

FraudLabs™ Web Service

~~ Preventing Online Frauds ~~

1. Overview	3
1.1 Overview of the FraudLabs Web Service	4
1.2 Front End of FraudLabs Web Service	5
1.2.1. Integration with In-house System	5
1.3 Back End of FraudLabs Web Service	7
1.3.1. Fraud Score Calculation	7
1.3.1.1. Formula	7
1.3.1.2. Example	8
1.4 Process Flow	9
1.4.1. Overview	9
1.4.2. Fraud Score Recommendation	9
2. Implementation and WSDL	10
2.1 Basic Parameters - Input	11
2.2 Basic Parameters - Output	12
2.3 List of possible value for MESSAGE field	13
3. Design Information	14
3.1 Placement of FraudLabs Web Service	14
Appendix I : ISO3166 Country Code	15
Appendix II : Sample Code	21

1. Overview

This documentation provides a basic understanding and information to help you get started with our products. Look over this documentation to gain a high-level understanding of the process flow that underlies the FraudLabs Web Services.

For more information, please visit <http://www.fraudlabs.com> or contact your FraudLabs representative:

Email: sales@fraudlabs.com

1.1 Overview of the FraudLabs Web Service

FraudLabs Web Service is the proprietary credit card fraud detection service that integrated with our IP2Location™ technology (geolocation service provider) to reduce fraud for online merchants. It screens and detects online credit card fraud where every transaction goes through a strict assessment process which reviews over a dozen aspects of online purchase parameters to determine high risk orders, such as IP address, email address and billing address and returns fraud analysis results together with fraud score in real-time. Through our analysis, we have been able to identify traits and patterns that are associated with fraudulent orders, before the payment is processed. Merchants can make use of our service in order to speed up manual order verification, automate order process according to the fraud score that we provide.

FraudLabs Web Service is hosted on redundant servers, redundant Internet connections, and 24x7x365 monitoring. If we detect suspicious activity from IP address, it will be flagged as high risk throughout the network in real-time.

FraudLabs Web Service is using platform independent XML format to exchange data between systems. All customers can integrate our web service regardless of their web server and shopping cart solutions.

Key Features Include:

- Pinpoints the exact location of customers, with country, state, city, latitude and longitude and Internet Service Providers using IP address
- Calculate distance between delivery city and estimated city from IP address
- Verify whether the online buyer is behind anonymous proxy servers
- Verify whether the online buyer is using free anonymous email address
- Verify whether the online buyer is using a known mail drop or postal box
- Match between credit card Bank Identification Number (BIN) data and user-entered information
- Match between area code and postal code information in delivery address
- Provides a complete XML-based Web Services API
- Contains sample code examples for ease integration

1.2 Front End of FraudLabs Web Service

The general idea is that front end is responsible for collection input from the user and conforms to some specification that the back end can use. Front end of FraudLabs Web Service is rather simple to understand. As we are using platform independent XML format to exchange data between systems, you may either integrate our web service to your in-house system, or direct access to our hosted web service.

1.2.1. Integration with In-house System

- I. Log on to <http://www.fraudlabs.com/samplecodes.aspx> and download sample codes that you need (For sample codes in different languages please refer to Appendix II). Below are links to get sample codes:
 - i. Microsoft ASP.NET (v1.1) - VB.NET:
<http://www.fraudlabs.com/samplecode/fraudlabswebserviceclientvb.zip>
 - ii. Microsoft ASP.Net (v1.1) - C#:
<http://www.fraudlabs.com/samplecode/fraudlabswebserviceclientcsharp.zip>
 - iii. Microsoft ASP.NET (v2.0) - VB.NET:
<http://www.fraudlabs.com/samplecode/fraudlabswebserviceclientvb2005.zip>
 - iv. Microsoft ASP.Net (v2.0) - C#:
<http://www.fraudlabs.com/samplecode/fraudlabswebserviceclientcsharp2005.zip>
 - v. Java:
<http://www.fraudlabs.com/samplecode/fraudlabswebserviceclientjava.zip>
 - vi. PHP:
<http://www.fraudlabs.com/samplecode/fraudlabswebserviceclientphp.zip>
 - vii. Python:
<http://www.fraudlabs.com/samplecode/fraudlabswebserviceclientpython.zip>
 - viii. Perl:
<http://www.fraudlabs.com/samplecode/fraudlabswebserviceclientperl.zip>
 - ix. ColdFusion 9:
<http://www.fraudlabs.com/samplecode/fraudlabswebserviceclientcoldfusion.zip>

