vSafe™ v3.3.1 vGuaranteed™ Integration Guide

ThreatMetrix® Fingerprint Solution

January 19, 2017

Document Version: 2.0

Vesta Corporation (Americas)

4400 Alexander Drive
Alpharetta, GA 30022-3753 USA
Tel: +1 678.222.7200
Fax: +1 770.772.0600
www.trustvesta.com
info@trustvesta.com

11950 SW Garden Place
Portland, OR 97223-8248 USA
Tel: +1 503.790.2500
Fax: +1 503.790.252
info@trustvesta.com
vsafe.support@trustvesta.com

Av. Américas 1536 1A
Col. Country Club
C.P. 44637
Jalisco, México
info@trustvesta.com

Vesta Payment Solutions Limited (Europe)

Vesta Building
Finnabair Business Park
Dundalk
Co. Louth
Ireland A91 E934
Tel: +353 (0) 42 939 4600
Fax: +353 (0) 42 933 4121
info.ireland@trustvesta.com

Release History

Version	Date	Contributors	Notes
1.0	October 23, 2015	Vesta Market Solutions	Initial publication.
1.1	October 28, 2015	Vesta Market Solutions	Corrected code samples and reviewed content for technical accuracy and completeness.
1.1.1	November 2, 2015	Vesta Market Solutions	Corrected tokenization JavaScript file name.
1.1.2	December 11, 2015	Vesta Market Solutions	Updated Appendix A to include vGuaranteed™ Risk Information document.
1.1.3	December 23, 2015	Vesta Market Solutions	Added a note for using cURL code sample from the Windows® command prompt.
1.1.4	January 29, 2016	Vesta Market Solutions	Updated vGuaranteed™ API specification version number from v3.1 to v3.3.
1.1.5	February 17, 2016	Vesta Market Solutions	Updated content for vSafe™ v3.3 push to production.
1.2	March 16, 2016	Vesta Market Solutions	Updated content for vSafe™ v3.3.1 push to production.
1.2.1	April 5, 2016	Vesta Market Solutions	Published version specifically for partners using ThreatMetrix® fingerprint solution.
2.0	January 19, 2016	Vesta Market Solutions Vesta Service Delivery	Revised content for the vPortal™ push to production.

Contents

Figures

Tables

Introduction

This document is designed as a reference guide for software developers and architects who will be implementing, testing, and validating card-not-present (CNP) transactions using the vSafe™ vGuaranteed™ ChargeSale API call.

vGuaranteed™ one-step payment processing authorizes and completes a CNP transaction using the ChargeSale API call. Even though the transaction is authorized and completed in a single step, the process involves three API calls—GetSessionTags, ChargeAccountToTemporaryToken, and ChargeSale.

When a customer initiates a CNP transaction, the partner's application calls the GetSessionTags API to obtain device fingerprinting data. The GetSessionTags API call returns information that is embedded in a JavaScript call and placed on the Web form, along with the tokenization JavaScript and library that are used to create a temporary token. The temporary token and WebSessionID are then sent to the ChargeSale API to authorize and complete the transaction in one step.

Note: The vSafe™ v3.3.1 vGuaranteed™ API provides a two-step payment processing use case that authorizes a payment but delays completion of the payment until a second API is called. The two-step payment process is not addressed in this document. For information on the two-step payment process, see the vSafe™ v3.3.1 vGuaranteed™ API Use Cases document.

Figure 1 describes the one-step payment process used for capturing card details and calling the ChargeSale API. This flow assumes that the transaction is approved and completed after the ChargeSale API is called.

