Vermont Information Processing, Inc.
SRS Product Finder - Web Services Documentation

Last Modified: February 2017

Summary

Your web application server can query VIP's web application server for Product Finder data. There are three main service endpoints:

1. **Brands and stores:**
   - Contains a list of all recently purchased brands at your locations. The timeframe of "recent" can be configured; contact VIP.
   - Contains a list of store types. Also known as Class of Trade, or COT.
   - *Should be queried no more than once a day* because the data is only updated once a day (and typically changes even less often).
   - *However, can be queried on demand if using custom input filters for advanced search UIs.*

2. **Categories:**
   - Contains a list of any single category's values based on the current Item Master database maintained between VIP and the customer.
   - Categories 1 through 12 can be queried. One request per category.
   - *Should be queried no more than once a day* because the data rarely changes.
   - *However, can be queried on demand if using custom input filters for advanced search UIs.*

3. **Location results:**
   - Contains a list of matching locations that have sold the chosen product near the chosen location.
   - Is queried in real-time based on user input into your web application.

Requirements

1. Your own web application server.

2. The network infrastructure to make and receive HTTP GET or POST requests over port 80 or 443 to VIP's web application server.

3. The ability to save & cache certain simple, rarely-changing lists of data (brands, stores and categories).

4. The ability to perform a SHA-256 encryption algorithm to create a signature string for authentication.

Implementation Overview

1. VIP will provide you with both a VIP ID (custID) and a shared secret (a secret, random password-like string that helps with authentication).

2. Review the VIP web service XML standards.

3. Create your web application or a working prototype to test connectivity.

4. Test your connection to VIP from your web server over port 80 or 443.

5. Test a sample web service request to VIP to ensure your authentication process is valid.

6. Test your application using VIP web services.

7. Review your web application with VIP.

8. Consult your regional managers over the possible restrictions applied to this type of this service because it goes across state lines. For instance, Texas once restricted usage of this service for companies not based within Texas – but that is no longer the case. Give VIP a list of these states so we can exclude them.

9. Go live!

Secure Your Data

Your data is extremely important to you, VIP, and your competition. VIP recommends following these standards to help keep your data safe and lower the risk of widely exposing where your brands are sold::

1. Ensure queries in your Finder **cannot be submitted via a script**. Use a session token (random string) that you generate on Finder form load. Pass that token on the Finder page submit so the server can validate the token. The token should change on every new page load. The submit validation process should only accept that 1 new token for the user.

2. **Prevent direct calling** of your Finder submit page by domains other than your own. Check the referral HTTP header for this. Block blank referral values and those not coming from your domain (or sub-domains).

3. **Validate all user input** data before submitting them to the VIP Finder. If any piece of data is unexpected, then revert the input to known defaults or prevent the submission.

4. **Diligently monitor your Finder activity.** Consider logging the IPs and user-agents of users in a database and counting use per hour or day. Stop continued use if over a certain threshold. Or provide the IP and user-agent to the VIP service so we can rate-limit.

These are just a few of the more common ways to secure your web service implementation of the VIP Product Finder web service.

Request Details – *all service endpoints*

Method: HTTP GET or POST request
URL to query: https://www.vtinfo.com/PF/product_finder-service.asp

All requests share these settings. Most are required but some are optional and available on every request.

GET or POST input parameters:

custID	[your VIP customer ID; a 3 or 5 character code]
terr	[optional] The territory number setup by VIP to group your products.
format	[optional] "XML" is the only option at this time
ip	[optional] the end users IP address; v4 or v6; strongly recommended for security
userAgent	[optional] the end users browser/device user-agent; same reason as IP Examples can be seen here: http://www.useragentstring.com/pages/Browserlist/

HTTP Request Headers:

vipCustID	your unique VIP ID
vipTimestamp	a date/time stamp in the following format: **Wed, 1 Dec 2010 13:01:00 GMT** Note: the seconds must be set to zero. Also, your web application server's time must be within 10 minutes of VIP's server time (in GMT). Our server's time is updated regularly with an official time keeper. Time is formatted using 24 hours in a day ("military time").
vipSignature	SHA-256 encrypted signature string computed from: vipTimestamp + your secret code + URL GET/POST string + vipCustID

- Do not include a leading "?" in the URL querystring.
- Do not include " + ", each piece should be placed next to each other. (Note: the actual HTTP attributes get converted into "HTTP_vipCustID". Your application server may not automatically prepend "HTTP_" to each request. If it does not, you must include it by hand.)
- Timestamp must be in GMT format with zeros for seconds.
- Do not include a leading zero in the timestamp day number.
- Timestamp hours, minutes and seconds must each be 2 digits; add a leading zero if required.
- Timestamp hours use 24 hour format ("military time").

