

master


...

paypal-adaptive-sdk-nodejs / lib / paypal-adaptive.js / &lt;&gt; Jump to



artur-krueger Set subject header if configured ...

History

3 contributors   

252 lines (284 sloc) | 7.75 KB

...

```
1  var https = require('https')
2    , util = require('util');
3
4  var adaptiveAccountsMethods = [
5    'AddBankAccount',
6    'AddPaymentCard',
7    'CheckComplianceStatus',
8    'CreateAccount',
9    'GetUserAgreement',
10   'GetVerifiedStatus',
11   'SetFundingSourceConfirmed',
12   'UpdateComplianceStatus'
13 ];
14
15 var adaptivePaymentsMethods = [
16   'CancelPreapproval',
17   'ConvertCurrency',
18   'ExecutePayment',
19   'GetFundingPlans',
20   'GetShippingAddresses',
21   'PreapprovalDetails',
22   'SetPaymentOptions'
23 ];
24
25 function merge(a, b) {
26   for (var p in b) {
27     try {
28       if (b[p].constructor === Object) {
29         a[p] = merge(a[p], b[p]);
30       } else {
31         a[p] = b[p];
32       }
33     } catch (e) {
34       a[p] = b[p];
35     }
36   }
37   return a;
38 }
39
40 function defaultPayload() {
41   return {
42     requestEnvelope: {
43       errorLanguage: 'en_US',
44       detailLevel: 'ReturnAll'
45     }
46   };
47 }
48
49 function httpsPost(options, callback) {
50   options.method = 'POST';
51   options.headers = options.headers || {};
52
53   var data = (typeof options.data !== 'string') ? JSON.stringify(options.data) : options.data;
54
55   options.headers['Content-Length'] = Buffer.byteLength(data);
56
57   var req = https.request(options);
58
59   req.on('response', function (res) {
60     var response = '';
61     //do not setEncoding with browserify
62     if (res.setEncoding) {
63       res.setEncoding('utf8');
64     }
65
66     res.on('data', function (chunk) {
67       response += chunk;
68     });
69
70     res.on('end', function () {
71       return callback(null, {
72         statusCode: res.statusCode,
73         body: response
74       });
75     });
76   });
77
78   req.on('error', function (e) {
```

```

79     callback(e);
80 });
81
82 if (data) {
83     req.write(data);
84 }
85
86 req.end();
87 }
88
89 var Paypal = function (config) {
90     if (!config) throw new Error('Config is required');
91     if (!config.userId) throw new Error('Config must have userId');
92     if (!config.password) throw new Error('Config must have password');
93     if (!config.signature) throw new Error('Config must have signature');
94     if (!config.appId && !config.sandbox) throw new Error('Config must have appId');
95
96     var defaultConfig = {
97         requestFormat: 'JSON',
98         responseFormat: 'JSON',
99         sandbox: false,
100         productionHostname: 'svcs.paypal.com',
101         sandboxHostname: 'svcs.sandbox.paypal.com',
102         appId: 'APP-80W284485P519543T',
103         approvalUrl: 'https://www.paypal.com/cgi-bin/webscr?cmd=_ap-payment&paykey=%s',
104         sandboxApprovalUrl: 'https://www.sandbox.paypal.com/cgi-bin/webscr?cmd=_ap-payment&paykey=%s',
105         preapprovalUrl: 'https://www.paypal.com/webscr?cmd=_ap-preapproval&preapprovalkey=%s',
106         sandboxPreapprovalUrl: 'https://www.sandbox.paypal.com/webscr?cmd=_ap-preapproval&preapprovalkey=%s'
107     };
108
109     this.config = merge(defaultConfig, config);
110 };
111
112 Paypal.prototype.callApi = function (apiMethod, data, callback) {
113     var config = this.config;
114
115     var options = {
116         hostname: config.sandbox ? config.sandboxHostname : config.productionHostname,
117         port: 443,
118         path: '/' + apiMethod,
119         data: data,
120         headers: {
121             'X-PAYPAL-SECURITY-USERID': config.userId,
122             'X-PAYPAL-SECURITY-PASSWORD': config.password,
123             'X-PAYPAL-SECURITY-SIGNATURE': config.signature,
124             'X-PAYPAL-APPLICATION-ID': config.appId,
125             'X-PAYPAL-REQUEST-DATA-FORMAT': config.requestFormat,
126             'X-PAYPAL-RESPONSE-DATA-FORMAT': config.responseFormat
127         }
128     };
129
130     if (config.sandboxEmailAddress)
131         options.headers['X-PAYPAL-SANDBOX-EMAIL-ADDRESS'] = config.sandboxEmailAddress;
132
133     if (config.deviceIpAddress)
134         options.headers['X-PAYPAL-DEVICE-IPADDRESS'] = config.deviceIpAddress;
135
136     if (config.subject)
137         options.headers['X-PAYPAL-SECURITY-SUBJECT'] = config.subject;
138
139     httpsPost(options, function (error, response) {
140         if (error) { return callback(error); }
141
142         var body = response.body;
143         var statusCode = response.statusCode;
144
145         if (config.responseFormat === 'JSON') {
146             try {
147                 body = JSON.parse(body);
148             } catch (e) {
149                 var err = new Error('Invalid JSON Response Received');
150                 err.response = body;
151                 err.httpStatusCode = response.statusCode;
152                 return callback(err);
153             }
154         }
155
156         if (statusCode < 200 || statusCode >= 300) {
157             error = new Error('Response Status: ' + statusCode);
158             error.response = body;
159             error.httpStatusCode = statusCode;
160             return callback(error);
161         }
162
163         body.httpStatusCode = statusCode;
164
165         if (/^(Success|SuccessWithWarning)$/.test(body.responseEnvelope.ack)) {
166             callback(null, body);
167         } else {
168             var err = new Error('Response ack is ' + body.responseEnvelope.ack + '. Check the response for more info');
169             return callback(err, body);
170         }
171     });
172 };
173
174 // Paypal Adaptive Payments API methods
175 Paypal.prototype.getPaymentOptions = function (payKey, callback) {
176     if (!payKey) {

