

Алгоритм формирования подписи для API v 2.0.

Создать нормализованную строку запроса для использования на следующих стадиях:

1. Отсортировать параметры по имени в utf-8, сравнивая побайтово. Параметры берутся из GET URI или из тела POST-запроса (когда Content-Type равен application/x-www-form-urlencoded)
2. URL-кодировать имена и значения параметров по следующим правилам:
 - Не кодировать определённые в RFC 3986 незарезервированные символы. Эти символы: A-z, a-z, 0-9, минус (--), подчёркивание (_), точка (.) и тильда (~).
 - Все остальные символы должны быть закодированы как %XY, где X и Y -это шестнадцатеричные символы от 0 до 9 и от A до F (заглавные). Расширенные utf-8 символы кодируются как %XY%ZA...
 - Пробел кодируется как %20 (и не как +, как обычно делается в URL)
 - Кодированные имена параметров отделяются от кодированных значений знаком равно (=, ASCII-код 61), даже если параметр имеет пустое значение.
 - Пары имя – значение разделяются символом амперсанда (&, ASCII – код 38).
3. Создать строку для подписи в соответствии со следующей псевдо – грамматикой (где “\n” это ASCII – символов перевода строки):

```
1   StringToSign = HTTPVerb + "\n" +
2       ValueOfHostHeaderInLowercase + "\n" +
3       HTTPRequestURI + "\n" +
4       CanonicalizedQueryString
5
6   # HTTPVerb -- метод POST, GET, PUT или DELETE.
7   # ValueOfHostHeaderInLowercase - параметр host из заголовка HTTP запроса.
8   # HTTPRequestURI -- компонент URI, абсолютный путь до, но не включая GET--
9   параметров.
10  # Для пустого пути ожидается '/'.
    # CanonicalizedQueryString - строка с предыдущего шага.
```

4. Рассчитать совместимый с RFC 2104 HMAC по только что созданной строке StringToSign, используя секретный ключ партнёра как ключ алгоритма, и SHA256 как способ хэширования.
5. Сконвертировать полученную подпись в base64.
6. Использовать результат как значение параметра check.

[Пример расчета: php](#)

```

1  function http_build_query_rfc_3986($queryData, $argSeparator('&'))
2  {
3      $r = "";
4      $queryData = (array) $queryData;
5      if(!empty($queryData))
6      {
7          foreach($queryData as $k=>$queryVar)
8          {
9              $r .= $argSeparator.$k.'='.rawurlencode($queryVar);
10         }
11     }
12     return trim($r,$argSeparator);
13 }
14
15 function sign($method, $url, $params, $secretKey, $skipPort=False)
16 {
17     ksort($params, SORT_LOCALE_STRING);
18
19     $urlParsed = parse_url($url);
20     $path = $urlParsed['path'];
21     $host = isset($urlParsed['host'])? $urlParsed['host']: "";
22     if (isset($urlParsed['port']) && $urlParsed['port'] != 80) {
23         if (!$skipPort) {
24             $host .= ":{ $urlParsed['port']}";
25         }
26     }
27
28     $method = strtoupper($method) == 'POST'? 'POST': 'GET';
29
30     $data = implode("\n",
31         array(
32             $method,
33             $host,
34             $path,
35             http_build_query_rfc_3986($params)
36         )
37     );
38
39     $signature = base64_encode(
40         hash_hmac("sha256",
41             "{$data}",
42             "{$secretKey}",
43             TRUE
44         )
45     );
46
47     return $signature;
48 }
49
50 // Пример вызова, где $req - массив с параметрами транзакции:
51 sign("GET", "https://partner.life-pay.ru/alba/input/", $req, 'secret');
52
53
54

```

- python

```

1 from urllib.parse import urlparse,quote
2 import hashlib
3 import urllib
4 import base64
5 import hmac
6
7
8 def sign(method, url, params, secret_key, exclude=['check', 'mac']):
9     """
10    Типовой метод для подписи HTTP запросов
11    """
12    url_parsed = urlparse(url)
13    keys = [param for param in params if param not in exclude]
14    keys.sort()
15
16    result = []
17    for key in keys:
18        value = quote(
19            (params.get(key) or '').encode('utf-8'),
20            safe='~'
21        )
22        result.append('{key}={value}'.format(key, value))
23
24    data = "\n".join([
25        method,
26        url_parsed.hostname,
27        url_parsed.path,
28        "&".join(result)
29    ])
30    secrkey = secret_key.encode('utf-8')
31    mesg=data.encode('utf-8')
32    print(secrkey,mesg)
33    digest = hmac.new(
34        secrkey,
35        mesg,
36        hashlib.sha256
37    ).digest()
38    signature = base64.b64encode(digest)
39    print(signature.decode('utf-8'))
40    return signature

```

Пример вызова

```

sign("GET", "https://partner.life-pay.ru/alba/input/", {'login': 'newlogin~_-'}, '165165165sd');

```

- java (обратите внимание, пробел кодируется как %20 (и не как +, как обычно делается в URL))

```

1 package ru.life-paycb.alba;
2
3 import java.io.UnsupportedEncodingException;
4 import java.net.URLEncoder;
5 import java.nio.charset.Charset;
6 import java.security.InvalidKeyException;
7 import java.security.NoSuchAlgorithmException;
8 import java.util.ArrayList;
9 import java.util.Collections;
10 import java.util.List;
11 import java.util.Map;
12 import java.net.URI;
13 import java.net.URISyntaxException;
14 import javax.crypto.Mac;
15 import javax.crypto.spec.SecretKeySpec;
16 import javax.xml.bind.DatatypeConverter;
17
18 /**
19  * Подпись версии 2.0+
20  */
21 public class AlbaSigner {
22
23     public static String sign(String method, String url, Map<String, String> params, String secretKey)
24         throws URISyntaxException, UnsupportedEncodingException, NoSuchAlgorithmException, InvalidKeyException {
25
26         URI uri = new URI(url);
27
28         List keys = new ArrayList<>(params.keySet());
29         Collections.sort(keys);
30
31         StringBuilder sb = new StringBuilder();
32         for (String key: keys) {
33             if (sb.length() > 0) {
34                 sb.append("&");
35             }
36
37             sb.append(String.format("%s=%s", key, URLEncoder.encode(params.get(key), "UTF-8")));
38         }
39         String urlParameters = sb.toString();
40         String data = method.toUpperCase() + "\n" +
41             uri.getHost() + "\n" +
42             uri.getPath() + "\n" +
43             urlParameters;
44
45         Mac hmacInstance = Mac.getInstance("HmacSHA256");
46         Charset charSet = Charset.forName("UTF-8");
47         SecretKeySpec keySpec = new SecretKeySpec(charSet.encode(secretKey).array(), "HmacSHA256");
48         hmacInstance.init(keySpec);
49
50         return DatatypeConverter.printBase64Binary(hmacInstance.doFinal(data.getBytes("UTF-8")));
51     }
52 }

```