See the code samples at the end of this document for more information.

Request Details - brands and stores

Data rarely changes. We strongly suggest you cache this output on your own webserver. Update the cache once a day. This will ensure optimal performance for the end user.

GET or POST input parameters:

action	"brands"
category[1-12]	[optional] 1 or more category values to filter on; comma list
type	[optional] 1 package type or size to filter on (defined at customer level)
pkgtype	[optional] 1 or more package types to filter on; comma list
pkgsize	[optional] 1 or more package sizes to filter on; comma list

Sample XML Output:

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
    <custID>VIP</custID>
    <input>
       <territory>0</territory>
        <action>brands</action>
        <brand></brand>
        <category1></category1>
        <category2></category2>
        <category3></category3>
        <category4></category4>
        <category5></category5>
        <category6></category6>
        <category7></category7>
        <category8></category8>
        <category9></category9>
        <category10></category10>
        <category11></category11>
        <category12></category12>
        <packageTypeSize></packageTypeSize>
        <packageType></packageType>
        <packageSize></packageSize>
    </input>
    <timestamp>Mon, 12 Sep 2009 13:01:00 GMT</timestamp>
    <brands>
        <brand>BRAND 1</brand>
        <brand>BRAND 2</brand>
        ...
    </brands>
    <locations>
        <location code="01">Location XYZ</location>
        <location code="02">Location 123</location>
        ...
    </locations>
</result>
```

Request Details - categories

Category values are custom and defined between VIP and the SRS customer. Data rarely changes. We strongly suggest you cache this output on your own webserver.

GET or POST input parameters:

action	"category1" or "category2" … through "category6"
category[1-12]	[optional] 1 or more category values to filter on; comma list
type	[optional] 1 package type/size to filter on
pkgtype	[optional] 1 or more package types to filter on; comma list
pkgsize	[optional] 1 or more package sizes to filter on; comma list
brand	[optional] 1 or more brand codes to filter on; comma list

Sample XML Output:

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
    <custID>VIP</custID>
    <input>
        <territory>0</territory>
        <action>category3</action>
        <brand></brand>
        <category1></category1>
        <category2></category2>
        <category3></category3>
        <category4></category4>
        <category5></category5>
        <category6></category6>
        <category7></category7>
        <category8></category8>
        <category9></category9>
        <category10></category10>
        <category11></category11>
        <category12></category12>
        <packageTypeSize></packageTypeSize>
        <packageType></packageType>
        <packageSize></packageSize>
    </input>
    <timestamp>Mon, 16 Apr 2012 13:01:00 GMT</timestamp>
    <categories id="3">
         <category>Beer</category>
         <category>Wine</category>
         ...
    </categories>
</result>
```

Request Details - locations

Locations are returned 20 at a time based on the "page" parameter.
Brand and storeType accept multiple codes separated by comma. This list must then be URL Encoded (for instance, “,” becomes “%2C” while the space character becomes “%20”).

GET or POST input parameters:

action	"results"
brand	[optional] 1 or more brands to search for; comma list
category[1-12]	[optional] 1 or more category values to search for There are 12 possible categories (viewable within the VIP Site -> Item Master). For each category, you can search for 1+ values. The typical category definitions are this: ● *Category 1 = VIP Product Class* ● *Category 2 = VIP Product Brand* ● *Category 3 = VIP Product Supplier* ● *Category 4 = VIP Product User Alpha 1* ● *Category 5 = VIP Product User Alpha 2* ● *Category 6 = VIP Product User Alpha 3* Example #1: “&category1=craft,domestic” *Returns locations that have sold craft OR domestic* Example #2: “&category1=import&category4=lager” *Returns locations that have sold both import AND lager*
type	[optional] 1 package type/size to filter on
pkgtype	[optional] 1 or more package types to filter on; comma list
pkgsize	[optional] 1 or more package sizes to filter on; comma list
zip	5-digit zip code *(if using lat/long it’s recommended to also input zip)*
lat	[optional] latitude *(required if you don’t specify zip)*
long	[optional] longitude *(required if you don’t specify zip)*
miles	[optional] # of miles to search, 1 to 100; defaults to 10.
storeType	[optional] “on”|”off”|locationCode(s). Location codes <10 must start with zero.
page	[optional] page number; 0 is the first page; can be up to 19.
pagesize	[optional] defaults to 20; can be up to 100.

