(success_body);
            onSmsStatusReport(failed_body);
        }

        /// <summary>
        /// Parse the status report data.
        /// </summary>
        ///<param name="data">Status report data reported by the SMS platform</param>
        static void onSmsStatusReport(string data)
        {
            var keyValues = HttpUtility.ParseQueryString(data); // Parse the status report data.

            /**
             * Example: Parsing status is used as an example.
             *
             * 'smsMsgId': Unique ID of an SMS
             * 'total': Number of SMS segments
             * 'sequence': Sequence number of an SMS after splitting
             * 'source': Status report source
             * 'updateTime': Resource update time
             * 'status': Enumeration values of the status report
             * 'orgCode': Status code
             */
            string status = keyValues.Get("status"); // Enumerated values of the status report
            // Check whether the SMS is successfully sent based on the value of status.
            if ("DELIVRD".Equals(status.ToUpper()))
            {
                Console.WriteLine("Send sms success. smsMsgId: " + keyValues.Get("smsMsgId"));
            }
            else
            {
                // The SMS fails to be sent. The values of status and orgCode are recorded.
                Console.WriteLine("Send sms failed. smsMsgId: " + keyValues.Get("smsMsgId"));
                Console.WriteLine("Failed status: " + status);
                Console.WriteLine("Failed orgCode: " + keyValues.Get("orgCode"));
            }
        }
    }
}
```

3.1.5 Node.js

Example	Example of Sending SMSs, Example of Sending SMSs in Batches, Receiving Status Reports

Environment	Install the Node.js plug-in (bundled 211.7628.21) on IntelliJ IDEA (2021.1.3 version).

NOTICE

- Sending SMSs shows an example of sending group SMSs with a single template. Sending SMSs in batches shows an example of sending group SMSs with multiple templates.
- The examples described in this document may involve the use of personal data. You are advised to comply with relevant laws and regulations and take measures to ensure that personal data is fully protected.
- The examples described in this document are for demonstration purposes only. Commercial use of the examples is prohibited.
- Information in this document is for your reference only. It does not constitute any offer or commitment.

Download required SDKs and demos for **JavaScript**. Replace the content of the **node_demo.js** file with the following code example.

Example of Sending SMSs

```
var signer = require('./signer');
var https = require("https");
var url = require('url'); // Introduce the URL module.
var querystring = require('querystring'); // Introduce the querystring module.

// Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
var realUrl = 'https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendSms/v1'; // Application access address (obtain it from the Application Management page on the console) and API access URI.
// Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
var appKey = 'c8RWg3ggEcyd4D3p94bf3Y7x1Ile';
var appSecret = 'q4Ii87Bh**********80SfD7Al';
var sender = 'csms12345678';  // Channel number for Chinese mainland SMS or international SMS
var templateId = '8ff55eac1d0b478ab3c06c3c6a492300'; // Template ID

// Mandatory for Chinese mainland SMS. This parameter is valid and mandatory when Template Type corresponding to templateId is Universal template. The signature name must be approved and be the same type as the template.
// This parameter is not required for international SMS.
var signature = "Huawei Cloud SMS test"; // Signature

// Mandatory. Global number format (including the country code), for example, +86151****6789. Use commas (,) to separate multiple numbers.
var receiver = '+86151****6789,+86152****7890'; // Recipient numbers

// Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
var statusCallBack = '';

/**
 * Optional. If a non-variable template is used, assign an empty value to this parameter, for example, var templateParas = '';
 * Example of a single-variable template: If the template content is "Your verification code is ${NUM_6}", templateParas can be set to '["369751"]'.
 * Example of a dual-variable template: If the template content is "You have ${NUM_2} delivered to ${TXT_20}", templateParas can be set to '["3","main gate of People's Park"]'.
```



```
 * Each variable in the template must be assigned a value, and the value cannot be empty.
 * To view more information, choose Service Overview > Template and Variable Specifications.
 */
var templateParas = '["369751"]'; // Template variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

var urlobj = url.parse(realUrl); // Parse the realUrl character string and return a URL object.

var sig = new signer.Signer();
sig.Key = appKey;
sig.Secret = appSecret;

var r = new signer.HttpRequest("POST", realUrl);
r.headers = {"Content-Type": "application/x-www-form-urlencoded"};
r.body = buildRequestBody(sender, receiver, templateId, templateParas, statusCallBack, signature);

var opt = sig.Sign(r);
opt.hostname=urlobj.hostname
opt.port=urlobj.port
console.log(opt)

var req = https.request(opt, function (res) {
    console.log(res.statusCode);
    console.log('headers:', JSON.stringify(res.headers));
    res.on("data", function (chunk) {
        console.log(chunk.toString())
    })
});

req.on("error", function (err) {
    console.log(err.message)
});
req.write(r.body);
req.end();


function buildRequestBody(sender, receiver, templateId, templateParas, statusCallBack, signature){
    if (null !== signature && signature.length > 0) {
        return querystring.stringify({
            'from': sender,
            'to': receiver,
            'templateId': templateId,
            'templateParas': templateParas,
            'statusCallback': statusCallBack,
            'signature': signature
        });
    }

    return querystring.stringify({
        'from': sender,
        'to': receiver,
        'templateId': templateId,
        'templateParas': templateParas,
        'statusCallback': statusCallBack
    });
}
```

Example of Sending SMSs in Batches

```
var signer = require('./signer');
var https = require("https");
var url = require('url'); // Introduce the URL module.

// Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
var realUrl = 'https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendDiffSms/v1'; // Application access address and API access URI
// Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store
```



```
them in the configuration file or environment variables.
var appKey = 'c8RWg3ggEcyd4D3p94bf3Y7x1Ile'; //Application Key
var appSecret = 'q4Ii87Bh**********80SfD7Al'; //Application Secret
var sender = 'csms12345678'; // Channel number for Chinese mainland SMS or international SMS
var templateId1 = '8ff55eac1d0b478ab3c06c3c6a492300'; // Template ID 1
var templateId2 = '8ff55eac1d0b478ab3c06c3c6a492300'; // Template ID 2

// Mandatory for Chinese mainland SMS. This parameter is valid and mandatory when Template Type corresponding to templateId is Universal template. The signature name must be approved and be the same type as the template.
// This parameter is not required for international SMS.
var signature1 = "Huawei Cloud SMS test"; // Signature 1
var signature2 = "Huawei Cloud SMS test"; // Signature 2

// Mandatory. Global number format (including the country code), for example, +8615123456789. Use commas (,) to separate multiple numbers.
var receiver1 = ['+8615123456789','+8615234567890']; // Recipient number of template 1
var receiver2 = ['+8615123456789','+8615234567890']; // Recipient number of template 2

// Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
var statusCallBack = '';

/**
 * Optional. If a non-variable template is used, assign an empty value to this parameter, for example, var templateParas = [];
 * Example of a single-variable template: If the template content is "Your verification code is ${NUM_6}", templateParas can be set to ['369751'].
 * Example of a dual-variable template: If the template content is "You have ${NUM_2} parcels delivered to ${TXT_20}", templateParas can be set to ['3','main gate of People's Park'].
 * To view more information, choose Service Overview > Template and Variable Specifications.
 */
var templateParas1 = ['123456']; // Template 1 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).
var templateParas2 = ['234567']; // Template 2 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

/**
 * Construct the value of smsContent.
 *
 * @param receiver from
 * @param templateId to
 * @param templateParas template params value
 * @param signature | Signature name, which must be specified when the universal template for Chinese mainland SMS is used.
 * @returns
 */
function initDiffSms(receiver, templateId, templateParas, signature){
    if (null !== signature && signature.length > 0) {
        return {'to': receiver, 'templateId': templateId, 'templateParas': templateParas, 'signature': signature};
    }
    return {'to': receiver, 'templateId': templateId, 'templateParas': templateParas};
}

var body = JSON.stringify ({ //Request body)
    'from': sender,
    'statusCallback': statusCallBack,
    'smsContent': [
        // smsContent. If the signature name is not required, set signature to null.
        initDiffSms(receiver1, templateId1, templateParas1, signature1),
        initDiffSms(receiver2, templateId2, templateParas2, signature2)
    ]}
);


var urlobj = url.parse(realUrl); // Parse the realUrl character string and return a URL object.
```



```
var sig = new signer.Signer();
sig.Key = appKey;
sig.Secret = appSecret;

var r = new signer.HttpRequest("POST", realUrl);
r.headers = {"Content-Type": "application/json"};
r.body = body;

var opt = sig.Sign(r);
opt.hostname=urlobj.hostname
opt.port=urlobj.port
console.log(opt)

var req = https.request(opt, function (res) {
    console.log(res.statusCode);
    console.log('headers:', JSON.stringify(res.headers));
    res.on("data", function (chunk) {
        console.log(chunk.toString())
    })
});

req.on("error", function (err) {
    console.log(err.message)
});
req.write(r.body);
req.end();
```

Receiving Status Reports

```
package main

import (
    "fmt"
    "net/url"
    "strings"
)

func main() {
    // Data example (urlencode) of the status report reported by the SMS platform
    //success_body :=
"sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId=2ea20735-
f856-4376-afbf-570bd70a46ee_11840135&status=DELIVRD";
    failed_body :=
"orgCode=E200027&sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId
=2ea20735-f856-4376-afbf-570bd70a46ee_11840135&status=RTE_ERR";
    //onSmsStatusReport(success_body);
    onSmsStatusReport(failed_body);
}

func onSmsStatusReport(data string) {
    ss, _ := url.QueryUnescape(data)
    params := strings.Split(ss, "&")
    keyValues := make(map[string]string)
    for i := range params {
        temp := strings.Split(params[i],"=")
        keyValues[temp[0]] = temp[1];
    }
    /**
    * Example: Parsing status is used as an example.
    *
    * 'smsMsgId': Unique ID of an SMS
    * 'total': Number of SMS segments
    * 'sequence': Sequence number of an SMS after splitting
    * 'source': Status report source
    * 'updateTime': Resource update time
    * 'status': Enumeration values of the status report
    * 'orgCode': Status code
    */
    status := keyValues["status"];
```



```
    if status == "DELIVRD" {
        fmt.Println("Send sms success. smsMsgId: " + keyValues["smsMsgId"])
    } else {
        fmt.Println("Send sms failed. smsMsgId: " + keyValues["smsMsgId"])
        fmt.Println("Failed status:  " + keyValues["status"])
        fmt.Println("Failed orgCode:  " + keyValues["orgCode"])
    }
}
```

3.1.6 Go

Example	**Example of Sending SMSs**, **Example of Sending SMSs in Batches** **Receiving Status Reports**
Environment	**go1.17.1**. Install the Go 211.7628.1 plug-in on IntelliJ IDEA (2021.1.3 version).

NOTICE

- Sending SMSs shows an example of sending group SMSs with a single template. Sending SMSs in batches shows an example of sending group SMSs with multiple templates.
- The examples described in this document may involve the use of personal data. You are advised to comply with relevant laws and regulations and take measures to ensure that personal data is fully protected.
- The examples described in this document are for demonstration purposes only. Commercial use of the examples is prohibited.
- Information in this document is for your reference only. It does not constitute any offer or commitment.

Download required SDKs and demos for **Go**. Replace the content of the **demo.go** file with the following code example.

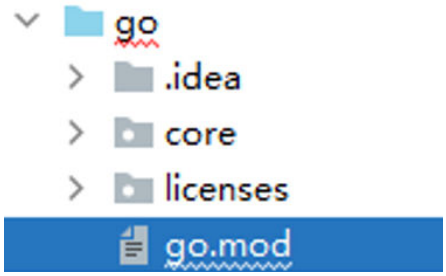
Modify the content of the **go.mod** file at the first layer as follows:

```
module huaweicloud.com/apig/go/signer

require huaweicloud.com/apig/signer v0.0.0

replace huaweicloud.com/apig/signer v0.0.0 => ./core

go 1.12
```



Example of Sending SMSs

```
package main

import (
  "bytes"
  "crypto/tls"
  "fmt"
  core "huaweicloud.com/apig/signer"
  "io/ioutil"
  "net/http"
  "net/url"
)

func main() {
  // Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
  appInfo := core.Signer{
    // Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
    Key:    "c8RWg3ggEcyd4D3p94bf3Y7x1Ile", //App Key
    Secret: "q4Ii87Bh**********80SfD7Al", //App Secret
  }
  apiAddress := "https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendSms/v1" // Application access address (obtain it from the Application Management page on the console) and API access URI.
  sender := "csms12345678"                    // Channel number for Chinese mainland SMS or international SMS
  templateId := "8ff55eac1d0b478ab3c06c3c6a492300"          // Template ID

  // Mandatory for Chinese mainland SMS. This parameter is valid and mandatory when Template Type corresponding to templateId is Universal template. The signature name must be approved and be the same type as the template.
  // This parameter is not required for international SMS.
  signature := "Huawei Cloud SMS test" // Signature

  // Mandatory. Global number format (including the country code), for example, +86151****6789. Use commas (,) to separate multiple numbers.
  receiver := "+86151****6789" // Recipient numbers

  // Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
  statusCallBack := ""

  /*
  * Optional. If a non-variable template is used, assign an empty value to this parameter, for example, string templateParas = "";
  * Example of a single-variable template: If the template content is "Your verification code is ${NUM_6}", templateParas can be set to '["369751"]'.
  * Example of a dual-variable template: If the template content is "You have ${NUM_2} delivered to ${TXT_20}", templateParas can be set to '["3","main gate of People's Park"]'.
  * To view more information, choose Service Overview > Template and Variable Specifications.
  */
  templateParas := "[\"369751\"]" // Template variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

  body := buildRequestBody(sender, receiver, templateId, templateParas, statusCallBack, signature)
  resp, err := post(apiAddress, []byte(body), appInfo)

  if err != nil {
    return
  }
  fmt.Println(resp)
}

/**
* sender, receiver, and templateId cannot be empty.
 */
func buildRequestBody(sender, receiver, templateId, templateParas, statusCallBack, signature string) string {
```



```
  param := "from=" + url.QueryEscape(sender) + "&to=" + url.QueryEscape(receiver) + "&templateId=" +
url.QueryEscape(templateId)
  if templateParas != "" {
    param += "&templateParas=" + url.QueryEscape(templateParas)
  }
  if statusCallBack != "" {
    param += "&statusCallback=" + url.QueryEscape(statusCallBack)
  }
  if signature != "" {
    param += "&signature=" + url.QueryEscape(signature)
  }
  return param
}

func post(url string, param []byte, appInfo core.Signer) (string, error) {
  if param == nil || appInfo == (core.Signer{}) {
    return "", nil
  }