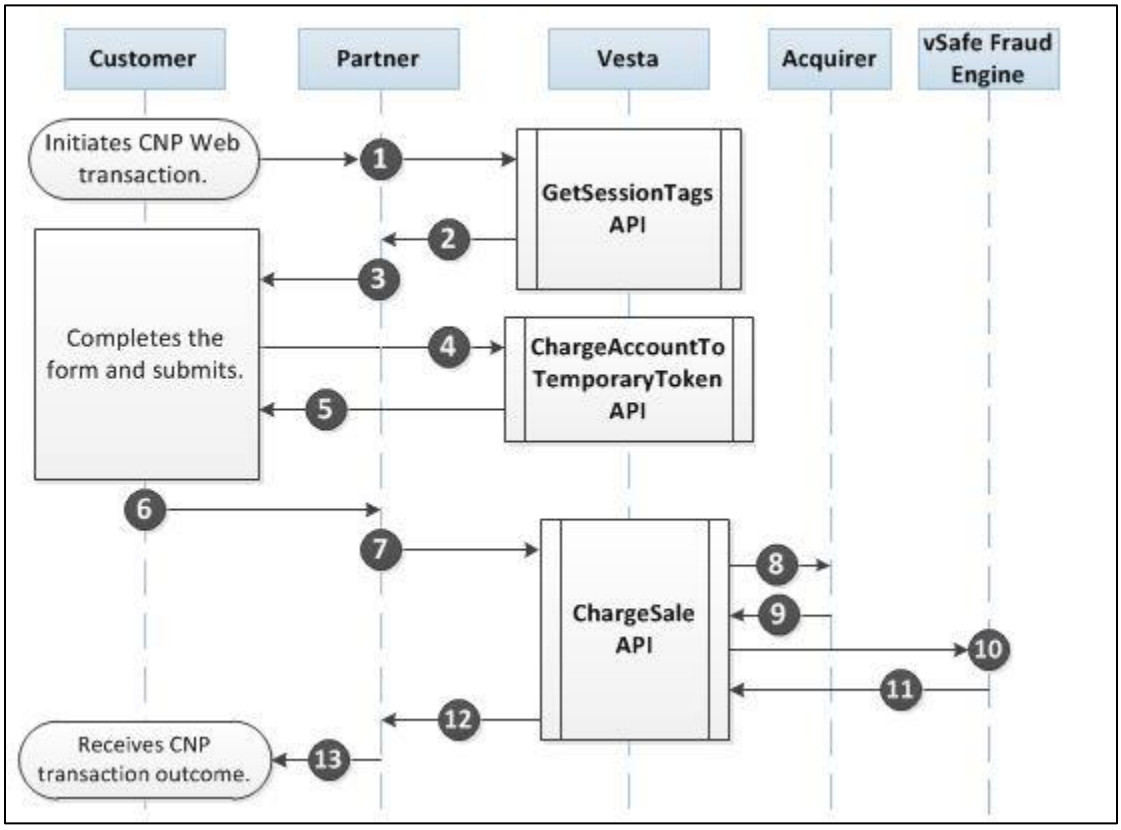

Figure 1. vGuaranteed™ One-Step Payment Flow Diagram (WEB)

Table 1 provides details about the callouts in Figure 1.

Table 1. vGuaranteed™ One-Step Payment Flow Description

Callout	Description
1	The partner receives the payment request from the customer and calls the GetSessionTags API.
2	The GetSessionTags API returns the WebSessionID and OrgID parameters to the partner.
3	The partner replaces the WebSessionID and OrgID in the device fingerprinting JavaScript, generates the form page, and then adds the tokenization JavaScript and library, device fingerprinting JavaScript, and form fields to capture the card and customer details.
4	After the customer submits the payment request, the form JavaScript invokes the tokenization library, which submits the account number to Vesta to be tokenized.
5	Vesta returns a temporary token to the customer's client browser.
6	The form JavaScript replaces the card number stored on the client form with the temporary token and submits the form to the partner's Web server.

Callout	Description
7	The partner assembles the risk information and then calls the ChargeSale API with the risk information, temporary token, and all of the customer's information.
8	vSafe™ requests the transaction authorization from the acquirer.
9	The acquirer returns the authorization response to vSafe™.
10	The vSafe™ fraud engine evaluates the fraud risk.
11	The vSafe™ fraud engine returns the fraud response to vSafe™.
12	vSafe™ returns the payment status to the partner.
13	The partner sends the payment outcome to the customer's browser.

Connecting to vSafe™ vPortal™

Vesta offers a secure vSafe™ vPortal™ environment for application testing. To connect to the vSafe™ vPortal™ environment, you will need the following information:

- vPortal™ account
- API username
- API password
- vPortal™ Sandbox URLs
- Risk information XML blob
- Tokenization JavaScript

vPortal™ Account

To create a vPortal™ account, please register using the following URL: `https://vsafeportal.trustvesta.com/`.