Sample XML Output:

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
    <custID>VIP</custID>
    <input>
        <action>results</action>
         <territory></territory>
         <brand></brand>
         <category1></category1>
         <category2></category2>
         <category3></category3>
         <category4></category4>
         <category5></category5>
         <category6></category6>
         <category7></category7>
         <category8></category8>
         <category9></category9>
         <category10></category10>
         <category11></category11>
         <category12></category12>
        <storeType></storeType>
        <packageTypeSize></packageTypeSize>
        <packageType></packageType>
        <packageSize></packageSize>
        <miles>10</miles>
        <page>0</page>
        <pagesize>20</pagesize>
        <lat></lat>
        <long></long>
        <zip>05446</zip>
    </input>
    <timestamp>Wed, 1 Dec 2015 13:01:00 GMT</timestamp>
    <page>1</page>
    <start>1</start>
    <end>20</end>
    <total>34</total>
    <errMsg></errMsg>
    <locations>
        <location num="1">
            <dba>VIP Sample Store</dba>
            <street>402 Watertower Circle</street>
            <city>Colchester</city>
            <state>Vermont</state>
            <zip>05446</zip>
             <lat>40.12345</lat>
            <long>-74.12345</long>
            <phone>8026559400</phone>
            <storeType>01</storeType> <!-- see brands and stores service for descriptions -->
            <distance>1.2</distance>
             <lastSold>20150701</lastSold> <!-- YYYYMMDD format -->
            <otherBrands>
                <otherBrand>BRAND 1</otherBrand>
                <otherBrand>BRAND 2</otherBrand>
                ...
            </otherBrands>
            <packages>
                <package>PACKAGE 1</package>
                <package>PACKAGE 2</package>
                ...
            </packages>
           <packageTypes>
                <packageType>Bottle</packageType>
                <packageType>Keg</packageType>
                ...
            </packageTypes>
           <packageSizes>
                <packageType>6-pack</packageSize>
                <packageType>Loose</packageSize>
                ...
            </packageSizes>
        </location>
        ...
    </locations>
</result>
```

HTTP Request Testing

VIP offers a way to test your Web Service request. This will help you verify the data you're sending to VIP in addition to how VIP interprets it. Change your "URL to query" and your response content type from XML to HTML. Here's the test URL:

https://www.vtinfo.com/PF/product_finder-test.asp

Please confirm each POST or GET variable is correct. Then ensure all 3 HTTP Headers are correct (HTTP_vipCustID, HTTP_vipTimestamp and HTTP_vipSignature).

Encryption Method

SHA-256. A highly-secure, industry-standard, HASH encoder method. The resulting string should be **64** characters long and all **lowercase**.

Learn more on the SHA Wikipedia page.

Sample Code - Classic ASP

```
dim objRequest,strCmd,strResponse,timestamp
// 1. build the request
[code to connect to your IBM DB2 database; or equivalent]
strSQL = "select (CURRENT_TIMESTAMP - CURRENT_TIMEZONE) as GMTTIME from SYSIBM/SYSDUMMY1"
[execute this SQL statement and place the resulting string into ASP variable "timestamp"]
timestamp = left(GetDayName(weekday(timestamp)),3) &  ", "& day(timestamp) & " "&
left(MonthName(month(timestamp)),3) & " "& year(timestamp) & " "& hour(timestamp) &”:”& minute(timestamp) & ":00
GMT"

