# API v 2.0.

:

1. utf-8, . GET URI POST- ( Content-Type application/x-www-form-urlencoded)
2. URL- :
   - RFC 3986 . : A-z, a-z, 0-9, (--), (_), (.) (~).
   - %XY, X Y - 0 9 A F (). utf-8 %XY%ZA…
   - %20 ( +, URL)
   - (=, ASCII- 61), .
   - – (&, ASCII – 38).
3. – ( “\n” ASCII – ):

```
StringToSign = HTTPVerb + "\n" +
               ValueOfHostHeaderInLowercase + "\n" +
               HTTPRequestURI + "\n" +
               CanonicalizedQueryString

# HTTPVerb -- POST, GET, PUT DELETE.
# ValueOfHostHeaderInLowercase - host HTTP .
# HTTPRequestURI -- URI, , GET--.
#  '/'.
# CanonicalizedQueryString - .
```

4. RFC 2104 HMAC StringToSign, , SHA256 .
5. base64.
6. check.

:php

```php
    function http_build_query_rfc_3986($queryData, $argSeparator='&')
    {
        $r = '';
        $queryData = (array) $queryData;
        if(!empty($queryData))
        {
            foreach($queryData as $k=>$queryVar)
            {
                $r .= $argSeparator.$k.'='.rawurlencode($queryVar);
            }
        }
        return trim($r,$argSeparator);
    }

    function sign($method, $url, $params, $secretKey, $skipPort=False)
    {
        ksort($params, SORT_LOCALE_STRING);

        $urlParsed = parse_url($url);
        $path = $urlParsed['path'];
        $host = isset($urlParsed['host'])? $urlParsed['host']: '';
        if (isset($urlParsed['port']) && $urlParsed['port'] != 80) {
            if (!$skipPort) {
                $host .= ':{$urlParsed['port']}';
            }
        }

        $method = strtoupper($method) == 'POST'? 'POST': 'GET';

        $data = implode("\n",
            array(
                $method,
                $host,
                $path,
                http_build_query_rfc_3986($params)
            )
        );

        $signature = base64_encode(
            hash_hmac("sha256",
                "{$data}",
                "{$secretKey}",
                TRUE
            )
        );

        return $signature;
    }

//  ,  $req -    :
sign("GET", "https://partner.rficb.ru/alba/input/", $req, 'secret');
```

- python

```
from urllib.parse import urlparse,quote
import hashlib
import urllib
import base64
import hmac


def sign(method, url, params, secret_key, exclude=['check', 'mac']):
    """
        HTTP
    """
    url_parsed = urlparse(url)
    keys = [param for param in params if param not in exclude]
    keys.sort()

    result = []
    for key in keys:
        value = quote(
            (params.get(key) or '').encode('utf-8'),
            safe='~'
        )
        result.append('{}={}'.format(key, value))

    data = "\n".join([
        method,
        url_parsed.hostname,
        url_parsed.path,
        "&".join(result)
    ])
    secrkey = secret_key.encode('utf-8')
    mesg=data.encode('utf-8')
    print(secrkey,mesg)
    digest = hmac.new(
        secrkey,
        mesg,
        hashlib.sha256
    ).digest()
    signature = base64.b64encode(digest)
    print(signature.decode('utf-8'))
    return signature

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

- java

```java
package ru.rficb.alba;

import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.nio.charset.Charset;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Map;
import java.net.URI;
import java.net.URISyntaxException;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;

/**
*   2.0+
*/
public class AlbaSigner {

    public static String sign(String method, String url, Map<String, String> params, String secretKey)
            throws URISyntaxException, UnsupportedEncodingException, NoSuchAlgorithmException, InvalidKeyException {

        URI uri = new URI(url);

        List keys = new ArrayList<>(params.keySet());
        Collections.sort(keys);

        StringBuilder sb = new StringBuilder();
        for (String key: keys) {
            if (sb.length() > 0) {
                sb.append("&");
            }

            sb.append(String.format("%s=%s", key, URLEncoder.encode(params.get(key), "UTF-8")));
        }
        String urlParameters = sb.toString();
        String data = method.toUpperCase() + "\n" +
                uri.getHost() + "\n" +
                uri.getPath() + "\n" +
                urlParameters;

        Mac hmacInstance = Mac.getInstance("HmacSHA256");
        Charset charSet = Charset.forName("UTF-8");
        SecretKeySpec keySpec = new javax.crypto.spec.SecretKeySpec(charSet.encode(secretKey).array(), "HmacSHA256");
        hmacInstance.init(keySpec);

        return DatatypeConverter.printBase64Binary(hmacInstance.doFinal(data.getBytes("UTF-8")));
    }
}
```