API Username and Password

After creating a vPortal™ account, you can retrieve your API user name and credentials by logging in to your vPortal™ account and clicking the Account Info link on your vPortal™ landing page, as shown in Figure 2.

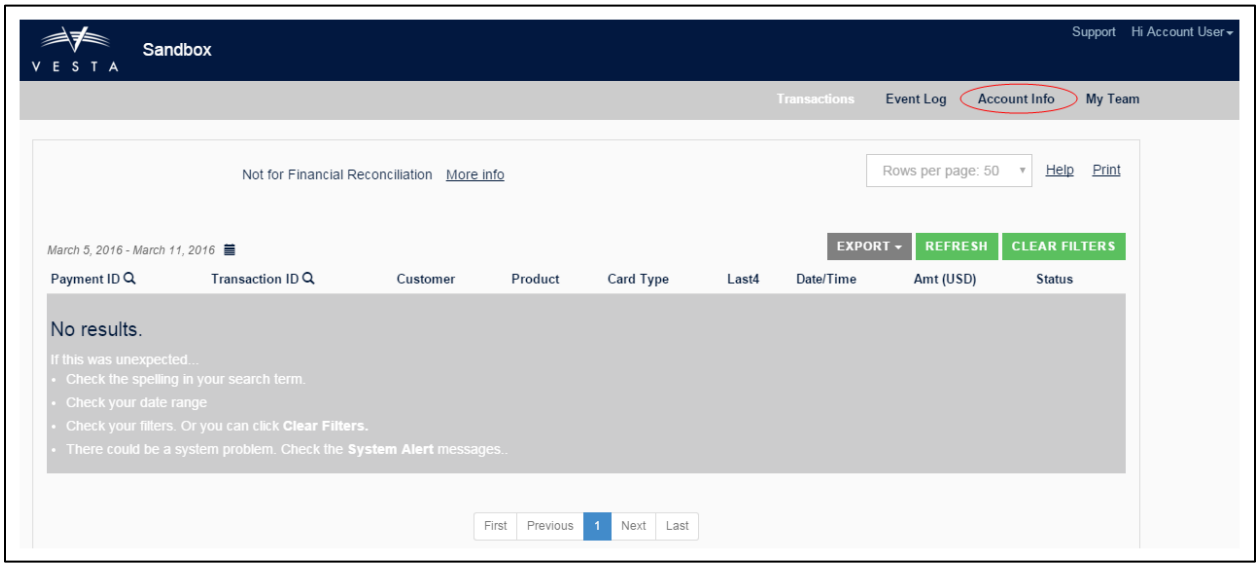

Figure 2. vPortal™ Account Info Link

After clicking Account Info, retrieve your API username and API password, as shown in Figure 3.

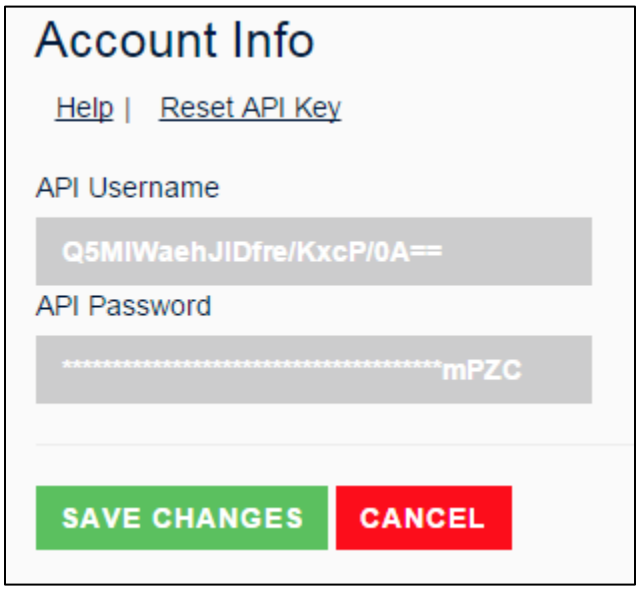

Figure 3. API Username and API Password

vPortal™ Sandbox URLs

Use the URLs in Table 2 to access the Payment API, Tokenization API, and Fingerprint API.