```

```

177         return callback(new Error('Required "payKey"'));
178     }
179
180     var data = defaultPayload();
181     data.payKey = payKey;
182
183     this.callApi('AdaptivePayments/GetPaymentOptions', data, callback);
184 };
185
186 Paypal.prototype.paymentDetails = function (params, callback) {
187     if (!params.payKey && !params.transactionId && !params.trackingId) {
188         return callback(new Error('Required "payKey" or "transactionId" or "trackingId" on first param'));
189     }
190
191     var data = merge(defaultPayload(), params);
192
193     this.callApi('AdaptivePayments/PaymentDetails', data, callback);
194 };
195
196 Paypal.prototype.pay = function (data, callback) {
197     var config = this.config;
198
199     this.callApi('AdaptivePayments/Pay', data, function (err, res) {
200         if (err) { return callback(err, res); }
201
202         if (res.paymentExecStatus === 'CREATED') {
203             var url = config.sandbox ? config.sandboxApprovalUrl : config.approvalUrl;
204             res.paymentApprovalUrl = util.format(url, res.payKey);
205         }
206
207         return callback(null, res);
208     });
209 };
210
211 Paypal.prototype.preapproval = function (data, callback) {
212     var config = this.config;
213
214     this.callApi('AdaptivePayments/Preapproval', data, function (err, res) {
215         if (err) { return callback(err, res); }
216
217         if (res.preapprovalKey) {
218             var url = config.sandbox ? config.sandboxPreapprovalUrl : config.preapprovalUrl;
219             res.preapprovalUrl = util.format(url, res.preapprovalKey);
220         }
221
222         return callback(null, res);
223     });
224 };
225
226 Paypal.prototype.refund = function (params, callback) {
227     if (!params.payKey && !params.transactionId && !params.trackingId) {
228         return callback(new Error('Required "payKey" or "transactionId" or "trackingId" on first param'));
229     }
230
231     var data = merge(defaultPayload(), params);
232
233     this.callApi('AdaptivePayments/Refund', data, callback);
234 };
235
236 adaptivePaymentsMethods.forEach(function (method) {
237     var prototypeMethodName = method.charAt(0).toLowerCase() + method.slice(1);
238     var apiMethodName = 'AdaptivePayments/' + method;
239     Paypal.prototype[prototypeMethodName] = function (data, callback) {
240         this.callApi(apiMethodName, data, callback);
241     };
242 });
243
244 adaptiveAccountsMethods.forEach(function (method) {
245     var prototypeMethodName = method.charAt(0).toLowerCase() + method.slice(1);
246     var apiMethodName = 'AdaptiveAccounts/' + method;
247     Paypal.prototype[prototypeMethodName] = function (data, callback) {
248         this.callApi(apiMethodName, data, callback);
249     };
250 });
251
252 module.exports = Paypal;

```