  // In the code example, certificate verification is disabled. You need to enable certificate verification in the
commercial environment.
  tr := &http.Transport{
    TLSClientConfig: &tls.Config{InsecureSkipVerify: true},
  }
  client := &http.Client{Transport: tr}

  req, err := http.NewRequest("POST", url, bytes.NewBuffer(param))
  if err != nil {
    return "", err
  }

  // Add the content format to the request. The header is fixed.
  req.Header.Add("Content-Type", "application/x-www-form-urlencoded")
  // Sign the request using the HMAC algorithm and add the signing result to the Authorization header.
  appInfo.Sign(req)

  fmt.Println(req.Header)
  // Send an SMS request.
  resp, err := client.Do(req)
  if err != nil {
    fmt.Println(err)
  }

  // Response to obtaining the SMS message
  defer resp.Body.Close()
  body, err := ioutil.ReadAll(resp.Body)
  if err != nil {
    return "", err
  }
  return string(body), nil
}
```

Example of Sending SMSs in Batches

```
package main

import (
  "bytes"
  "crypto/tls"
  "encoding/json"
  "fmt"
  core "huaweicloud.com/apig/signer"
  "io/ioutil"
  "net/http"
)

func main() {
    // Mandatory. The values here are example values only. Obtain the actual values based on
Development Preparation.
```



```
  apiAddress := "https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendDiffSms/v1" // Application access address (obtain it from the Application Management page on the console) and API access URI.
  // Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
  appKey := "c8RWg3ggEcyd4D3p94bf3Y7x1Ile"                    //APP_Key
  appSecret := "q4Ii87Bh**********80SfD7Al"                  //APP_Secret
  sender := "csms12345678"                                   // Channel number for Chinese mainland SMS or international SMS
  templateId1 := "8ff55eac1d0b478ab3c06c3c6a492300"          // Template ID1
  templateId2 := "8ff55eac1d0b478ab3c06c3c6a492300"          // Template ID 2

  // Mandatory for Chinese mainland SMS. This parameter is valid and mandatory when Template Type corresponding to templateId is Universal template. The signature name must be approved and be the same type as the template.
  // This parameter is not required for international SMS.
  signature1 := "Huawei Cloud SMS test" // Signature 1
  signature2 := "Huawei Cloud SMS test" // Signature 2

  // Mandatory. Global number format (including the country code), for example, +8615123456789. Use commas (,) to separate multiple numbers.
  receiver1 := []string{"+86151****6789", "+86152****7890"} // Recipient number of template 1
  receiver2 := []string{"+86151****6789", "+86152****7890"} // Recipient number of template 2

  // Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
  statusCallBack := ""

  /**
   * Optional. If a non-variable template is used, assign an empty value to this parameter, for example, templateParas := []string{}.
   * Example of a single-variable template: If the template content is "Your verification code is ${NUM_6}", templateParas can be set to []string{"369751"}.
   * Example of a dual-variable template: If the template content is "You have ${NUM_2} delivered to ${TXT_20}", templateParas can be set to []string{"3","main gate of People's Park"}.
   * ${DATE}${TIME} cannot be empty. You can use spaces or dots (.) to replace the empty value of ${TXT_20} and use 0 to replace the empty value of ${NUM_6}.
   * To view more information, choose Service Overview > Template and Variable Specifications.
   */
  templateParas1 := []string{"123456"} // Template 1 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).
  templateParas2 := []string{"234567"} // Template 2 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

  item1 := initDiffSms(receiver1, templateId1, templateParas1, signature1)
  item2 := initDiffSms(receiver2, templateId2, templateParas2, signature2)

  item := []map[string]interface{}{item1, item2}
  body := buildRequestBody(sender, item, statusCallBack)

  appInfo := core.Signer{
    Key:    appKey, //App Key
    Secret: appSecret, //App Secret
  }
  resp, err := post(apiAddress, []byte(body), appInfo)
  if err != nil {
    return
  }
  fmt.Println(resp)
}

func buildRequestBody(sender string, item []map[string]interface{}, statusCallBack string) []byte {
  body := make(map[string]interface{})
  body["smsContent"] = item
  body["from"] = sender
```



```
  if statusCallBack != "" {
    body["statusCallback"] = statusCallBack
  }
  res, _ := json.Marshal(body)
  return res
}

func initDiffSms(reveiver []string, templateId string, templateParas []string, signature string)
map[string]interface{} {
  diffSms := make(map[string]interface{})
  diffSms["to"] = reveiver
  diffSms["templateId"] = templateId
  if templateParas != nil && len(templateParas) > 0 {
    diffSms["templateParas"] = templateParas
  }
  if signature != "" {
    diffSms["signature"] = signature
  }
  return diffSms
}

func post(url string, param []byte, appInfo core.Signer) (string, error) {
  if param == nil || appInfo == (core.Signer{}) {
    return "", nil
  }

  // In the code example, certificate verification is disabled. You need to enable certificate verification in the
commercial environment.
  tr := &http.Transport{
    TLSClientConfig: &tls.Config{InsecureSkipVerify: true},
  }
  client := &http.Client{Transport: tr}

  req, err := http.NewRequest("POST", url, bytes.NewBuffer(param))
  if err != nil {
    return "", err
  }

  // Add the content format to the request. The header is fixed.
  req.Header.Add("Content-Type", "application/json")
  // Sign the request using the HMAC algorithm and add the signing result to the Authorization header.
  appInfo.Sign(req)

  fmt.Println(req.Header)
  // Send an SMS request.
  resp, err := client.Do(req)
  if err != nil {
    fmt.Println(err)
  }

  // Response to obtaining the SMS message
  defer resp.Body.Close()
  body, err := ioutil.ReadAll(resp.Body)
  if err != nil {
    return "", err
  }
  return string(body), nil
}
```

Receiving Status Reports

```
package main

import (
    "fmt"
    "net/url"
    "strings"
)
```



```
func main() {
    // Data example (urlencode) of the status report reported by the SMS platform
    //success_body :=
"sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId=2ea20735-
f856-4376-afbf-570bd70a46ee_11840135&status=DELIVRD";
    failed_body :=
"orgCode=E200027&sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId
=2ea20735-f856-4376-afbf-570bd70a46ee_11840135&status=RTE_ERR";
    //onSmsStatusReport(success_body);
    onSmsStatusReport(failed_body);
}

func onSmsStatusReport(data string) {
    ss, _ := url.QueryUnescape(data)
    params := strings.Split(ss, "&")
    keyValues := make(map[string]string)
    for i := range params {
        temp := strings.Split(params[i],"=")
        keyValues[temp[0]] = temp[1];
    }
    /**
    * Example: Parsing status is used as an example.
    *
    * 'smsMsgId': Unique ID of an SMS
    * 'total': Number of SMS segments
    * 'sequence': Sequence number of an SMS after splitting
    * 'source': Status report source
    * 'updateTime': Resource update time
    * 'status': Enumeration values of the status report
    * 'orgCode': Status code
    */
    status := keyValues["status"];
    if status == "DELIVRD" {
        fmt.Println("Send sms success. smsMsgId: " + keyValues["smsMsgId"])
    } else {
        fmt.Println("Send sms failed. smsMsgId: " + keyValues["smsMsgId"])
        fmt.Println("Failed status:  " + keyValues["status"])
        fmt.Println("Failed orgCode:  " + keyValues["orgCode"])
    }
}
```

3.2 X-WSSE Authentication

3.2.1 Java

Example	**Sending SMSs**, **Sending SMSs in Batches** **Receiving Status Reports**
Environment	JDK 1.6 or later
Library	**httpclient**, **httpcore**, **commons-codec**, **commons-logging**, **org.json**

NOTICE

- Sending SMSs shows an example of sending group SMSs with a single template. Sending SMSs in batches shows an example of sending group SMSs with multiple templates.
- The examples described in this document may involve the use of personal data. You are advised to comply with relevant laws and regulations and take measures to ensure that personal data is fully protected.
- The examples described in this document are for demonstration purposes only. Commercial use of the examples is prohibited.
- Information in this document is for your reference only. It does not constitute any offer or commitment.

Sending SMSs

```
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.UnsupportedEncodingException;
import java.io.Writer;
import java.net.URL;
import java.net.URLEncoder;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;
import java.text.SimpleDateFormat;
// If the JDK version is 1.8, use the native Base64 class.
import java.util.Base64;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;

import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;

// If the JDK version is earlier than 1.8, use a third-party library to provide the Base64 class.
//import org.apache.commons.codec.binary.Base64;

public class SendSms {

    // This parameter does not need to be modified. It is used to format the Authentication header field and assign a value to the X-WSSE parameter.
    private static final String WSSE_HEADER_FORMAT = "UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s\"";
    // This parameter does not need to be modified. It is used to format the Authentication header field and assign a value to the Authorization parameter.
    private static final String AUTH_HEADER_VALUE = "WSSE realm=\"SDP\",profile=\"UsernameToken\",type=\"Appkey\"";

    public static void main(String[] args) throws Exception {

        // Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
        String url = "https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendSms/v1"; // Application access address and API access URI
        // Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and
```



```
store them in the configuration file or environment variables.
        String appKey = "gHWYq2Jte2LnUjphSI51DW70vq4a"; //Application Key
        String appSecret = "q4Ii87Bh**********80SfD7Al"; //Application Secret
        String sender = "888888888090"; // International SMS channel number
        String templateId = "8347faad35024918bf16673556fe1b54"; // Template ID

        // Mandatory. Global number format (including the country code), for example, +9100****11. Use
commas (,) to separate multiple numbers.
        String receiver = "+9100****11"; // Recipient number

        // Optional. Address for receiving SMS status reports. The domain name is recommended. If this
parameter is set to an empty value or left unspecified, customers do not receive status reports.
        String statusCallBack = "";

        /**
         * Optional. If a non-variable template is used, assign an empty value to this parameter, for example,
String templateParas = "";
         * Example of a single-variable template: If the template content is "Your verification code is $
{NUM_6}", templateParas can be set to "[\"369751\"]".
         * Example of a dual-variable template: If the template content is "You have ${NUM_2} parcel(s)
delivered to ${TXT_20}", templateParas can be set to "[\"3\",\"main gate of People's Park\"]".
         * To view more information, choose Service Overview > Template and Variable Specifications.
         */
        String templateParas = "[\"369751\"]"; // Template variable. The following uses a single-variable
verification code SMS message as an example. The customer needs to generate a 6-digit verification code
and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to
2569).

        // Request body
        String body = buildRequestBody(sender, receiver, templateId, templateParas, statusCallBack);
        if (null == body || body.isEmpty()) {
            System.out.println("body is null.");
            return;
        }

        // The value of X-WSSE in the request headers is as follows:
        String wsseHeader = buildWsseHeader(appKey, appSecret);
        if (null == wsseHeader || wsseHeader.isEmpty()) {
            System.out.println("wsse header is null.");
            return;
        }

        Writer out = null;
        BufferedReader in = null;
        StringBuilder result = new StringBuilder();
        HttpsURLConnection connection;
        InputStream is = null;

        // Ignore the certificate trust issues to prevent API calling failures caused by HTTPS certificate
authentication failures.
        HostnameVerifier hv = new HostnameVerifier() {

            @Override
            public boolean verify(String hostname, SSLSession session) {
                return true;
            }
        };
        trustAllHttpsCertificates();

        try {
            URL realUrl = new URL(url);
            connection = (HttpsURLConnection) realUrl.openConnection();

            connection.setHostnameVerifier(hv);
            connection.setDoOutput(true);
            connection.setDoInput(true);
            connection.setUseCaches(true);
             // Request method
            connection.setRequestMethod("POST");
```



```
        // Parameters in the request headers
        connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
        connection.setRequestProperty("Authorization", AUTH_HEADER_VALUE);
        connection.setRequestProperty("X-WSSE", wsseHeader);

        connection.connect();
        out = new OutputStreamWriter(connection.getOutputStream());
        out.write(body); // Parameters in the request body
        out.flush();
        out.close();

        int status = connection.getResponseCode();
        if (200 == status) { //200
            is = connection.getInputStream();
        } else { //400/401
            is = connection.getErrorStream();
        }
        in = new BufferedReader(new InputStreamReader(is, "UTF-8"));
        String line = "";
        while ((line = in.readLine()) != null) {
            result.append(line);
        }
        System.out.println(result); // Record the response entity.
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (null != out) {
                out.close();
            }
            if (null != is) {
                is.close();
            }
            if (null != in) {
                in.close();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

/**
 * Construct the request body.
 *
 * @param sender
 * @param receiver
 * @param templateId
 * @param templateParas
 * @param statusCallBack
 * @return
 */
static String buildRequestBody(String sender, String receiver, String templateId, String templateParas,
                               String statusCallBack) {
    if (null == receiver || null == templateId || receiver.isEmpty()
            || templateId.isEmpty()) {
        System.out.println("buildRequestBody(): sender, receiver or templateId is null.");
        return null;
    }
    Map<String, String> map = new HashMap<String, String>();

    map.put("from", sender);
    map.put("to", receiver);
    map.put("templateId", templateId);
    if (null != templateParas && !templateParas.isEmpty()) {
        map.put("templateParas", templateParas);
    }
    if (null != statusCallBack && !statusCallBack.isEmpty()) {
        map.put("statusCallback", statusCallBack);
```



```
        }

        StringBuilder sb = new StringBuilder();
        String temp = "";

        for (String s : map.keySet()) {
            try {
                temp = URLEncoder.encode(map.get(s), "UTF-8");
            } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
            }
            sb.append(s).append("=").append(temp).append("&");
        }

        return sb.deleteCharAt(sb.length() - 1).toString();
    }

    /**
     * Construct the value of X-WSSE.
     *
     * @param appKey
     * @param appSecret
     * @return
     */
    static String buildWsseHeader(String appKey, String appSecret) {
        if (null == appKey || null == appSecret || appKey.isEmpty() || appSecret.isEmpty()) {
            System.out.println("buildWsseHeader(): appKey or appSecret is null.");
            return null;
        }
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'");
        String time = sdf.format(new Date()); //Created
        String nonce = UUID.randomUUID().toString().replace("-", ""); //Nonce

        MessageDigest md;
        byte[] passwordDigest = null;

        try {
            md = MessageDigest.getInstance("SHA-256");
            md.update((nonce + time + appSecret).getBytes());
            passwordDigest = md.digest();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }

        // If the JDK version is 1.8, load the native Base64 class and use the following code:
        String passwordDigestBase64Str = Base64.getEncoder().encodeToString(passwordDigest); //
PasswordDigest
        // If the JDK version is earlier than 1.8, load a third-party library to provide the Base64 class and use
the following code:
        //String passwordDigestBase64Str = Base64.encodeBase64String(passwordDigest); //PasswordDigest
        // If passwordDigestBase64Str contains newline characters, run the following code:
        //passwordDigestBase64Str = passwordDigestBase64Str.replaceAll("[\\s*\t\n\r]", "");
        return String.format(WSSE_HEADER_FORMAT, appKey, passwordDigestBase64Str, nonce, time);
    }

    /**
     * Ignore the certificate trust issues.
     *
     * @throws Exception
     */
    static void trustAllHttpsCertificates() throws Exception {
        TrustManager[] trustAllCerts = new TrustManager[]{
                new X509TrustManager() {
                    public void checkClientTrusted(X509Certificate[] chain, String authType) throws
CertificateException {
                        return;
                    }

                    public void checkServerTrusted(X509Certificate[] chain, String authType) throws
```



```
CertificateException {
                    return;
                }

                public X509Certificate[] getAcceptedIssuers() {
                    return null;
                }
            }
        };
        SSLContext sc = SSLContext.getInstance("SSL");
        sc.init(null, trustAllCerts, null);
        HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());
    }
}
```

Sending SMSs in Batches

```
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.UnsupportedEncodingException;
import java.io.Writer;
import java.net.URL;
import java.net.URLEncoder;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;
import java.text.SimpleDateFormat;
// If the JDK version is 1.8, use the native Base64 class.
import java.util.Base64;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;

import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;

// If the JDK version is earlier than 1.8, use a third-party library to provide the Base64 class.
//import org.apache.commons.codec.binary.Base64;

public class SendSms {

    // This parameter does not need to be modified. It is used to format the Authentication header field and assign a value to the X-WSSE parameter.
    private static final String WSSE_HEADER_FORMAT = "UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s\"";
    // This parameter does not need to be modified. It is used to format the Authentication header field and assign a value to the Authorization parameter.
    private static final String AUTH_HEADER_VALUE = "WSSE realm=\"SDP\",profile=\"UsernameToken\",type=\"Appkey\"";

    public static void main(String[] args) throws Exception {

        // Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
        String url = "https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendSms/v1"; // Application access address and API access URI
        // Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
        String appKey = "gHWYq2Jte2LnUjphSI51DW70vq4a"; //Application Key
        String appSecret ="QCMVfZ5t**********7Qksb6JB"; //Application Secret
        String sender = "888888888090"; // International SMS channel number
```



```
        String templateId = "8347faad35024918bf16673556fe1b54"; // Template ID

        // Mandatory. Global number format (including the country code), for example, +9100****11. Use commas (,) to separate multiple numbers.
        String receiver = "+9100****11"; // Recipient number

        // Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
        String statusCallBack = "";

        /**
         *Optional. If a non-variable template is used, assign an empty value to this parameter, for example, String templateParas = "";
         *Example of a single-variable template: If the template content is "Your verification code is ${NUM_6}", templateParas can be set to "[\"369751\"]".
         *Example of a dual-variable template: If the template content is "You have ${NUM_2} parcel(s) delivered to ${TXT_20}", templateParas can be set to "[\"3\",\"main gate of People's Park\"]".
         * To view more information, choose Service Overview > Template and Variable Specifications.
         */
        String templateParas = "[\"369751\"]"; // Template variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

        // Request body
        String body = buildRequestBody(sender, receiver, templateId, templateParas, statusCallBack);
        if (null == body || body.isEmpty()) {
            System.out.println("body is null.");
            return;
        }

        // The value of X-WSSE in the request headers is as follows:
        String wsseHeader = buildWsseHeader(appKey, appSecret);
        if (null == wsseHeader || wsseHeader.isEmpty()) {
            System.out.println("wsse header is null.");
            return;
        }

        Writer out = null;
        BufferedReader in = null;
        StringBuilder result = new StringBuilder();
        HttpsURLConnection connection;
        InputStream is = null;

        // Ignore the certificate trust issues to prevent API calling failures caused by HTTPS certificate authentication failures.
        HostnameVerifier hv = new HostnameVerifier() {

            @Override
            public boolean verify(String hostname, SSLSession session) {
                return true;
            }
        };
        trustAllHttpsCertificates();

        try {
            URL realUrl = new URL(url);
            connection = (HttpsURLConnection) realUrl.openConnection();

            connection.setHostnameVerifier(hv);
            connection.setDoOutput(true);
            connection.setDoInput(true);
            connection.setUseCaches(true);
             // Request method
            connection.setRequestMethod("POST");
            // Parameters in the request headers
            connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
            connection.setRequestProperty("Authorization", AUTH_HEADER_VALUE);
            connection.setRequestProperty("X-WSSE", wsseHeader);
```



```
        connection.connect();
        out = new OutputStreamWriter(connection.getOutputStream());
        out.write(body); // Parameters in the request body
        out.flush();
        out.close();

        int status = connection.getResponseCode();
        if (200 == status) { //200
            is = connection.getInputStream();
        } else { //400/401
            is = connection.getErrorStream();
        }
        in = new BufferedReader(new InputStreamReader(is, "UTF-8"));
        String line = "";
        while ((line = in.readLine()) != null) {
            result.append(line);
        }
        System.out.println(result); // Record the response entity.
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (null != out) {
                out.close();
            }
            if (null != is) {
                is.close();
            }
            if (null != in) {
                in.close();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

/**
 * Construct the request body.
 *
 * @param sender
 * @param receiver
 * @param templateId
 * @param templateParas
 * @param statusCallBack
 * @return
 */
static String buildRequestBody(String sender, String receiver, String templateId, String templateParas,
                               String statusCallBack) {
    if (null == receiver || null == templateId || receiver.isEmpty()
            || templateId.isEmpty()) {
        System.out.println("buildRequestBody(): sender, receiver or templateId is null.");
        return null;
    }
    Map<String, String> map = new HashMap<String, String>();

    map.put("from", sender);
    map.put("to", receiver);
    map.put("templateId", templateId);
    if (null != templateParas && !templateParas.isEmpty()) {
        map.put("templateParas", templateParas);
    }
    if (null != statusCallBack && !statusCallBack.isEmpty()) {
        map.put("statusCallback", statusCallBack);
    }

    StringBuilder sb = new StringBuilder();
    String temp = "";
```



```
        for (String s : map.keySet()) {
            try {
                temp = URLEncoder.encode(map.get(s), "UTF-8");
            } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
            }
            sb.append(s).append("=").append(temp).append("&");
        }

        return sb.deleteCharAt(sb.length() - 1).toString();
    }

    /**
     * Construct the value of X-WSSE.
     *
     * @param appKey
     * @param appSecret
     * @return
     */
    static String buildWsseHeader(String appKey, String appSecret) {
        if (null == appKey || null == appSecret || appKey.isEmpty() || appSecret.isEmpty()) {
            System.out.println("buildWsseHeader(): appKey or appSecret is null.");
            return null;
        }
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'");
        String time = sdf.format(new Date()); //Created
        String nonce = UUID.randomUUID().toString().replace("-", ""); //Nonce

        MessageDigest md;
        byte[] passwordDigest = null;

        try {
            md = MessageDigest.getInstance("SHA-256");
            md.update((nonce + time + appSecret).getBytes());
            passwordDigest = md.digest();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }

        // If the JDK version is 1.8, load the native Base64 class and use the following code:
        String passwordDigestBase64Str = Base64.getEncoder().encodeToString(passwordDigest); //
PasswordDigest
        // If the JDK version is earlier than 1.8, load a third-party library to provide the Base64 class and use
the following code:
        //String passwordDigestBase64Str = Base64.encodeBase64String(passwordDigest); //PasswordDigest
        // If passwordDigestBase64Str contains newline characters, run the following code:
        //passwordDigestBase64Str = passwordDigestBase64Str.replaceAll("[\\s*\t\n\r]", "");
        return String.format(WSSE_HEADER_FORMAT, appKey, passwordDigestBase64Str, nonce, time);
    }

    /**
     * Ignore the certificate trust issues.
     *
     * @throws Exception
     */
    static void trustAllHttpsCertificates() throws Exception {
        TrustManager[] trustAllCerts = new TrustManager[]{
                new X509TrustManager() {
                    public void checkClientTrusted(X509Certificate[] chain, String authType) throws
CertificateException {
                        return;
                    }

                    public void checkServerTrusted(X509Certificate[] chain, String authType) throws
CertificateException {
                        return;
                    }
```



```
                public X509Certificate[] getAcceptedIssuers() {
                    return null;
                }
            }
        };
        SSLContext sc = SSLContext.getInstance("SSL");
        sc.init(null, trustAllCerts, null);
        HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());
    }
}
```

Receiving Status Reports

The Maven dependency to be introduced is org.springframework:spring-web:5.3.21 (example version).

```
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class DemoController {
    /**
     * Synchronize SMS receipts.
     */
    @PostMapping("/report")
    public void smsHwReport(@RequestParam String smsMsgId,  // Unique SMS identifier returned after an SMS is successfully sent.
                            @RequestParam(required = false) String total,     // Number of SMS segments split from a long SMS message. If the SMS message is not split, set this parameter to 1.
                            @RequestParam(required = false) String sequence,  // Sequence number after a long SMS is split. This parameter is valid only when the value of total is greater than 1. If the SMS is not split, set this parameter to 1.
                            @RequestParam String status,    // Enumerated values of an SMS status report. For details about the values, see API Reference.
                            @RequestParam(required = false) String source,    // Source of the SMS status report. 1: generated by the SMS platform. 2: returned by the SMSC. 3: generated by the cloud platform.
                            @RequestParam(required = false) String updateTime,// SMS resource update time, which is generally the UTC time when the Message & SMS platform receives the SMS status report. The value is in the format of yyyy-MM-dd'T'HH:mm:ss'Z'. The time is converted to %3a using urlencode. // When the Message & SMS platform does not receive the SMS status report from the SMSC, the platform constructs a status report that does not contain the updateTime parameter.
                            @RequestParam(required = false) String orgCode,   // Status codes of southbound NEs are transparently transmitted. // When the status code is not returned, the parameter is not used.
                            @RequestParam(required = false) String extend,    // Extended field in the request sent by a user. If the SMS sent by a user does not carry the extend parameter, the status report does not contain the extend parameter.
                            @RequestParam(required = false) String to) {      // Recipient number of the SMS corresponding to the status report. This parameter is carried only when the status report contains the extend parameter.
        System.out.println("  ================receive smsStatusReport ======================");
        System.out.println("smsMsgId: " + smsMsgId);
        System.out.println("total: " + total);
        System.out.println("sequence: " + sequence);
        System.out.println("status: " + status);
        System.out.println("source: " + source);
        System.out.println("updateTime: " + updateTime);
        System.out.println("orgCode: " + orgCode);
        System.out.println("extend: " + extend);
        System.out.println("to: " + to);
    }
}
```

References

- Code examples: **PHP**, **Python**, **C#**, **Node.js**, and **Go**

- APIs: SMS Sending API, Batch SMS Sending API, and Status Report Receiving API

3.2.2 PHP

Example	**Sending SMSs (Example 1)**, **Sending SMSs (Example 2)** **Sending SMSs in Batches (Example 1)**, **Sending SMSs in Batches (Example 2)** **Receiving Status Reports**
Environment	PHP 7.0 or later versions are required. The examples are developed based on PHP 7.2.9.
Dependency Configuration	Composer and Guzzle (only for example 1) 1. Install and configure the Composer by following the instructions provided in **https://getcomposer.org/download/**. 2. Create the **composer.json** file in the project folder. The content of the file is as follows: `{ "require": { "guzzlehttp/guzzle": "~6.0@dev" } }` 3. Open the CLI window, enter the project folder, and run **composer install** to install Guzzle. Note: After the installation is complete, the **vendor/autoload.php** file is automatically generated in the project folder.

NOTICE

- Sending SMSs shows an example of sending group SMSs with a single template. Sending SMSs in batches shows an example of sending group SMSs with multiple templates.
- The examples described in this document may involve the use of personal data. You are advised to comply with relevant laws and regulations and take measures to ensure that personal data is fully protected.
- The examples described in this document are for demonstration purposes only. Commercial use of the examples is prohibited.
- Information in this document is for your reference only. It does not constitute any offer or commitment.

Sending SMSs (Example 1)

```
<?php
require 'vendor/autoload.php'; //This file is automatically generated when composer install is executed.

use GuzzleHttp\Psr7;
use GuzzleHttp\Client;
```

```
use GuzzleHttp\Exception\RequestException;

// Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
$url = 'https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendSms/v1'; // Application access address (obtain it from the Application Management page on the console) and API access URI.
// Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
$APP_KEY = 'c8RWg3ggEcyd4D3p94bf3Y7x1Ile'; //APP_Key
$APP_SECRET = 'q4Ii87Bh**********80SfD7Al'; //APP_Secret
$sender = 'csms12345678'; // SMS channel number.
$TEMPLATE_ID = '8ff55eac1d0b478ab3c06c3c6a492300'; // Template ID

// Mandatory. Global number format (including the country code), for example, +9100****11. Use commas (,) to separate multiple numbers.
$receiver = '9100****11,9100****12'; // Recipient numbers

// Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
$statusCallback = '';

/**
 * Optional. If a template with no variable is used, assign an empty value to this parameter, for example, $TEMPLATE_PARAS = '';
 * Example of a single-variable template: If the template content is "Your verification code is ${1}", $TEMPLATE_PARAS can be set to '["369751"]'.
 * Example of a dual-variable template: If the template content is "You have ${1} parcel delivered to ${2}", $TEMPLATE_PARAS can be set to '["3","main gate of People's Park"]'.
 * Each variable in the template must be assigned a value, and the value cannot be empty.
 * To view more information, choose Service Overview > Template and Variable Specifications.
 * @var string $TEMPLATE_PARAS
 */
$TEMPLATE_PARAS = '["369751"]'; // Template variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

$client = new Client();
try {
    $response = $client->request('POST', $url, [
        'form_params' => [
            'from' => $sender,
            'to' => $receiver,
            'templateId' => $TEMPLATE_ID,
            'templateParas' => $TEMPLATE_PARAS,
            'statusCallback' => $statusCallback
        ],
        'headers' => [
            'Authorization' => 'WSSE realm="SDP",profile="UsernameToken",type="Appkey"',
            'X-WSSE' => buildWsseHeader($APP_KEY, $APP_SECRET)
        ],
        'verify' => false // Ignore the certificate trust issues to prevent API calling failures caused by HTTPS certificate authentication failures.
    ]);
    echo Psr7\str($response); // Record the response data.
} catch (RequestException $e) {
    echo $e;
    echo Psr7\str($e->getRequest()), "\n";
    if ($e->hasResponse()) {
        echo Psr7\str($e->getResponse());
    }
}