Table 2. Payment API, Tokenization API, and Fingerprint API URLs

Application	URL
Payment API	`https://vsafesandbox.ecustomersupport.com/GatewayV4Proxy/Service`
Tokenization API	`https://vsafesandboxtoken.ecustomersupport.com/GatewayV4ProxyJSON/Service`
Fingerprint API	`https://fingerprint.ecustomerpayments.com/ThreatMetrixUIRedirector`

Risk Information XML Blob

The risk information XML blob is customized to your business market. For example, the risk information XML blob will be different for a business that issues gift cards than a risk information XML blob for a business that performs money transfers. For information about creating a custom risk information XML blob, see the vSafe™ v3.x vGuaranteed™ Risk Information document located on the Vesta Tech Docs page (`http://trustvesta.com/technical-documents/`).

Tokenization JavaScript

To obtain the tokenization JavaScript, contact your Vesta Service Delivery Manager.

Device Fingerprinting

Device fingerprinting integration allows Vesta's risk team to conduct real-time monitoring by detecting transaction and device anomalies and by enabling detection of the transaction's originating-device IP address. Device fingerprinting is required for all Web transactions.

Implementation

This section describes the information that is required on the Web form.

When the GetSessionTags API is called, a `{WebSessionID}` and `{OrgID}` are returned. These two parameters must be embedded in the device fingerprinting JavaScript on the Web form.

Note: The script could take up to five seconds to process the tag links. Vesta recommends placing the tag links code at the bottom of the page and disabling the Submit button until the page loads completely.

When the form is submitted to the Web server, the `{WebSessionID}` parameter is required for the ChargeSale API call.

Sample Code

Copy and paste the following code into your Web form, and then replace the information within the brackets with the OrgID and WebSessionID response parameters returned by the GetSessionTags API call.

```
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c"%>
<!-- Begin fingerprinting tags below -->
<p
style="background:url(https://fingerprint.ecustomerpayments.com/ThreatMetrixUIRedirector/fp/clear
.png?org_id="{OrgID}"&session_id="{WebSessionID}"&m=1)"></p>
    <img
        src="https://fingerprint.ecustomerpayments.com/ThreatMetrixUIRedirector/fp/clear.png?org
        _id="{OrgID}"&session_id={WebSessionID}"&m=2" alt=""/>
    <script
        src="https://fingerprint.ecustomerpayments.com/ThreatMetrixUIRedirector/fp/check.js?org_i
        d"{OrgID}"&session_id="{WebSessionID}" type="text/javascript"></script>
    <object data="
        https://fingerprint.ecustomerpayments.com/ThreatMetrixUIRedirector/fp/fp.swf?org_id="{Or
        gID}"&session_id="{WebSessionID}" type="application/x-shockwave-flash" width="1"
        height="1" id="obj_id">
            <param value="
                https://fingerprint.ecustomerpayments.com/ThreatMetrixUIRedirector/fp/fp.swf?org
                _id="{OrgID}"&session_id="{WebSessionID}" name="movie" />
    </object>
<!-- End fingerprinting tags -->
```

Figure 4. Web Fingerprint Code Sample

Temporary Tokens

Temporary tokens are used for one-time payments and are not stored permanently in vSafe™. You must obtain a copy of the `vestatoken-1.0.3.js` file from Vesta to use tokenization (see Connecting to vSafe™ on page 4).

Implementation

To create a temporary token, copy and paste the following sample code into your Web form, and then replace the information in the brackets with the information provided by Vesta (see Connecting to vSafe™ on page 4).