// 2. send the request
Set objRequest = Server.CreateObject( "Winhttp.WinhttpRequest.5.1" )   strCmd =
"https://www.vtinfo.com/PF/product_finder-service.asp?action=brands"
objRequest.Open  "GET", strCmd,  false
objRequest.setRequestHeader  "vipCustID", request("CustID")        'comes through pre-pended with "HTTP_"
objRequest.setRequestHeader  "vipSignature", sha256(timestamp &  "__yourSecretCode__"  & "action=brands"  &
YourCustID)
objRequest.setRequestHeader  "vipTimestamp", timestamp
objRequest.Send

// 3. receive response back from VIP server & parse
If Err.Number <> 0 Then
     response.write( "error: "& err.linenumber & ") "& err.description)
end if
strResponse = objRequest.responseText
response.contentType =  "text/xml"
response.write(strResponse)
```

Sample Code – ASP.NET C#

```
//first the ASPX page…
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"  Inherits="BrandFinderAPI._Default"
%>

//then the Code-Behind…
using System;
using System.Text;
using System.Web;
using System.Web.UI;
using System.Net;
using System.Security.Cryptography;

namespace BrandFinderAPI
{
        public partial class _Default : System.Web.UI. Page
        {
                protected void Page_Load(object sender, EventArgs e)
                {
                        doSHA256Hash();
                }
                protected void doSHA256Hash()
                {
                        // 1. build the request
                        string timestamp = String.Format("{0} GMT", DateTime.UtcNow.ToString( "ddd, d MMM yyyy
HH:mm:00")); // Wed, 1 Dec 2010 12:21:00 GMT
                        string key = "__yourSecretKeyGoesHere__" ;
                        string querystring = "action=brands" ;
                        string custId = "__yourCustomerID__" ;
                        string signature = timestamp + key + querystring + custId;
                        System.Text. ASCIIEncoding encoding = new System.Text. ASCIIEncoding ();
                        SHA256 sha = new SHA256Managed ();
                        Byte[] hash = sha.ComputeHash(encoding.GetBytes(signature));
                        string encrypted = toHexString(hash).ToLower();

                        // 2. send the request
                        HttpWebRequest request =
        (HttpWebRequest )WebRequest .Create("https://www.vtinfo.com/PF/product_finder-service.asp?action=brands
        ");
                        request.Headers.Add( "vipCustID", custId);
                        request.Headers.Add( "vipTimestamp" , timestamp);
                        request.Headers.Add( "vipSignature" , encrypted);
                        request.Timeout = 30 * 1000;
                        request.Method = "GET";

                        // 3. receive response back from VIP server & parse
                        HttpWebResponse response = ( HttpWebResponse )request.GetResponse();
                        System.IO. Stream receiveStream = response.GetResponseStream();
                        System.IO. StreamReader readStream = new System.IO. StreamReader (receiveStream,
encoding);
                        string xml = readStream.ReadToEnd();
                        response.Close();
                        readStream.Close();
                }
                public static string toHexString( String value)
                {
                        StringBuilder sb = new StringBuilder ();
                        using (SHA256 hash = SHA256Managed.Create())
                        {
                                Encoding enc = Encoding.UTF8;
                                Byte[] result = hash.ComputeHash(enc.GetBytes(value));
                                        foreach (Byte b in result)
                                                        Sb.Append(b.ToString("x2"));
                        }
                        return sb.ToString();
                }
        }
}
```

Sample Code - PHP

```
$custID = ' place customer ID here ';
$secret = ' place secret code here ';
$parms = 'action=brands';

date_default_timezone_set('GMT');
$stamp = date("D, j M Y H:i:00 T", time());
$url = "https://www.vtinfo.com/PF/product_finder-service.asp";
$sigString = $stamp . $secret . $parms . $custID;
$sigHash = hash('sha256',$sigString);

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL,$url .'?'. $parms);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        'vipCustID: '. $custID
        ,'vipTimestamp: '. $stamp
        ,'vipSignature: '. $sigHash
));
curl_setopt($ch, CURLOPT_POSTFIELDS, array(
    'action' => 'brands'
));
$result = curl_exec($ch);
```

Sample Code – Flash Actionscript

Note: Incomplete, for reference only. Flash requires POST instead of GET. Flash security requires appropriate *crossdomain.xml* document to exist on the VIP website – contact VIP with the domains you need access from. Flash also doesn't ensure each hour, minute and second pair include a leading zero, so please manually ensure one exists.