/**
 * Construct the value of X-WSSE.
 * @param string $appKey
 * @param string $appSecret
 * @return string
 */
function buildWsseHeader(string $appKey, string $appSecret){
```



```
    $now = date('Y-m-d\TH:i:s\Z'); //Created
    $nonce = uniqid(); //Nonce
    $base64 = base64_encode(hash('sha256', ($nonce . $now . $appSecret))); //PasswordDigest
    return sprintf("UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s\"",
        $appKey, $base64, $nonce, $now);
}
?>
```

Sending SMSs (Example 2)

```
<?php
// Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
$url = 'https://smsapi.cn-north-4.myhuaweicloud.com:443/sms/batchSendSms/v1'; // Application access address (obtain it from the Application Management page on the console) and API access URI.
// Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
$APP_KEY = 'c8RWg3ggEcyd4D3p94bf3Y7x1Ile'; //APP_Key
$APP_SECRET = 'q4Ii87Bh**********80SfD7Al'; //APP_Secret
$sender = 'csms12345678'; // SMS channel number.
$TEMPLATE_ID = '8ff55eac1d0b478ab3c06c3c6a492300'; // Template ID

// Mandatory. Global number format (including the country code), for example, +44020****6789. Use commas (,) to separate multiple numbers.
$receiver = '+44020****6789,+44021****7890'; // Recipient numbers

// Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
$statusCallback = '';

/**
 * Optional. If a template with no variable is used, assign an empty value to this parameter, for example, $TEMPLATE_PARAS = '';
 * Example of a single-variable template: If the template content is "Your verification code is ${1}", $TEMPLATE_PARAS can be set to '["369751"]'.
 * Example of a dual-variable template: If the template content is "You have ${1} parcel delivered to ${2}", $TEMPLATE_PARAS can be set to '["3","main gate of People's Park"]'.
 * Each variable in the template must be assigned a value, and the value cannot be empty.
 * To view more information, choose Service Overview > Template and Variable Specifications.
 * @var string $TEMPLATE_PARAS
 */
$TEMPLATE_PARAS = '["369751"]'; // Template variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

// Request headers
$headers = [
    'Content-Type: application/x-www-form-urlencoded',
    'Authorization: WSSE realm="SDP",profile="UsernameToken",type="Appkey"',
    'X-WSSE: ' . buildWsseHeader($APP_KEY, $APP_SECRET)
];
// Request body
$data = http_build_query([
    'from' => $sender,
    'to' => $receiver,
    'templateId' => $TEMPLATE_ID,
    'templateParas' => $TEMPLATE_PARAS,
    'statusCallback' => $statusCallback
]);

$context_options = [
    'http' => ['method' => 'POST', 'header'=> $headers, 'content' => $data, 'ignore_errors' => true],
    'ssl' => ['verify_peer' => false, 'verify_peer_name' => false] // Ignore the certificate trust issues to prevent API calling failures caused by HTTPS certificate authentication failures.
];
print_r($context_options) . PHP_EOL; // Record the request.

$response = file_get_contents($url, false, stream_context_create($context_options));
print_r($response) . PHP_EOL; // Record the response.
```



```
/**
 * Construct the value of X-WSSE.
 * @param string $appKey
 * @param string $appSecret
 * @return string
 */
function buildWsseHeader(string $appKey, string $appSecret){
    date_default_timezone_set('Asia/Shanghai');
    $now = date('Y-m-d\TH:i:s\Z'); //Created
    $nonce = uniqid(); //Nonce
    $base64 = base64_encode(hash('sha256', ($nonce . $now . $appSecret))); //PasswordDigest
    return sprintf("UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s\"",
        $appKey, $base64, $nonce, $now);
}
?>
```

Sending SMSs in Batches (Example 1)

```
<?php
require 'vendor/autoload.php'; // This file is automatically generated when composer install is executed.

use GuzzleHttp\Psr7;
use GuzzleHttp\Client;
use GuzzleHttp\Exception\RequestException;

// Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
$url = 'https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendDiffSms/v1'; // Application access address (obtain it from the Application Management page on the console) and API access URI.
// Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
$APP_KEY = 'c8RWg3ggEcyd4D3p94bf3Y7x1Ile'; //APP_Key
$APP_SECRET = 'q4Ii87Bh**********80SfD7Al'; //APP_Secret
$sender = 'csms12345678'; // SMS channel number.
$TEMPLATE_ID_1 = '8ff55eac1d0b478ab3c06c3c6a492300'; // Template ID 1
$TEMPLATE_ID_2 = '8ff55eac1d0b478ab3c06c3c6a492300'; // Template ID 2

// Mandatory. Global number format (including the country code), for example, +9100****11. Use commas (,) to separate multiple numbers.
$receiver_1 = ['+9100****11', '+9100****12']; // Recipient number of template 1
$receiver_2 = ['+9100****13', '+9100****14']; // Recipient number of template 2

// Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
$statusCallback = '';

/**
 * Optional. If a template with no variable is used, assign an empty value to this parameter, for example, $TEMPLATE_PARAS = [];
 * Example of a single-variable template: If the template content is "Your verification code is ${1}", $TEMPLATE_PARAS can be set to ['369751'].
 * Example of a dual-variable template: If the template content is "You have ${1} parcel delivered to ${2}", $TEMPLATE_PARAS can be set to ['3','main gate of People's Park'].
 * Each variable in the template must be assigned a value, and the value cannot be empty.
 * To view more information, choose Service Overview > Template and Variable Specifications.
 */
$TEMPLATE_PARAS_1 = ['123456']; // Template 1 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).
$TEMPLATE_PARAS_2 = ['234567']; // Template 2 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

// Mandatory. Set variables based on the number of template IDs.
$smsContent = [
    initDiffSms($receiver_1, $TEMPLATE_ID_1, $TEMPLATE_PARAS_1),
    initDiffSms($receiver_2, $TEMPLATE_ID_2, $TEMPLATE_PARAS_2)
];
```



```
$client = new Client();
try {
    $response = $client->request('POST', $url, [
        'body' => json_encode([
            'from' => $sender,
            'statusCallback' => $statusCallback,
            'smsContent' => $smsContent
        ]),
        'headers' => [
            'Authorization' => 'WSSE realm="SDP",profile="UsernameToken",type="Appkey"',
            'X-WSSE' => buildWsseHeader($APP_KEY, $APP_SECRET),
            'Content-Type' => 'application/json'
        ],
        'verify' => false // Ignore the certificate trust issues to prevent API calling failures caused by HTTPS certificate authentication failures.
    ]);
    echo Psr7\str($response); // Record the response data.
} catch (RequestException $e) {
    echo $e;
    echo Psr7\str($e->getRequest()), "\n";
    if ($e->hasResponse()) {
        echo Psr7\str($e->getResponse());
    }
}

/**
 * Construct the value of smsContent.
 * @param array $receiver
 * @param string $templateId
 * @param array $templateParas
 * @return string[]
 */
function initDiffSms(array $receiver, string $templateId, array $templateParas) {
    return ['to' => $receiver, 'templateId' => $templateId, 'templateParas' => $templateParas];
}

/**
 * Construct the value of X-WSSE.
 * @param string $appKey
 * @param string $appSecret
 * @return string
 */
function buildWsseHeader(string $appKey, string $appSecret){
    $now = date('Y-m-d\TH:i:s\Z'); //Created
    $nonce = uniqid();  //Nonce
    $base64 = base64_encode(hash('sha256', ($nonce . $now . $appSecret))); //PasswordDigest
    return sprintf("UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s\"",
        $appKey, $base64, $nonce, $now);
}
?>
```

Sending SMSs in Batches (Example 2)

```
<?php
// Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
$url = 'https://smsapi.cn-north-4.myhuaweicloud.com:443/sms/batchSendDiffSms/v1'; // Application access address (obtain it from the Application Management page on the console) and API access URI.
$APP_KEY = 'c8RWg3ggEcyd4D3p94bf3Y7x1Ile'; //APP_Key
// Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
$APP_SECRET = 'q4Ii87Bh**********80SfD7Al'; //APP_Secret
$sender = 'csms12345678'; // SMS channel number.
$TEMPLATE_ID_1 = '8ff55eac1d0b478ab3c06c3c6a492300'; // Template ID 1
$TEMPLATE_ID_2 = '8ff55eac1d0b478ab3c06c3c6a492300'; // Template ID 2

// Mandatory. Global number format (including the country code), for example, +9100****11. Use commas (,) to separate multiple numbers.
```



```
$receiver_1 = ['+9100****11', '+9100****12']; // Recipient number of template 1
$receiver_2 = ['+9100****13', '+9100****14']; // Recipient number of template 2

// Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
$statusCallback = '';

/**
 * Optional. If a template with no variable is used, assign an empty value to this parameter, for example, $TEMPLATE_PARAS = [];
 * Example of a single-variable template: If the template content is "Your verification code is ${1}", $TEMPLATE_PARAS can be set to ['369751'].
 * Example of a dual-variable template: If the template content is "You have ${1} parcel delivered to ${2}", $TEMPLATE_PARAS can be set to ['3','main gate of People's Park'].
 * Each variable in the template must be assigned a value, and the value cannot be empty.
 * To view more information, choose Service Overview > Template and Variable Specifications.
 */
$TEMPLATE_PARAS_1 = ['123456']; // Template 1 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).
$TEMPLATE_PARAS_2 = ['234567']; // Template 2 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

// Request headers
$headers = [
    'Content-Type: application/json',
    'Authorization: WSSE realm="SDP",profile="UsernameToken",type="Appkey"',
    'X-WSSE: ' . buildWsseHeader($APP_KEY, $APP_SECRET)
];
// Request body
$data = json_encode([
    'from' => $sender,
    'statusCallback' => $statusCallback,
    'smsContent' => [// Mandatory. Set the parameters based on the number of template IDs.
        initDiffSms($receiver_1, $TEMPLATE_ID_1, $TEMPLATE_PARAS_1),
        initDiffSms($receiver_2, $TEMPLATE_ID_2, $TEMPLATE_PARAS_2)
    ]
]);

$context_options = [
    'http' => ['method' => 'POST', 'header'=> $headers, 'content' => $data, 'ignore_errors' => true],
    'ssl' => ['verify_peer' => false, 'verify_peer_name' => false] // Ignore the certificate trust issues to prevent API calling failures caused by HTTPS certificate authentication failures.
];
print_r($context_options) . PHP_EOL; // Record the request.

$response = file_get_contents($url, false, stream_context_create($context_options));
print_r($response) . PHP_EOL; // Record the response.

/**
 * Construct the value of smsContent.
 * @param array $receiver
 * @param string $templateId
 * @param array $templateParas
 * @return string[]
 */
function initDiffSms(array $receiver, string $templateId, array $templateParas) {
    return ['to' => $receiver, 'templateId' => $templateId, 'templateParas' => $templateParas];
}

/**
 * Construct the value of X-WSSE.
 * @param string $appKey
 * @param string $appSecret
 * @return string
 */
function buildWsseHeader(string $appKey, string $appSecret){
    date_default_timezone_set('Asia/Shanghai');
```



```
    $now = date('Y-m-d\TH:i:s\Z'); //Created
    $nonce = uniqid(); //Nonce
    $base64 = base64_encode(hash('sha256', ($nonce . $now . $appSecret))); //PasswordDigest
    return sprintf("UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s\"",
        $appKey, $base64, $nonce, $now);
}
?>
```

Receiving Status Reports

```
<?php
// Data example (urlencode) of the status report reported by the SMS platform
// $success_body =
'sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId=2ea20735-
f856-4376-afbf-570bd70a46ee_11840135&status=DELIVRD';
$failed_body =
'orgCode=E200027&sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId
=2ea20735-f856-4376-afbf-570bd70a46ee_11840135&status=RTE_ERR';

// onSmsStatusReport($success_body);
onSmsStatusReport($failed_body);

/**
 * Parse the status report data.
 *
 * @param string $data: Status report data reported by the SMS platform
 */
function onSmsStatusReport(string $data)
{
    $keyValues = [];
    parse_str(urldecode($data), $keyValues); // Parse the status report data.

    /**
            * Example: Parsing status is used as an example.
     *
     * 'smsMsgId': Unique ID of an SMS
     * 'total': Number of SMS segments
     * 'sequence': Sequence number of an SMS after splitting
     * 'source': Status report source
     * 'updateTime': Resource update time
     * 'status': Enumeration values of the status report
     * 'orgCode': Status code
     */
    $status = $keyValues['status']; // Enumerated values of the status report
    // Check whether the SMS is successfully sent based on the value of status.
    if ('DELIVRD' === strtoupper($status)) {
        print 'Send sms success. smsMsgId: ' . $keyValues['smsMsgId'] . PHP_EOL;
    } else {
        // The SMS fails to be sent. The values of status and orgCode are recorded.
        print 'Send sms failed. smsMsgId: ' . $keyValues['smsMsgId'] . PHP_EOL;
        print 'Failed status: ' . $status . PHP_EOL;
        print 'Failed orgCode: ' . $keyValues['orgCode'] . PHP_EOL;
    }
}
?>
```

References

- Code examples: **Java**, **Python**, **C#**, **Node.js**, and **Go**
- APIs: SMS Sending API, Batch SMS Sending API, and Status Report Receiving API

3.2.3 Python

Example	**Sending SMSs (Example 1)**, **Sending SMSs (Example 2)** **Sending SMSs in Batches (Example 1)**, **Sending SMSs in Batches (Example 2)** **Receiving Status Reports**
Environment	Python 3.7 or later versions are required. The examples are developed based on Python 3.7.0.
Library	requests 2.18.1 (only for example 1) 1. Download and install Python 3.7 and configure the environment. 2. Open the CLI window and run **pip install requests**. 3. Run **pip list** to check the installation result.

NOTICE

- Sending SMSs shows an example of sending group SMSs with a single template. Sending SMSs in batches shows an example of sending group SMSs with multiple templates.
- The examples described in this document may involve the use of personal data. You are advised to comply with relevant laws and regulations and take measures to ensure that personal data is fully protected.
- The examples described in this document are for demonstration purposes only. Commercial use of the examples is prohibited.
- Information in this document is for your reference only. It does not constitute any offer or commitment.

Sending SMSs (Example 1)

```
# -*- coding: utf-8 -*-
import time
import uuid
import hashlib
import base64
import requests #Run pip install requests to install the dependency.

# Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
url = 'https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendSms/v1' #Application access address (obtain it from the Application Management page on the console) and API access URI.
// Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
APP_KEY = "c8RWg3ggEcyd4D3p94bf3Y7x1Ile" #APP_Key
APP_SECRET = "q4Ii87Bh**********80SfD7Al" #APP_Secret
sender = "csms12345678" #SMS channel number.
TEMPLATE_ID = "8ff55eac1d0b478ab3c06c3c6a492300" #Template ID

# Mandatory. Global number format (including the country code), for example, +9100****11. Use commas (,) to separate multiple numbers.
receiver = "+9100****11,+9100****12" #Recipient numbers
```



```
# Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
statusCallBack = ""

'''
Optional. If a template with no variable is used, assign an empty value to this parameter, for example, TEMPLATE_PARAM = '';
Example of a single-variable template: If the template content is "Your verification code is ${1}", TEMPLATE_PARAM can be set to '["369751"]'.
Example of a dual-variable template: If the template content is "You have ${1} parcel delivered to ${2}", TEMPLATE_PARAM can be set to '["3","main gate of People's Park"]'.
Each variable in the template must be assigned a value, and the value cannot be empty.
To view more information, choose Service Overview > Template and Variable Specifications.
'''
TEMPLATE_PARAM = '["369751"]' #Template variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

'''
Construct the value of X-WSSE.
@param appKey: string
@param appSecret: string
@return: string
'''
def buildWSSEHeader(appKey, appSecret):
    now = time.strftime('%Y-%m-%dT%H:%M:%SZ') #Created
    nonce = str(uuid.uuid4()).replace('-', '') #Nonce
    digest = hashlib.sha256((nonce + now + appSecret).encode()).hexdigest()

    digestBase64 = base64.b64encode(digest.encode()).decode() #PasswordDigest
    return 'UsernameToken Username="{}",PasswordDigest="{}",Nonce="{}",Created="{}"'.format(appKey, digestBase64, nonce, now)

def main():
    #Request headers
    header = {'Authorization': 'WSSE realm="SDP",profile="UsernameToken",type="Appkey"',
              'X-WSSE': buildWSSEHeader(APP_KEY, APP_SECRET)}
    # Request body
    formData = {'from': sender,
                'to': receiver,
                'templateId': TEMPLATE_ID,
                'templateParas': TEMPLATE_PARAM,
                'statusCallback': statusCallBack
                }
    print(header)

    # Ignore the certificate trust issues to prevent API calling failures caused by HTTPS certificate authentication failures.
    r = requests.post(url, data=formData, headers=header, verify=False)
    print(r.text) #Record the response.

if __name__ == '__main__':
    main()
```

Sending SMSs (Example 2)

```
# -*- coding: utf-8 -*-
import time
import uuid
import hashlib
import base64
import ssl
import urllib.parse
import urllib.request

# Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
url = 'https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendSms/v1' # Application access address (obtain it from the Application Management page on the console) and API access URI.
```



```
APP_KEY = "c8RWg3ggEcyd4D3p94bf3Y7x1Ile" #APP_Key
// Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
APP_SECRET = "q4Ii87Bh**********80SfD7Al" #APP_Secret
sender = "csms12345678" # SMS channel number.
TEMPLATE_ID = "8ff55eac1d0b478ab3c06c3c6a492300" # Template ID

# Mandatory. Global number format (including the country code), for example, +9100****11. Use commas (,) to separate multiple numbers.
receiver = "+9100****11,9100****12" # Recipient numbers

# Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
statusCallBack = ""

'''
Optional. If a template with no variable is used, assign an empty value to this parameter, for example, TEMPLATE_PARAM = '';
Example of a single-variable template: If the template content is "Your verification code is ${1}", TEMPLATE_PARAM can be set to '["369751"]'.
Example of a dual-variable template: If the template content is "You have ${1} parcel delivered to ${2}", TEMPLATE_PARAM can be set to '["3","main gate of People's Park"]'.
Each variable in the template must be assigned a value, and the value cannot be empty.
To view more information, choose Service Overview > Template and Variable Specifications.
'''
TEMPLATE_PARAM = '["369751"]' # Template variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

'''
Construct the value of X-WSSE.
@param appKey: string
@param appSecret: string
@return: string
'''
def buildWSSEHeader(appKey, appSecret):
    now = time.strftime('%Y-%m-%dT%H:%M:%SZ') #Created
    nonce = str(uuid.uuid4()).replace('-', '') #Nonce
    digest = hashlib.sha256((nonce + now + appSecret).encode()).hexdigest()

    digestBase64 = base64.b64encode(digest.encode()).decode() #PasswordDigest
    return 'UsernameToken Username="{}",PasswordDigest="{}",Nonce="{}",Created="{}"'.format(appKey, digestBase64, nonce, now)

def main():
    # Request body
    formData = urllib.parse.urlencode({
                'from': sender,
                'to': receiver,
                'templateId': TEMPLATE_ID,
                'templateParas': TEMPLATE_PARAM,
                'statusCallback': statusCallBack
                }).encode('ascii')

    req = urllib.request.Request(url=url, data=formData, method='POST') # The request method is POST.
    # Parameters in the request headers
    req.add_header('Authorization', 'WSSE realm="SDP",profile="UsernameToken",type="Appkey"')
    req.add_header('X-WSSE', buildWSSEHeader(APP_KEY, APP_SECRET))
    req.add_header('Content-Type', 'application/x-www-form-urlencoded')

    # Ignore the certificate trust issues to prevent API calling failures caused by HTTPS certificate authentication failures.
    ssl._create_default_https_context = ssl._create_unverified_context

    try:
        r = urllib.request.urlopen(req) # Send a request.
        print(r.read().decode('utf-8')) # Record the response.
    except urllib.error.HTTPError as e:
        print(e.code)
```



```
        print(e.read().decode('utf-8'))
    except urllib.error.URLError as e:
        print(e.reason)

if __name__ == '__main__':
    main()
```

Sending SMSs in Batches (Example 1)

```
# -*- coding: utf-8 -*-
import time
import uuid
import hashlib
import base64
import requests #Run pip install requests to install the dependency.

# Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
url = 'https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendDiffSms/v1' # Application access address (obtain it from the Application Management page on the console) and API access URI.
// Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
APP_KEY = "c8RWg3ggEcyd4D3p94bf3Y7x1Ile" #APP_Key
APP_SECRET = "q4Ii87Bh**********80SfD7Al" #APP_Secret
sender = "csms12345678" #SMS channel number.
TEMPLATE_ID_1 = "8ff55eac1d0b478ab3c06c3c6a492300" # Template ID 1
TEMPLATE_ID_2 = "8ff55eac1d0b478ab3c06c3c6a492300" # Template ID 2

# Mandatory. Global number format (including the country code), for example, +9100****11. Use commas (,) to separate multiple numbers.
receiver_1 = ["+9100****11", "+9100****12"] # Recipient number of template 1
receiver_2 = ["+9100****13", "+9100****14"] # Recipient number of template 2

# Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
statusCallBack = ""

'''
Optional. If a template with no variable is used, assign an empty value to this parameter, for example, TEMPLATE_PARAM = [];
Example of a single-variable template: If the template content is "Your verification code is ${1}", TEMPLATE_PARAM can be set to ["369751"].
Example of a dual-variable template: If the template content is "You have ${1} parcel delivered to ${2}", TEMPLATE_PARAM can be set to ["3","main gate of People's Park"].
Each variable in the template must be assigned a value, and the value cannot be empty.
To view more information, choose Service Overview > Template and Variable Specifications.
'''
TEMPLATE_PARAM_1 = ["123456"] # Template 1 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).
TEMPLATE_PARAM_2 = ["234567"] # Template 2 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

'''
Construct the value of X-WSSE.
@param appKey: string
@param appSecret: string
@return: string
'''
def buildWSSEHeader(appKey, appSecret):
    now = time.strftime('%Y-%m-%dT%H:%M:%SZ') #Created
    nonce = str(uuid.uuid4()).replace('-', '') #Nonce
    digest = hashlib.sha256((nonce + now + appSecret).encode()).hexdigest()

    digestBase64 = base64.b64encode(digest.encode()).decode() #PasswordDigest
    return 'UsernameToken Username="{}",PasswordDigest="{}",Nonce="{}",Created="{}"'.format(appKey, digestBase64, nonce, now)
```



```
def main():
    # Request headers
    header = {'Authorization':'WSSE realm="SDP",profile="UsernameToken",type="Appkey"',
              'X-WSSE': buildWSSEHeader(APP_KEY, APP_SECRET)}
    # Request body
    jsonData = {'from': sender,
                'statusCallback': statusCallBack,
                'smsContent':[
                    {'to':receiver_1,
                     'templateId':TEMPLATE_ID_1,
                     'templateParas':TEMPLATE_PARAM_1
                     },
                    {'to':receiver_2,
                     'templateId':TEMPLATE_ID_2,
                     'templateParas':TEMPLATE_PARAM_2
                     }]
                }
    print(header)

    # Ignore the certificate trust issues to prevent API calling failures caused by HTTPS certificate
authentication failures.
    r = requests.post(url, json=jsonData, headers=header, verify=False)
    print(r.text) # Record the response.

if __name__ == '__main__':
    main()
```

Sending SMSs in Batches (Example 2)

```
# -*- coding: utf-8 -*-
import time
import uuid
import hashlib
import base64
import json
import ssl
import urllib.request

# Mandatory. The values here are example values only. Obtain the actual values based on Development
Preparation.
url = 'https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendDiffSms/v1' # Application
access address (obtain it from the Application Management page on the console) and API access URI.
// Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store
them in the configuration file or environment variables.
APP_KEY = "c8RWg3ggEcyd4D3p94bf3Y7x1Ile" #APP_Key
APP_SECRET = "q4Ii87Bh**********80SfD7Al" #APP_Secret
sender = "csms12345678" #SMS channel number.
TEMPLATE_ID_1 = "8ff55eac1d0b478ab3c06c3c6a492300" # Template ID 1
TEMPLATE_ID_2 = "8ff55eac1d0b478ab3c06c3c6a492300" # Template ID 2

# Mandatory. Global number format (including the country code), for example, +9100****11. Use commas
(,) to separate multiple numbers.
receiver_1 = ["+9100****11", "+9100****12"] # Recipient number of template 1
receiver_2 = ["+9100****13", "+9100****14"] # Recipient number of template 2

# Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter
is set to an empty value or left unspecified, customers do not receive status reports.
statusCallBack = ""

'''
Optional. If a template with no variable is used, assign an empty value to this parameter, for example,
TEMPLATE_PARAM = [];
Example of a single-variable template: If the template content is "Your verification code is ${1}",
TEMPLATE_PARAM can be set to ["369751"].
Example of a dual-variable template: If the template content is "You have ${1} parcel delivered to ${2}",
TEMPLATE_PARAM can be set to ["3","main gate of People's Park"].
Each variable in the template must be assigned a value, and the value cannot be empty.
To view more information, choose Service Overview > Template and Variable Specifications.
'''
```



```
TEMPLATE_PARAM_1 = ["123456"] #Template 1 variable. The following uses a single-variable verification
code SMS message as an example. The customer needs to generate a 6-digit verification code and define it
as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).
TEMPLATE_PARAM_2 = ["234567"] #Template 2 variable. The following uses a single-variable verification
code SMS message as an example. The customer needs to generate a 6-digit verification code and define it
as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

'''
Construct the value of X-WSSE.
@param appKey: string
@param appSecret: string
@return: string
'''
def buildWSSEHeader(appKey, appSecret):
    now = time.strftime('%Y-%m-%dT%H:%M:%SZ') #Created
    nonce = str(uuid.uuid4()).replace('-', '') #Nonce
    digest = hashlib.sha256((nonce + now + appSecret).encode()).hexdigest()

    digestBase64 = base64.b64encode(digest.encode()).decode() #PasswordDigest
    return 'UsernameToken Username="{}",PasswordDigest="{}",Nonce="{}",Created="{}"'.format(appKey,
digestBase64, nonce, now)

def main():
    # Request body
    jsonData = json.dumps({'from': sender,
                           'statusCallback': statusCallBack,
                           'smsContent':[
                               {'to':receiver_1,
                                'templateId':TEMPLATE_ID_1,
                                'templateParas':TEMPLATE_PARAM_1
                                },
                               {'to':receiver_2,
                                'templateId':TEMPLATE_ID_2,
                                'templateParas':TEMPLATE_PARAM_2
                                }]
                           }).encode('ascii')

    req = urllib.request.Request(url=url, data=jsonData, method='POST') #The request method is POST.
    # Parameters in the request headers
    req.add_header('Authorization', 'WSSE realm="SDP",profile="UsernameToken",type="Appkey"')
    req.add_header('X-WSSE', buildWSSEHeader(APP_KEY, APP_SECRET))
    req.add_header('Content-Type', 'application/json')

    # Ignore the certificate trust issues to prevent API calling failures caused by HTTPS certificate
authentication failures.
    ssl._create_default_https_context = ssl._create_unverified_context

    try:
        r = urllib.request.urlopen(req) # Send a request.
        print(r.read().decode('utf-8')) # Record the response.
    except urllib.error.HTTPError as e:
        print(e.code)
        print(e.read().decode('utf-8'))
    except urllib.error.URLError as e:
        print(e.reason)

if __name__ == '__main__':
    main()
```

Receiving Status Reports

```
import urllib.parse

# Data example (urlencode) of the status report reported by the SMS platform
#success_body =
"sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId=2ea20735-
f856-4376-afbf-570bd70a46ee_11840135&status=DELIVRD";
failed_body =
"orgCode=E200027&sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId
```



```
=2ea20735-f856-4376-afbf-570bd70a46ee_11840135&status=RTE_ERR";

'''
Parse the status report data.
@param data: Status report data reported by the SMS platform.
@return:
'''
def onSmsStatusReport(data):
    keyValues = urllib.parse.parse_qs(data); # Parse the status report data.

    '''
    Example: Parsing status is used as an example.