Sample Code

In the Web form header, add the tokenization library, and then initialize the Vesta token object using the code in Figure 5.

```
<script type="text/javascript" src="{Path to JS File}/vestatoken-1.0.3.js"></script>
<script type="text/javascript">
        vestatoken.init({
                ServiceURL      : "{Tokenization API}",
                AccountName     : "{API Username}"
        });
</script>
```

Figure 5. Add the Libraries and Initialize the Vesta Token Object

After adding the libraries and initializing the Vesta token object, add fields to the page to collect the customer information, payment information, and credit card number using the code in Figure 6.

```
<form id="paymentForm" method="post">
        <!-- Add additional form fields -->
        <input type="text" id="pdCardNumber" />
        <input type="submit" />
</form>
```

Figure 6. Collect the Credit Card Data

On submission, the Web form JavaScript will invoke the `vesta.token-1.0.3.js` file, which will return a charge account number token. Using the sample code in Figure 7, replace the card number on the form with the token before submitting it to the Web server.

```
<script type="text/javascript">
document.getElementById('paymentForm').onsubmit = function() {
        vestatoken.getcreditcardtoken({
                ChargeAccountNumber: document.getElementById('pdCardNumber').value,
                onSuccess: function(data) {
                        // make use of 'data.ChargeAccountNumberToken' (String),
                        'data.PaymentDeviceLast4' (String) and 'data.PaymentDeviceTypeCD'
                        (String)
                },
                onFailed: function(failure) {
                        // make use of `failure` (String)
                },
                onInvalidInput: function(failure) {
                        // make use of `failure` (String)
                }
        });
        return false;
}
</script>
```

Figure 7. Tokenization of Account Number

vGuaranteed™ One-Step Payment Processing

vSafe™ supports communications using REST and JSON over the HTTPS protocol. POST should be used to submit a request. All responses are handled using the ResponseCode and ResponseText fields in the response portion of each API.

One-step payment processing authorizes and completes a CNP transaction using the ChargeSale API call. The ChargeSale API call is the most common vGuaranteed™ API call used to process payments.

Implementation

After fingerprinting the device (see Device Fingerprinting on page 6) and tokenizing the account number (see Temporary Tokens on page 7), the Web form is submitted to the partner's Web server to be processed. The partner must create the risk information XML that will be used for risk analysis. The Fingerprint, card account number token, risk information, customer information, and order details are then passed to the vSafe™ API for processing.

Sample Code

Use the following Java code to call the ChargeSale API. This sample uses the library `org.apache.http`.

When using the input sample code, you must replace information in the brackets with the information provided by Vesta (see Connecting to vSafe™ on page 4) or with parameters that were returned in previous API calls.

Note: This sample does not take into consideration any underlying errors or exceptions.

```
HttpClient client = new DefaultHttpClient();
HttpPost post = new HttpPost("{Payment API}");

ArrayList<NameValuePair> payload = new ArrayList<NameValuePair>();
payload.add(new BasicNameValuePair("TransactionID", UUID.randomUUID().toString()));
payload.add(new BasicNameValuePair("AccountName", "{ API username}"));
payload.add(new BasicNameValuePair("Password", "{API password}"));
payload.add(new BasicNameValuePair("CardHolderAddressLine1", "{Form Data}");
payload.add(new BasicNameValuePair("CardHolderCity", "{Form Data}");
payload.add(new BasicNameValuePair("CardHolderRegion", "{Form Data}");
payload.add(new BasicNameValuePair("CardHolderPostalCode", "{Form Data}");
payload.add(new BasicNameValuePair("CardHolderCountryCode", "{Form Data}");
payload.add(new BasicNameValuePair("CardHolderFirstName", "{Form Data}");
payload.add(new BasicNameValuePair("CardHolderLastName", "{Form Data}");
payload.add(new BasicNameValuePair("ChargeAccountNumber", "{ChargeAccountNumberToken}");
```

```
payload.add(new BasicNameValuePair("ChargeAccountNumberIndicator", "2");
payload.add(new BasicNameValuePair("ChargeAmount", {Form Data});
payload.add(new BasicNameValuePair("ChargeCVN", "{Form Data}");
payload.add(new BasicNameValuePair("ChargeExpirationMMYY", "{Form Data}");
payload.add(new BasicNameValuePair("ChargeSource", "WEB");
payload.add(new BasicNameValuePair("RiskInformation", "{RI XML Blob}");
payload.Add(new BasicNameValuePair("StoreCard", "false"));
payload.add(new BasicNameValuePair("WebSessionID", "{WebSessionID}");
payload.add(new BasicNameValuePair("MerchantRoutingID", "SandboxCredit01");

try {
  post.setEntity(new UrlEncodedFormEntity(payload, "utf-8"));

  HttpResponse response = client.execute(post);

  List<NameValuePair> resultList =
URLEncodedUtils.parse(IOUtils.toString(response.getEntity().getContent(), "utf-8"),
Charset.forName("utf-8"));
  HashMap<String, String> result = new HashMap<String, String>();
  for (NameValuePair item : resultList) {
    result.put(item.getName(), item.getValue());
  }
  if ("0".equals(result.get("ResponseCode"))) {
    String paymentID = result.get("PaymentID");
  } else {
    System.err.println(result.get("ResponseText"));
  }
} catch (Exception ex) {
  System.err.println(ex.getMessage());
}
```