```
var pvars:URLVariables = new URLVariables();
pvars.action = "results";
var sig:String = Encryption.encrypt( stamp + secret + pvars.toString() + cid, Encryption.CRYPTO_SHA256 );
var request:URLRequest = new URLRequest();
request.url = "https://www.vtinfo.com/PF/product_finder-service.asp";
request.requestHeaders.push(new URLRequestHeader("vipCustID", cid));
request.requestHeaders.push(new URLRequestHeader("vipTimestamp", stamp));
request.requestHeaders.push(new URLRequestHeader("vipSignature", sig));
request.method = URLRequestMethod.POST;
request.data = pvars;
```

Sample Code – Ruby on Rails

Special thanks to BlenderBox.com for providing this. Updates found here: https://gist.github.com/2003247

```
class ProductFinder < ActiveRecord::Base

  class << self

    def search(miles, zip, brand=nil, storeType=nil)
      now = DateTime.now.utc
      d = now.strftime("%d")
      params = "action=results&zip=#{zip}&miles=#{miles}"
      params += "&brand=#{brand}" unless brand.blank?
      params += "&storeType=#{storeType}" unless storeType.blank?
      timestamp = now.strftime("%a, ") + (d.index("0") == 0 ? d.slice!(1) : d) + now.strftime(" %b %Y
%H:%M:00 GMT")
      signature = timestamp + <brand_finder_secret> + params + <brand_finder_supplier_id>
      encrypted = Digest::SHA256.hexdigest(signature)
      url = "<brand_finder_url>?#{params}"
      #puts "Get information from #{url}"

      response = nil
      uri = URI(url)
      Net::HTTP.start(uri.host, uri.port) do |http|
        request = Net::HTTP::Get.new uri.request_uri

        request['vipCustID'] = Settings.beer_finder.supplier_id
        request['vipTimestamp'] = timestamp
        request['vipSignature'] = encrypted

        response = http.request request # Net::HTTPResponse object
        return { :errMsg => "Cannot connect to service" } unless response.is_a?(Net::HTTPSuccess)
        # Headers
        logger.info "Headers: #{request.to_hash.inspect}"

        # Status
        logger.info response.code       # => '200'
        logger.info response.message    # => 'OK'
      end

      # Open up the XML
      doc = Nokogiri::XML(response.body)

      locs = []
      doc.xpath("//locations/location").each do |l|
        location = {
          :num => l.attribute("num").text.to_i,
          :dba => l.xpath("dba").text,
          :street => l.xpath("street").text,
          :city => l.xpath("city").text,
          :state => l.xpath("state").text,
          :zip => l.xpath("zip").text,
          :lat => l.xpath("lat").text,
          :long => l.xpath("long").text,
          :phone => l.xpath("phone").text,
          :storeType => l.xpath("storeType").text,
          :distance => l.xpath("distance").text,
        }
      end
    end
  end
end
```

Sample Code – Node.js

Special thanks to Jason Hummel for providing this. Updates found here:
https://gist.github.com/jhummel/98639dd9ec70f5a2e0df

```
var moment = require('moment');
var http   = require('http');
var crypto = require('crypto');

function makeRequest(callback) {
  var secret    = '<SHARED SECRET>'
  var id        = '<CLIENT ID>'

  var params    = 'action=brands'

  var timestamp = moment.utc().format("ddd, D MMM YYYY HH:mm:00 [GMT]");
  var signature = timestamp + secret + params + id

  var sha       = crypto.createHash('sha256')
  var encrypt   = sha.update(signature).digest('hex')

  return http.request({
    host: 'www.vtinfo.com',
    path: '/PF/product_finder-service.asp?' + params,
    headers: {
      'vipCustID': id,
      'vipTimestamp': timestamp,
      'vipSignature': encrypt
    }
  }, function(response){
    var body = ''

    response.on('data', function(d) {
      body += d;
    });

    response.on('end', function() {
      console.log(body);
    });
  });
}

var req = makeRequest();

req.on('error', function(e){
  console.log( "ERROR:" + e.message);
});

req.end();
```