    'smsMsgId': Unique ID of an SMS
    'total': Number of SMS segments
    'sequence': Sequence number of an SMS after splitting
    'source': Status report source
    'updateTime': Resource update time
    'status': Enumeration values of the status report
     'orgCode': Status code
    '''
    status = keyValues.get('status'); #Enumerated values of the status report
    # Check whether the SMS is successfully sent based on the value of status.
    if 'DELIVRD' == str.upper(status[0]):
        print('Send sms success. smsMsgId: ', keyValues.get('smsMsgId')[0]);
    else:
        # The SMS fails to be sent. The values of status and orgCode are recorded.
        print('Send sms failed. smsMsgId: ', keyValues.get('smsMsgId')[0]);
        print('Failed status: ', status[0]);
        print('Failed orgCode: ', keyValues.get('orgCode')[0]);

if __name__ == '__main__':
#     onSmsStatusReport(success_body)
    onSmsStatusReport(failed_body)
```

References

- Code examples: **Java**, **PHP**, **C#**, **Node.js**, and **Go**
- APIs: SMS Sending API, Batch SMS Sending API, and Status Report Receiving API

3.2.4 C#

Example	**Sending SMSs (Example 1)**, **Sending SMSs (Example 2)** **Sending SMSs in Batches (Example 1)**, **Sending SMSs in Batches (Example 2)** **Receiving Status Reports**
Environment	Example 1: .NET Core 2.0 or later, or .NET Framework 4.6 or later Example 2: .NET Core 1.0 or later, or .NET Framework 2.0 or later
Library	For Newtonsoft.Json 11.0.2 and later versions, see **https://www.newtonsoft.com/json**.

NOTICE

- Sending SMSs shows an example of sending group SMSs with a single template. Sending SMSs in batches shows an example of sending group SMSs with multiple templates.
- The examples described in this document may involve the use of personal data. You are advised to comply with relevant laws and regulations and take measures to ensure that personal data is fully protected.
- The examples described in this document are for demonstration purposes only. Commercial use of the examples is prohibited.
- Information in this document is for your reference only. It does not constitute any offer or commitment.

Sending SMSs (Example 1)

```
using System;
using System.Collections.Generic;
using System.Net;
using System.Net.Http;
using System.Security.Cryptography;
using System.Text;

namespace msgsms_csharp_demo
{
    class SendSms
    {
        static void Main(string[] args)
        {
            // Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
            string apiAddress = "https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendSms/v1"; // Application access address (obtain it from the Application Management page on the console) and API access URI.
            // Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
            string appKey = "c8RWg3ggEcyd4D3p94bf3Y7x1Ile"; //APP_Key
            string appSecret = "q4Ii87Bh**********80SfD7Al"; //APP_Secret
            string sender = "csms12345678"; // SMS channel number.
            string templateId = "8ff55eac1d0b478ab3c06c3c6a492300"; // Template ID

            // Mandatory. Global number format (including the country code), for example, +9100****11. Use commas (,) to separate multiple numbers.
            string receiver = "++9100****11,++9100****12"; // Recipient numbers

            // Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
            string statusCallBack = "";

            /*
             * Optional. If a non-variable template is used, assign an empty value to this parameter, for example, string templateParas = "";
              * Example of a single-variable template: If the template content is "Your verification code is ${1}", templateParas can be set to "[\"369751\"]".
               * Example of a dual-variable template: If the template content is "You have ${1} delivered to ${2}", templateParas can be set to "[\"3\",\"main gate of People's Park\"]".
              * Each variable in the template must be assigned a value, and the value cannot be empty.
              * To view more information, choose Service Overview > Template and Variable Specifications.
              */
            string templateParas = "[\"369751\"]"; //Template variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).
```



```
        try
        {
          // Ignore the certificate trust issues to prevent API calling failures caused by HTTPS certificate
authentication failures.
            HttpClient client = new HttpClient();
            ServicePointManager.ServerCertificateValidationCallback = delegate { return true; };

            // Request headers
            client.DefaultRequestHeaders.Add("Authorization", "WSSE realm=\"SDP\",profile=
\"UsernameToken\",type=\"Appkey\"");
            client.DefaultRequestHeaders.Add("X-WSSE", BuildWSSEHeader(appKey, appSecret));
            // Request body
            var body = new Dictionary<string, string>() {
                {"from", sender},
                {"to", receiver},
                {"templateId", templateId},
                {"templateParas", templateParas},
                {"statusCallback", statusCallBack}
            };

            HttpContent content = new FormUrlEncodedContent(body);

            var response = client.PostAsync(apiAddress, content).Result;
            Console.WriteLine(response.StatusCode); // Record the result code of the response.
            var res = response.Content.ReadAsStringAsync().Result;
            Console.WriteLine(res); // Record the response.
        }
        catch (Exception e)
        {
            Console.WriteLine(e.StackTrace);
            Console.WriteLine(e.Message);
        }
    }

    /// <summary>
    /// Construct the value of X-WSSE.
    /// </summary>
    /// <param name="appKey"></param>
    /// <param name="appSecret"></param>
    /// <returns></returns>
    static string BuildWSSEHeader(string appKey, string appSecret)
    {
        string now = DateTime.Now.ToString("yyyy-MM-ddTHH:mm:ssZ"); //Created
        string nonce = Guid.NewGuid().ToString().Replace("-", ""); //Nonce

        byte[] material = Encoding.UTF8.GetBytes(nonce + now + appSecret);
        byte[] hashed = SHA256Managed.Create().ComputeHash(material);
        string hexdigest = BitConverter.ToString(hashed).Replace("-", "");
        string base64 = Convert.ToBase64String(Encoding.UTF8.GetBytes(hexdigest)); //PasswordDigest

        return String.Format("UsernameToken Username=\"{0}\",PasswordDigest=\"{1}\",Nonce=
\"{2}\",Created=\"{3}\"",
                        appKey, base64, nonce, now);
    }
  }
}
```

Sending SMSs (Example 2)

```
using System;
using System.Collections.Specialized;
using System.IO;
using System.Net;
using System.Security.Cryptography;
using System.Text;
using System.Web;

namespace msgsms_csharp_demo
{
```



```
class SendSms
{
    static void Main(string[] args)
    {
        // Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
        string apiAddress = "https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendSms/v1"; // Application access address (obtain it from the Application Management page on the console) and API access URI.
        // Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
        string appKey = "c8RWg3ggEcyd4D3p94bf3Y7x1Ile"; //APP_Key
        string appSecret = "q4Ii87Bh**********80SfD7Al"; //APP_Secret
        string sender = "csms12345678"; // SMS channel number.
        string templateId = "8ff55eac1d0b478ab3c06c3c6a492300"; // Template ID

        // Mandatory. Global number format (including the country code), for example, +9100****11. Use commas (,) to separate multiple numbers.
        string receiver = "+9100****11,+9100****12"; // Recipient numbers

        // Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
        string statusCallBack = "";

        /*
         * Optional. If a non-variable template is used, assign an empty value to this parameter, for example, string templateParas = "";
          * Example of a single-variable template: If the template content is "Your verification code is ${1}", templateParas can be set to "[\"369751\"]".
          * Example of a dual-variable template: If the template content is "You have ${1} delivered to ${2}", templateParas can be set to "[\"3\",\"main gate of People's Park\"]".
         * Each variable in the template must be assigned a value, and the value cannot be empty.
         * To view more information, choose Service Overview > Template and Variable Specifications.
         */
        string templateParas = "[\"369751\"]"; // Template variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

        try
        {
          // Ignore the certificate trust issues to prevent API calling failures caused by HTTPS certificate authentication failures.
            ServicePointManager.ServerCertificateValidationCallback = delegate { return true; };
            // Set the security protocol type to 3072 to use TLS 1.2.
            ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls | (SecurityProtocolType)3072;

            HttpWebRequest myReq = (HttpWebRequest)WebRequest.Create(apiAddress);
            // Request method
            myReq.Method = "POST";
            // Request headers
            myReq.ContentType = "application/x-www-form-urlencoded";
            myReq.Headers.Add("Authorization", "WSSE realm=\"SDP\",profile=\"UsernameToken\",type=\"Appkey\"");
            myReq.Headers.Add("X-WSSE", BuildWSSEHeader(appKey, appSecret));
            // Request body
            NameValueCollection keyValues = new NameValueCollection
            {
                {"from", sender},
                {"to", receiver},
                {"templateId", templateId},
                {"templateParas", templateParas},
                {"statusCallback", statusCallBack}
            };
            string body = BuildQueryString(keyValues);

            // Send request data.
            StreamWriter req = new StreamWriter(myReq.GetRequestStream());
            req.Write(body);
```



```
            req.Close();

            // Obtain response data.
            HttpWebResponse myResp = (HttpWebResponse)myReq.GetResponse();
            StreamReader resp = new StreamReader(myResp.GetResponseStream());
            string result = resp.ReadToEnd();
            myResp.Close();
            resp.Close();

            Console.WriteLine(result);
        }
        catch (Exception e)
        {
            Console.WriteLine(e.StackTrace);
            Console.WriteLine(e.Message);
        }
    }

    /// <summary>
    /// Construct the value of X-WSSE.
    /// </summary>
    /// <param name="appKey"></param>
    /// <param name="appSecret"></param>
    /// <returns></returns>
    static string BuildWSSEHeader(string appKey, string appSecret)
    {
        string now = DateTime.Now.ToString("yyyy-MM-ddTHH:mm:ssZ"); //Created
        string nonce = Guid.NewGuid().ToString().Replace("-", ""); //Nonce

        byte[] material = Encoding.UTF8.GetBytes(nonce + now + appSecret);
        byte[] hashed = SHA256Managed.Create().ComputeHash(material);
        string hexdigest = BitConverter.ToString(hashed).Replace("-", "");
        string base64 = Convert.ToBase64String(Encoding.UTF8.GetBytes(hexdigest)); //PasswordDigest

        return String.Format("UsernameToken Username=\"{0}\",PasswordDigest=\"{1}\",Nonce=
\"{2}\",Created=\"{3}\"",
                        appKey, base64, nonce, now);
    }

    /// <summary>
    /// Construct the request body.
    /// </summary>
    /// <param name="keyValues"></param>
    /// <returns></returns>
    static string BuildQueryString(NameValueCollection keyValues)
    {
        StringBuilder temp = new StringBuilder();
        foreach (string item in keyValues.Keys)
        {

temp.Append(item).Append("=").Append(HttpUtility.UrlEncode(keyValues.Get(item))).Append("&");
        }
        return temp.Remove(temp.Length - 1, 1).ToString();
    }
  }
}
```

Sending SMSs in Batches (Example 1)

```
using Newtonsoft.Json;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Net;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Security.Cryptography;
using System.Text;
```



```
namespace msgsms_csharp_demo
{
    class SendDiffSms
    {
        static void Main(string[] args)
        {
            // Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
            string apiAddress = "https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendDiffSms/v1"; // Application access address (obtain it from the Application Management page on the console) and API access URI.
            // Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
            string appKey = "c8RWg3ggEcyd4D3p94bf3Y7x1Ile"; //APP_Key
            string appSecret = "q4Ii87Bh**********80SfD7Al"; //APP_Secret
            string sender = "csms12345678"; // SMS channel number.
            string templateId_1 = "979b639cbd0b4b6b88e0fd5de4ad6f85"; // Template ID 1
            string templateId_2 = "979b639cbd0b4b6b88e0fd5de4ad6f85"; // Template ID 2

            // Mandatory. Global number format (including the country code), for example, +9100****11. Use commas (,) to separate multiple numbers.
            string[] receiver_1 = { "+9100****11", "+9100****12" }; // Recipient number of template 1
            string[] receiver_2 = { "+9100****13", "+9100****14" }; // Recipient number of template 2

            // Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
            string statusCallBack = "";

            /*
             * Optional. If a non-variable template is used, assign an empty value to this parameter, for example, string[] templateParas = {};
            * Example of a single-variable template: If the template content is "Your verification code is ${1}", templateParas can be set to ["369751"].
             * Example of a dual-variable template: If the template content is "You have ${1} delivered to ${2}", templateParas can be set to {"3","main gate of People's Park"}.
             * Each variable in the template must be assigned a value, and the value cannot be empty.
             * To view more information, choose Service Overview > Template and Variable Specifications.
             */
            string[] templateParas_1 = {"123456"}; // Template 1 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).
            string[] templateParas_2 = {"234567"}; // Template 2 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

            ArrayList smsContent = new ArrayList
            {
                InitDiffSms(receiver_1, templateId_1, templateParas_1),
                InitDiffSms(receiver_2, templateId_2, templateParas_2)
            };

            try
            {
               // Ignore the certificate trust issues to prevent API calling failures caused by HTTPS certificate authentication failures.
                HttpClient client = new HttpClient();
                ServicePointManager.ServerCertificateValidationCallback = delegate { return true; };

                // Request headers
                client.DefaultRequestHeaders.Add("Authorization", "WSSE realm=\"SDP\",profile=\"UsernameToken\",type=\"Appkey\"");
                client.DefaultRequestHeaders.Add("X-WSSE", BuildWSSEHeader(appKey, appSecret));
                // Request body
                var body = new Dictionary<string, object>{
                    {"from", sender},
                    {"statusCallback", statusCallBack},
                    {"smsContent", smsContent}
```



```
            };

            HttpContent content = new StringContent(JsonConvert.SerializeObject(body));
            // Content-Type in the request headers
            content.Headers.ContentType = new MediaTypeHeaderValue("application/json");

            var response = client.PostAsync(apiAddress, content).Result;
            Console.WriteLine(response.StatusCode); // Record the result code of the response.
            var res = response.Content.ReadAsStringAsync().Result;
            Console.WriteLine(res); // Record the response.
        }
        catch (Exception e)
        {
            Console.WriteLine(e.StackTrace);
            Console.WriteLine(e.Message);
        }
    }

    /// <summary>
    /// Construct the value of smsContent.
    /// </summary>
    /// <param name="receiver"></param>
    /// <param name="templateId"></param>
    /// <param name="templateParas"></param>
    /// <returns></returns>
    static Dictionary<string, object> InitDiffSms(string[] receiver, string templateId, string[] templateParas)
    {
        Dictionary<string, object> dic = new Dictionary<string, object>
        {
            {"to", receiver},
            {"templateId", templateId},
            {"templateParas", templateParas}
        };

        return dic;
    }

    /// <summary>
    /// Construct the value of X-WSSE.
    /// </summary>
    /// <param name="appKey"></param>
    /// <param name="appSecret"></param>
    /// <returns></returns>
    static string BuildWSSEHeader(string appKey, string appSecret)
    {
        string now = DateTime.Now.ToString("yyyy-MM-ddTHH:mm:ssZ"); //Created
        string nonce = Guid.NewGuid().ToString().Replace("-", ""); //Nonce

        byte[] material = Encoding.UTF8.GetBytes(nonce + now + appSecret);
        byte[] hashed = SHA256Managed.Create().ComputeHash(material);
        string hexdigest = BitConverter.ToString(hashed).Replace("-", "");
        string base64 = Convert.ToBase64String(Encoding.UTF8.GetBytes(hexdigest)); //PasswordDigest

        return String.Format("UsernameToken Username=\"{0}\",PasswordDigest=\"{1}\",Nonce=
\"{2}\",Created=\"{3}\"",
                    appKey, base64, nonce, now);
    }
  }
}
```

Sending SMSs in Batches (Example 2)