Testing and Validation

This section describes how to test connectivity, account credentials, and API parameters. The following samples will return a successful response from the vPortal™. Use the following cURL scripts for sanity-checking and to test different request and response parameters.

Note: ChargeAccountToTemporaryToken should never be called from the partner's Web server. This call is invoked directly from the client's browser and is sent directly to vSafe™ using the tokenization JavaScript.

When using the input sample code in Table 3, you must replace information in the brackets with the information provided by Vesta (see Connecting to vSafe™ on page 4) or with parameters that were returned in previous API calls.

Note: Windows® users may experience problems using cURL from the Windows® command prompt. Vesta recommends using a UNIX®-like command-line interface, like Cygwin©.

Table 3. API Calls and Associated Input and Output Code

API Call	I/O	Sample Code
GetSessionTags	Input	`curl {Payment API}/GetSessionTags -H 'Content-Type: application/json' -d '{"AccountName":"{API Username}", "Password":"{API Password}", "TransactionID":"TestGetSessionTags"}'`
	Output	`{"OrgID":"f33plvkc","WebSessionID":"101_250372633","ResponseCode":"0","ProxiedMessageName":"GetSessionTags"}'`
ChargeAccountToTemporaryToken	Input	`curl {Payment API}/ChargeAccountToTemporaryToken -H 'Content-Type: application/json' -d '{"AccountName":"{API Username}", "ChargeAccountNumber":"{Card Number}"}'`
	Output	`{"PaymentDeviceTypeCD":"4","PaymentDeviceLast4":"1111","ChargeAccountNumberToken":"4111800000000050","ResponseCode":"0","ProxiedMessageName":"ChargeAccountToTemporaryToken"}'`
ChargeSale	Input	`curl {Payment API}/ChargeSale -H 'Content-Type: application/json' -d '{"AccountName":"{API Username}", "Password":"{API Password}", "TransactionID":"TestChargeSale1", "CardHolderAddressLine1":"123 Main St.", "CardHolderCity":"Portland", "CardHolderRegion":"OR", "CardHolderPostalCode":"97034", "CardHolderCountryCode":"US", "CardHolderFirstName":"John", "CardHolderLastName":"Doe", "ChargeAccountNumber":"{ChargeAccountNumberToken}", "ChargeAmount":"12.34", "ChargeCVN":"123", "ChargeAccountNumberIndicator":"2" "ChargeExpirationMMYY":"1118", "ChargeSource":"WEB",`

API Call	I/O	Sample Code
		"RiskInformation":"<riskinformation/>", "StoreCard":"false", "WebSessionID":"{WebSessionID}", "MerchantRoutingID":"SandboxCredit01"}'
	Output	{"ResponseCode":"0","PaymentAcquirerName":"Chase Paymentech","ChargeAccountLast4":"0015","PaymentID":"47USA1M4T","AuthorizedAmount":"12.34","PaymentDeviceTypeCD":"5","ChargeAccountFirst6":"513304","PaymentStatus":"10"}

Appendix A: Related Documents

After successfully completing a one-step payment as described in this document, please review the following related vGuaranteed™ documents:

- vSafe™ v3.x vGuaranteed™ API Use Cases
- vSafe™ v3.x vGuaranteed™ API Specification
- vSafe™ v3.x vGuaranteed™ Risk Information
- vSafe™ v3.x vGuaranteed™ TM Fingerprint Solution