```
using Newtonsoft.Json;
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Net;
using System.Security.Cryptography;
```



```
using System.Text;

namespace msgsms_csharp_demo
{
    class SendDiffSms
    {
        static void Main(string[] args)
        {
            // Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
            string apiAddress = "https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendDiffSms/v1"; // Application access address (obtain it from the Application Management page on the console) and API access URI.
            // Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
            string appKey = "c8RWg3ggEcyd4D3p94bf3Y7x1Ile"; //APP_Key
            string appSecret = "q4Ii87Bh**********80SfD7Al"; //APP_Secret
            string sender = "csms12345678"; // SMS channel number.
            string templateId_1 = "979b639cbd0b4b6b88e0fd5de4ad6f85"; // Template ID 1
            string templateId_2 = "979b639cbd0b4b6b88e0fd5de4ad6f85"; // Template ID 2

            // Mandatory. Global number format (including the country code), for example, +9100****11. Use commas (,) to separate multiple numbers.
            string[] receiver_1 = { "+9100****11", "+9100****12" }; // Recipient number of template 1
            string[] receiver_2 = { "+9100****13", "+9100****13" }; // Recipient number of template 2

            // Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
            string statusCallBack = "";

            /*
             * Optional. If a non-variable template is used, assign an empty value to this parameter, for example, string[] templateParas = {};
            * Example of a single-variable template: If the template content is "Your verification code is ${1}", templateParas can be set to ["369751"].
             * Example of a dual-variable template: If the template content is "You have ${1} parcel delivered to ${2}", templateParas can be set to {"3","main gate of People's Park"}.
             * Each variable in the template must be assigned a value, and the value cannot be empty.
             * To view more information, choose Service Overview > Template and Variable Specifications.
             */
            string[] templateParas_1 = {"123456"}; // Template 1 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).
            string[] templateParas_2 = {"234567"}; // Template 2 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

            ArrayList smsContent = new ArrayList
            {
                InitDiffSms(receiver_1, templateId_1, templateParas_1),
                InitDiffSms(receiver_2, templateId_2, templateParas_2)
            };

            try
            {
               // Ignore the certificate trust issues to prevent API calling failures caused by HTTPS certificate authentication failures.
                ServicePointManager.ServerCertificateValidationCallback = delegate { return true; };
                // Set the security protocol type to 3072 to use TLS 1.2.
                ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls | (SecurityProtocolType)3072;

                HttpWebRequest myReq = (HttpWebRequest)WebRequest.Create(apiAddress);
                // Request method
                myReq.Method = "POST";
                // Request headers
                myReq.ContentType = "application/json";
                myReq.Headers.Add("Authorization", "WSSE realm=\"SDP\",profile=\"UsernameToken\",type=
```



```
\"Appkey\"");
            myReq.Headers.Add("X-WSSE", BuildWSSEHeader(appKey, appSecret));
            // Request body
            var body = new Dictionary<string, object>() {
                {"from", sender},
                {"statusCallback", statusCallBack},
                {"smsContent", smsContent}
            };

            string jsonData = JsonConvert.SerializeObject(body);

            // Send request data.
            StreamWriter req = new StreamWriter(myReq.GetRequestStream());
            req.Write(jsonData);
            req.Close();

            // Obtain response data.
            HttpWebResponse myResp = (HttpWebResponse)myReq.GetResponse();
            StreamReader resp = new StreamReader(myResp.GetResponseStream());
            string result = resp.ReadToEnd();
            myResp.Close();
            resp.Close();

            Console.WriteLine(result);
        }
        catch (Exception e)
        {
            Console.WriteLine(e.StackTrace);
            Console.WriteLine(e.Message);
        }
    }

    /// <summary>
    /// Construct the value of smsContent.
    /// </summary>
    /// <param name="receiver"></param>
    /// <param name="templateId"></param>
    /// <param name="templateParas"></param>
    /// <returns></returns>
    static Dictionary<string, object> InitDiffSms(string[] receiver, string templateId, string[] templateParas)
    {
        Dictionary<string, object> dic = new Dictionary<string, object>
        {
            {"to", receiver},
            {"templateId", templateId},
            {"templateParas", templateParas}
        };

        return dic;
    }

    /// <summary>
    /// Construct the value of X-WSSE.
    /// </summary>
    /// <param name="appKey"></param>
    /// <param name="appSecret"></param>
    /// <returns></returns>
    static string BuildWSSEHeader(string appKey, string appSecret)
    {
        string now = DateTime.Now.ToString("yyyy-MM-ddTHH:mm:ssZ"); //Created
        string nonce = Guid.NewGuid().ToString().Replace("-", ""); //Nonce

        byte[] material = Encoding.UTF8.GetBytes(nonce + now + appSecret);
        byte[] hashed = SHA256Managed.Create().ComputeHash(material);
        string hexdigest = BitConverter.ToString(hashed).Replace("-", "");
        string base64 = Convert.ToBase64String(Encoding.UTF8.GetBytes(hexdigest)); //PasswordDigest

        return String.Format("UsernameToken Username=\"{0}\",PasswordDigest=\"{1}\",Nonce=
\"{2}\",Created=\"{3}\"",
```



```
                        appKey, base64, nonce, now);
        }
    }
}
```

Receiving Status Reports

```
using System;
using System.Web;

namespace msgsms_csharp_demo
{
    class Report
    {
        static void Main(string[] args)
        {
            //string success_body =
"sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId=2ea20735-
f856-4376-afbf-570bd70a46ee_11840135&status=DELIVRD";
            string failed_body =
"orgCode=E200027&sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId
=2ea20735-f856-4376-afbf-570bd70a46ee_11840135&status=RTE_ERR";

            //onSmsStatusReport(success_body);
            onSmsStatusReport(failed_body);
        }

        /// <summary>
        /// Parse the status report data.
        /// </summary>
        ///<param name="data">Status report data reported by the SMS platform</param>
        static void onSmsStatusReport(string data)
        {
            var keyValues = HttpUtility.ParseQueryString(data); // Parse the status report data.

            /**
             * Example: Parsing status is used as an example.
             *
             * 'smsMsgId': Unique ID of an SMS
             * 'total': Number of SMS segments
             * 'sequence': Sequence number of an SMS after splitting
             * 'source': Status report source
             * 'updateTime': Resource update time
             * 'status': Enumeration values of the status report
             * 'orgCode': Status code
             */
            string status = keyValues.Get("status"); // Enumerated values of the status report
            // Check whether the SMS is successfully sent based on the value of status.
            if ("DELIVRD".Equals(status.ToUpper()))
            {
                Console.WriteLine("Send sms success. smsMsgId: " + keyValues.Get("smsMsgId"));
            }
            else
            {
                // The SMS fails to be sent. The values of status and orgCode are recorded.
                Console.WriteLine("Send sms failed. smsMsgId: " + keyValues.Get("smsMsgId"));
                Console.WriteLine("Failed status: " + status);
                Console.WriteLine("Failed orgCode: " + keyValues.Get("orgCode"));
            }
        }
    }
}
```

References

- Code examples: **Java**, **PHP**, **Python**, **Node.js**, and **Go**
- APIs: SMS Sending API, Batch SMS Sending API, and Status Report Receiving API

3.2.5 Node.js

Example	**Sending SMSs**, **Sending SMSs in Batches**, **Receiving Status Reports**
Environment	Node.js 8.12.0 or later versions are required. The examples in this section are developed based on Node.js 8.12.0.

NOTICE

- Sending SMSs shows an example of sending group SMSs with a single template. Sending SMSs in batches shows an example of sending group SMSs with multiple templates.
- The examples described in this document may involve the use of personal data. You are advised to comply with relevant laws and regulations and take measures to ensure that personal data is fully protected.
- The examples described in this document are for demonstration purposes only. Commercial use of the examples is prohibited.
- Information in this document is for your reference only. It does not constitute any offer or commitment.

Sending SMSs

```
/*jshint esversion: 6 */
var https = require('https'); // Introduce the HTTPS module.
var url = require('url'); // Introduce the URL module.
var querystring = require('querystring'); // Introduce the querystring module.

// Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
var realUrl = 'https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendSms/v1'; // Application access address (obtain it from the Application Management page on the console) and API access URI.
var appKey = 'c8RWg3ggEcyd4D3p94bf3Y7x1Ile'; //APP_Key
// Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
var appSecret = 'q4Ii87Bh**********80SfD7Al'; //APP_Secret
var sender = 'csms12345678'; // SMS channel number.
var templateId = '8ff55eac1d0b478ab3c06c3c6a492300'; // Template ID

// Mandatory. Global number format (including the country code), for example, +9100****11. Use commas (,) to separate multiple numbers.
var receiver = '+9100****11,+9100****12'; // Recipient numbers

// Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
var statusCallBack = '';

/**
 * Optional. If a non-variable template is used, assign an empty value to this parameter, for example, var templateParas = '';
 * Example of a single-variable template: If the template content is "Your verification code is ${1}", templateParas can be set to '["369751"]'.
 * Example of a dual-variable template: If the template content is "You have ${1} parcel delivered to ${2}", templateParas can be set to '["3","main gate of People's Park"]'.
 * Each variable in the template must be assigned a value, and the value cannot be empty.
 * To view more information, choose Service Overview > Template and Variable Specifications.
 */
```



```
var templateParas = '["369751"]'; // Template variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

/**
 * Construct the request body.
 *
 * @param sender
 * @param receiver
 * @param templateId
 * @param templateParas
 * @param statusCallBack
 * @returns
 */
function buildRequestBody(sender, receiver, templateId, templateParas, statusCallBack){
    return querystring.stringify({
        'from': sender,
        'to': receiver,
        'templateId': templateId,
        'templateParas': templateParas,
        'statusCallback': statusCallBack
    });
}

/**
 * Construct the value of X-WSSE.
 *
 * @param appKey
 * @param appSecret
 * @returns
 */
function buildWsseHeader(appKey, appSecret){
    var crypto = require('crypto');
    var util = require('util');

    var time = new Date(Date.now()).toISOString().replace(/.[0-9]+\Z/, 'Z'); //Created
    var nonce = crypto.randomBytes(64).toString('hex'); //Nonce
    var passwordDigestBase64Str = crypto.createHash('sha256').update(nonce + time +
appSecret).digest('base64'); //PasswordDigest

    return util.format('UsernameToken Username="%s",PasswordDigest="%s",Nonce="%s",Created="%s"',
appKey, passwordDigestBase64Str, nonce, time);
}

var urlobj = url.parse(realUrl); // Parse the realUrl character string and return a URL object.

var options = {
    host: urlobj.hostname, // Host name
    port: urlobj.port, // Port
    path: urlobj.pathname, //URI
    method: 'POST', // The request method is POST.
    headers: { // Request headers
        'Content-Type': 'application/x-www-form-urlencoded',
        'Authorization': 'WSSE realm="SDP",profile="UsernameToken",type="Appkey"',
        'X-WSSE': buildWsseHeader(appKey, appSecret)
    },
    rejectUnauthorized: false // Ignore the certificate trust issues to prevent API calling failures caused by HTTPS certificate authentication failures.
};
// Request body
var body = buildRequestBody(sender, receiver, templateId, templateParas, statusCallBack);

var req = https.request(options, (res) => {
    console.log('statusCode:', res.statusCode); // The response code is recorded.

    res.setEncoding('utf8'); // Set the response data encoding format.
    res.on('data', (d) => {
        console.log('resp:', d); // The response data is recorded.
    });
```



```
});
req.on('error', (e) => {
    console.error(e.message); // When a request error occurs, error details are recorded.
});
req.write(body); // Send data in the request body.
req.end(); // End the request.
```

Sending SMSs in Batches

```
/*jshint esversion: 6 */
var https = require('https'); // Introduce the HTTPS module.
var url = require('url'); // Introduce the URL module.

// Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
var realUrl = 'https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendDiffSms/v1'; // Application access address (obtain it from the Application Management page on the console) and API access URI.
var appKey = 'c8RWg3ggEcyd4D3p94bf3Y7x1Ile'; //APP_Key
// Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
var appSecret = 'q4Ii87Bh**********80SfD7Al'; //APP_Secret
var sender = 'csms12345678'; // SMS channel number.
var templateId1 = '8ff55eac1d0b478ab3c06c3c6a492300'; // Template ID 1
var templateId2 = '8ff55eac1d0b478ab3c06c3c6a492300'; // Template ID 2

// Mandatory. Global number format (including the country code), for example, +9100****11. Use commas (,) to separate multiple numbers.
var receiver1 = ['+9100****11','+9100****12']; // Recipient number of template 1
var receiver2 = ['+9100****13','+9100****14']; // Recipient number of template 2

// Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
var statusCallBack = '';

/**
 * Optional. If a non-variable template is used, assign an empty value to this parameter, for example, var templateParas = [];
 * Example of a single-variable template: If the template content is "Your verification code is ${1}", templateParas can be set to ['369751'].
 * Example of a dual-variable template: If the template content is "You have ${1} parcel delivered to ${2}", templateParas can be set to ['3','main gate of People's Park'].
 * Each variable in the template must be assigned a value, and the value cannot be empty.
 * To view more information, choose Service Overview > Template and Variable Specifications.
 */
var templateParas1 = ['123456']; // Template 1 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).
var templateParas2 = ['234567']; // Template 2 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

/**
 * Construct the value of smsContent.
 *
 * @param receiver
 * @param templateId
 * @param templateParas
 * @returns
 */
function initDiffSms(receiver, templateId, templateParas){
    return {'to': receiver, 'templateId': templateId, 'templateParas': templateParas};
}

/**
 * Construct the value of X-WSSE.
 *
 * @param appKey
 * @param appSecret
```



```
 * @returns
 */
function buildWsseHeader(appKey, appSecret){
    var crypto = require('crypto');
    var util = require('util');

    var time = new Date(Date.now()).toISOString().replace(/.[0-9]+\Z/, 'Z'); //Created
    var nonce = crypto.randomBytes(64).toString('hex'); //Nonce
    var passwordDigestBase64Str = crypto.createHash('sha256').update(nonce + time +
appSecret).digest('base64'); //PasswordDigest

    return util.format('UsernameToken Username="%s",PasswordDigest="%s",Nonce="%s",Created="%s"',
appKey, passwordDigestBase64Str, nonce, time);
}

var body = JSON.stringify ({ // Request body
    'from': sender,
    'statusCallback': statusCallBack,
    'smsContent': [
        initDiffSms(receiver1, templateId1, templateParas1),
        initDiffSms(receiver2, templateId2, templateParas2)
    ]}
);

var urlobj = url.parse(realUrl); // Parse the realUrl character string and return a URL object.

var options = {
    host: urlobj.hostname, // Host name
    port: urlobj.port, // Port
    path: urlobj.pathname, //URI
    method: 'POST', // The request method is POST.
    headers: { // Request headers
        'Content-Type': 'application/json',
        'Authorization': 'WSSE realm="SDP",profile="UsernameToken",type="Appkey"',
        'X-WSSE': buildWsseHeader(appKey, appSecret)
    },
    rejectUnauthorized: false // Ignore the certificate trust issues to prevent API calling failures caused by
HTTPS certificate authentication failures.
};

var req = https.request(options, (res) => {
    console.log('statusCode:', res.statusCode); // The response code is recorded.

    res.setEncoding('utf8'); // Set the response data encoding format.
    res.on('data', (d) => {
        console.log('resp:', d); // The response data is recorded.
    });
});
req.on('error', (e) => {
    console.error(e.message); // When a request error occurs, error details are recorded.
});
req.write(body); // Send data in the request body.
req.end(); // End the request.
```

Receiving Status Reports

```
/*jshint esversion: 6 */

// Data example (urlencode) of the status report reported by the SMS platform
//var success_body =
"sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId=2ea20735-
f856-4376-afbf-570bd70a46ee_11840135&status=DELIVRD";
var failed_body =
"orgCode=E200027&sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId
=2ea20735-f856-4376-afbf-570bd70a46ee_11840135&status=RTE_ERR";

/**
 * Parse the status report data.
 *
```



```
 * @param data: Status report data reported by the SMS platform.
 * @returns
 */
function onSmsStatusReport(data) {
    var querystring = require('querystring');
    var keyValues = querystring.parse(data);// Parse the status report data.

    /**
     * Example: Parsing status is used as an example.
     *
     * 'smsMsgId': Unique ID of an SMS
     * 'total': Number of SMS segments
     * 'sequence': Sequence number of an SMS after splitting
     * 'source': Status report source
     * 'updateTime': Resource update time
     * 'status': Enumeration values of the status report
     * 'orgCode': Status code
     */
    var status = keyValues.status; // Enumerated values of the status report
    // Check whether the SMS is successfully sent based on the value of status.
    if ('DELIVRD' === status.toUpperCase()) {
        console.log('Send sms success. smsMsgId: ', keyValues.smsMsgId);
    } else {
        // The SMS fails to be sent. The values of status and orgCode are recorded.
        console.log('Send sms failed. smsMsgId: ', keyValues.smsMsgId);
        console.log('Failed status: ', status);
        console.log('Failed orgCode: ', keyValues.orgCode);
    }
}

// onSmsStatusReport(success_body);
onSmsStatusReport(failed_body);
```

References

- Code examples: **Java**, **PHP**, **Python**, **C#**, and **Go**
- APIs: SMS Sending API, Batch SMS Sending API, and Status Report Receiving API

3.2.6 Go

Example	**Sending SMSs**, **Sending SMSs in Batches** **Receiving Status Reports**
Environment	Go 1.11 or later
Library	**github.com/satori/go.uuid**

NOTICE

- Sending SMSs shows an example of sending group SMSs with a single template. Sending SMSs in batches shows an example of sending group SMSs with multiple templates.
- The examples described in this document may involve the use of personal data. You are advised to comply with relevant laws and regulations and take measures to ensure that personal data is fully protected.
- The examples described in this document are for demonstration purposes only. Commercial use of the examples is prohibited.
- Information in this document is for your reference only. It does not constitute any offer or commitment.

Sending SMSs

```
package main

import (
    "bytes"
    "crypto/sha256"
    "crypto/tls"
    "encoding/base64"
    "fmt"
    "github.com/satori/go.uuid"
    "io/ioutil"
    "net/http"
    "net/url"
    "strings"
    "time"
)

// This parameter does not need to be modified. It is used to format the Authentication header field and assign a value to the X-WSSE parameter.
const WSSE_HEADER_FORMAT = "UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s\""
// This parameter does not need to be modified. It is used to format the Authentication header field and assign a value to the Authorization parameter.
const AUTH_HEADER_VALUE = "WSSE realm=\"SDP\",profile=\"UsernameToken\",type=\"Appkey\""

func main()  {
    // Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
    apiAddress := "https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendSms/v1" // Application access address (obtain it from the Application Management page on the console) and API access URI.
    // Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
    appKey := "c8RWg3ggEcyd4D3p94bf3Y7x1Ile" //APP_Key
    appSecret := "q4Ii87Bh**********80SfD7Al" //APP_Secret
    sender := "csms12345678" // SMS channel number.
    templateId := "8ff55eac1d0b478ab3c06c3c6a492300" // Template ID

    // Mandatory. Global number format (including the country code), for example, +9100****11. Use commas (,) to separate multiple numbers.
    receiver := "+9100****11,+9100****12" // Recipient numbers

    // Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
    statusCallBack := ""

    /*
     * Optional. If a non-variable template is used, assign an empty value to this parameter, for example, string templateParas = "";
```



```
    * Example of a single-variable template: If the template content is "Your verification code is ${1}", templateParas can be set to "[\"369751\"]".
    * Example of a dual-variable template: If the template content is "You have ${1} delivered to ${2}", templateParas can be set to "[\"3\",\"main gate of People's Park\"]".
    * Each variable in the template must be assigned a value, and the value cannot be empty.
    * To view more information, choose Service Overview > Template and Variable Specifications.
    */
    templateParas := "[\"369751\"]" // Template variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

    body := buildRequestBody(sender,receiver,templateId,templateParas,statusCallBack)
    headers := make(map[string]string)
    headers["Content-Type"] = "application/x-www-form-urlencoded"
    headers["Authorization"] = AUTH_HEADER_VALUE;
    headers["X-WSSE"] = buildWsseHeader(appKey, appSecret);
    resp, err := post(apiAddress, []byte(body),headers)
    if err != nil {
        return
    }
    fmt.Println(resp);
}

/**
 * sender, receiver, and templateId cannot be empty.
 */
func buildRequestBody(sender, receiver, templateId, templateParas, statusCallBack) string {
    param := "from=" + url.QueryEscape(sender) + "&to=" + url.QueryEscape(receiver) + "&templateId=" + url.QueryEscape(templateId)
    if templateParas != "" {
        param += "&templateParas=" + url.QueryEscape(templateParas)
    }
    if statusCallBack != "" {
        param += "&statusCallback=" + url.QueryEscape(statusCallBack)
    }
    return param
}

func post(url string, param []byte, headers map[string]string)(string,error) {
    tr := &http.Transport{
        TLSClientConfig: &tls.Config{InsecureSkipVerify: true},
    }
    client := &http.Client{Transport: tr}

    req, err := http.NewRequest("POST",url, bytes.NewBuffer(param));
    if err != nil {
        return "", err
    }
    for key, header := range headers {
        req.Header.Set(key, header)
    }

    resp, err := client.Do(req)
    defer resp.Body.Close()
    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }
    return string(body), nil;
}

func buildWsseHeader(appKey,appSecret string)string {
    var cTime = time.Now().Format("2006-01-02T15:04:05Z")
    var nonce = uuid.NewV4().String()
    nonce = strings.ReplaceAll(nonce,"-","")

    h := sha256.New()
    h.Write([]byte(nonce + cTime + appSecret))
    passwordDigestBase64Str := base64.StdEncoding.EncodeToString(h.Sum(nil))
```



```
    return fmt.Sprintf(WSSE_HEADER_FORMAT,appKey,passwordDigestBase64Str,nonce, cTime);
}
```

Sending SMSs in Batches

```
package main

import (
    "bytes"
    "crypto/sha256"
    "crypto/tls"
    "encoding/base64"
    "encoding/json"
    "fmt"
    "github.com/satori/go.uuid"
    "io/ioutil"
    "net/http"
    "strings"
    "time"
)

// This parameter does not need to be modified. It is used to format the Authentication header field and assign a value to the X-WSSE parameter.
const WSSE_HEADER_FORMAT = "UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s\""
// This parameter does not need to be modified. It is used to format the Authentication header field and assign a value to the Authorization parameter.
const AUTH_HEADER_VALUE = "WSSE realm=\"SDP\",profile=\"UsernameToken\",type=\"Appkey\""

func main()  {
    // Mandatory. The values here are example values only. Obtain the actual values based on Development Preparation.
    url := "https://smsapi.ap-southeast-1.myhuaweicloud.com:443/sms/batchSendDiffSms/v1" // Application access address (obtain it from the Application Management page on the console) and API access URI.
    // Hard-coded or plaintext appKey/appSecret is risky. For security, encrypt your appKey/appSecret and store them in the configuration file or environment variables.
    appKey := "c8RWg3ggEcyd4D3p94bf3Y7x1Ile" //APP_Key
    appSecret := "q4Ii87Bh**********80SfD7Al" //APP_Secret
    sender := "csms12345678" // SMS channel number.
    templateId1 := "8ff55eac1d0b478ab3c06c3c6a492300" // Template ID 1
    templateId2 := "8ff55eac1d0b478ab3c06c3c6a492300" // Template ID 2

    // Mandatory. Global number format (including the country code), for example, +9100****11. Use commas (,) to separate multiple numbers.
    receiver1 := []string{"+9100****11", "+9100****12"}; // Recipient number of template 1
    receiver2 := []string{"+9100****13", "+9100****14"}; // Recipient number of template 2

    // Optional. Address for receiving SMS status reports. The domain name is recommended. If this parameter is set to an empty value or left unspecified, customers do not receive status reports.
    statusCallBack := "";

    /**
     * Optional. If a non-variable template is used, assign an empty value to this parameter, for example, templateParas := []string{}.
     *Example of a single-variable template: If the template content is "Your verification code is ${1}", templateParas can be set to []string{"369751"}.
     * Example of a dual-variable template: If the template content is "You have ${1} delivered to ${2}", templateParas can be set to []string{"3","main gate of People's Park"}.
     * ${DATE}${TIME} cannot be empty. You can use spaces or dots (.) to replace the empty value of ${TXT_20} and use 0 to replace the empty value of ${NUM_6}.
     * To view more information, choose Service Overview > Template and Variable Specifications.
     */
    templateParas1 := []string{"123456"}; // Template 1 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).
    templateParas2 := []string{"234567"}; // Template 2 variable. The following uses a single-variable verification code SMS message as an example. The customer needs to generate a 6-digit verification code
```


and define it as a character string to prevent the loss of first digits 0 (for example, 002569 is changed to 2569).

```
    item1 := initDiffSms(receiver1, templateId1, templateParas1);
    item2 := initDiffSms(receiver2, templateId2, templateParas2);

    item := []map[string]interface{}{item1,item2}
    body := buildRequestBody(sender, item, statusCallBack)

    headers := make(map[string]string)
    headers["Content-Type"] = "application/json;charset=utf-8"
    headers["Authorization"] = AUTH_HEADER_VALUE;
    headers["X-WSSE"] = buildWsseHeader(appKey, appSecret);
    resp, err := post(url,body,headers)
    if err != nil {
        return
    }
    fmt.Println(resp);
}

func buildRequestBody(sender string, item []map[string]interface{}, statusCallBack string) []byte{
    body := make(map[string]interface{})
    body["smsContent"] = item
    body["from"] = sender
    if statusCallBack != "" {
        body["statusCallback"] = statusCallBack
    }
    res, _ := json.Marshal(body)
    return res;
}

func initDiffSms(reveiver []string, templateId string, templateParas []string) map[string]interface{} {
    diffSms := make(map[string]interface{});
    diffSms["to"] = reveiver
    diffSms["templateId"] = templateId
    if templateParas != nil && len(templateParas) > 0 {
        diffSms["templateParas"] = templateParas
    }
    return diffSms;
}

func post(url string, param []byte, headers map[string]string)(string,error) {
    tr := &http.Transport{
        TLSClientConfig: &tls.Config{InsecureSkipVerify: true},
    }
    client := &http.Client{Transport: tr}

    req, err := http.NewRequest("POST",url, bytes.NewBuffer(param));
    if err != nil {
        return "", err
    }
    for key, header := range headers {
        req.Header.Set(key, header)
    }

    resp, err := client.Do(req)
    defer resp.Body.Close()
    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }
    return string(body), nil;
}

func buildWsseHeader(appKey,appSecret string)string {

    var cTime = time.Now().Format("2006-01-02T15:04:05Z")
    var nonce = uuid.NewV4().String()
    nonce = strings.ReplaceAll(nonce,"-","")
```



```
    h := sha256.New()
    h.Write([]byte(nonce + cTime + appSecret))
    passwordDigestBase64Str := base64.StdEncoding.EncodeToString(h.Sum(nil))

    return fmt.Sprintf(WSSE_HEADER_FORMAT,appKey,passwordDigestBase64Str,nonce, cTime);
}
```

Receiving Status Reports

```
package main

import (
    "fmt"
    "net/url"
    "strings"
)

func main() {
    // Data example (urlencode) of the status report reported by the SMS platform
    //success_body :=
"sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId=2ea20735-
f856-4376-afbf-570bd70a46ee_11840135&status=DELIVRD";
    failed_body :=
"orgCode=E200027&sequence=1&total=1&updateTime=2018-10-31T08%3A43%3A41Z&source=2&smsMsgId
=2ea20735-f856-4376-afbf-570bd70a46ee_11840135&status=RTE_ERR";
    //onSmsStatusReport(success_body);
    onSmsStatusReport(failed_body);
}

func onSmsStatusReport(data string) {
    ss, _ := url.QueryUnescape(data)
    params := strings.Split(ss, "&")
    keyValues := make(map[string]string)
    for i := range params {
        temp := strings.Split(params[i],"=")
        keyValues[temp[0]] = temp[1];
    }
    /**
    * Example: Parsing status is used as an example.
    *
    * 'smsMsgId': Unique ID of an SMS
    * 'total': Number of SMS segments
    * 'sequence': Sequence number of an SMS after splitting
    * 'source': Status report source
    * 'updateTime': Resource update time
    * 'status': Enumeration values of the status report
    * 'orgCode': Status code
    */
    status := keyValues["status"];
    if status == "DELIVRD" {
        fmt.Println("Send sms success. smsMsgId: " + keyValues["smsMsgId"])
    } else {
        fmt.Println("Send sms failed. smsMsgId: " + keyValues["smsMsgId"])
        fmt.Println("Failed status:  " + keyValues["status"])
        fmt.Println("Failed orgCode:  " + keyValues["orgCode"])
    }
}
```

References

- Code examples: **Java**, **PHP**, **Python**, **C#**, and **Node.js**
- APIs: SMS Sending API, Batch SMS Sending API, and Status Report Receiving API