- II. Please go through 'readme' file that we provide together with the sample codes for more set up information
- III. In order to use our service, you need to get your own license key – *How?*
 - i. log on to <http://www.fraudlabs.com>
 - ii. sign up as our registered member
 - iii. check your email to complete user activation
 - iv. get license key :
 - a) Free License Account:
 - 1) at the left menu bar, under category of FraudLabs™ Fraud Detection, click "free license "
 - 2) view Terms of Use (*please note that you must agree with our Terms of Use before proceed)
 - 3) click on "Get Free License Now"
 - 4) the license key will be sent to your email
 - b) Premium Subscription:
 - 1) at the left menu bar, under category of FraudLabs™ Fraud Detection, click "subscribe now"
 - 2) fill in required field in the secure payment form
 - 3) click on "make payment"
 - 4) the license key will be sent to your email after your payment is confirmed by our online payment merchant
- IV. Now fill in the required field
- V. Click on "Submit" and result is provided

1.3 Back End of FraudLabs Web Service

Back end is the part that processes the input from the front end. The process is seamless to the end-user and works by interaction with a SOAP API through FraudLabs Web Service.

The process works as follows:

1. User submits a form containing the parameters (inputs) that to be analyzed
2. System verify the license key before proceed
3. If the license key is valid, it will proceed and check the credits availability. If the remaining credits still available, it will start to analyze the parameters
4. The results then will be submitted for fraud score calculation process
5. Returns and display the results

1.3.1. Fraud Score Calculation

Based on the parameters and known risk factors from user, we calculate fraud score using our own algorithm. The fraud validation score is directly proportional to the risk of the input values. The higher the scores, the higher the risk of fraudulent transactions.

1.3.1.1. Formula

Risk Field	Weight
IP Validation	2
IP – Country Match	10
Anonymous Proxy	20
High Risk Country	10
Bin – Country Match	5
Bin – Name Match	5
Bin – Phone Match	5
Email Domain (Free Email)	15
Phone City	5
Postal City	5
Ship Forward	10
Distance > 400Km	2.5
> 1,000Km	5
> 5,000Km	10

Fraud Score = (Total Possible Weight / Total Weight) * 100%
--

1.3.1.2. Example

i. Total weight refers to the sum of the scores that we issued to every result field as long as the field is filled up by the user.

ii. Like for example, assume that we have 3 fields for this fraud checking, IP validation, IP-country matching and anonymous proxy checking. We issue 2 points for IP validation field, 10 points for country matching and 20 points for anonymous proxy checking. So the equation of the total weight will be:

$$\begin{aligned} \text{Total Weight} &= 2 + 10 + 20 \\ &= 32 \end{aligned}$$

iii. Total possible weight refers to the sum of the scores that will be added up only when the parameter of certain field meets certain criteria.

iv. Like for example, assume that the IP address is valid, but the country is mismatch and the request comes from anonymous proxy server. So we won't add 2 points into existing fraud score, but we will add 10 points for country mismatch and 20 points for anonymous proxy. The equation for the total possible weight is:

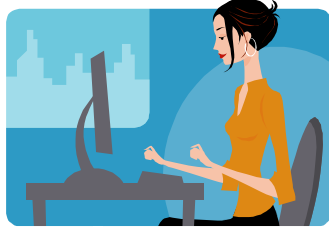
$$\begin{aligned} \text{Total Possible Weight} &= 0 + 10 + 20 \\ &= 30 \end{aligned}$$

v. **Fraud Score = (Total Possible Weight / Total Weight) * 100%.** So according to this equation, the grand fraud score is:

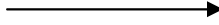
$$\begin{aligned} \text{Fraud Score} &= (30 / 32) * 100\% \\ &= 93.75 \end{aligned}$$

1.4 Process Flow

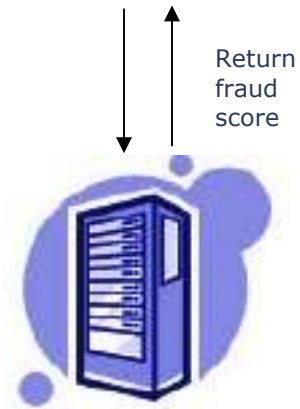
1.4.1. Overview



Step 1
Customer inputs from
online purchase



Step 2
Processing work done
by merchant



Step 3
FraudLabs Web Service
(Fraud score calculation)

1.4.2. Fraud Score Recommendation

Color	Fraud Score Range	Description
Green	≤ 30	Low Risk of Fraudulent
Yellow	31 - 70	Moderate Risk of Fraudulent
Red	> 70	High Risk of Fraudulent

2. Implementation and WSDL

This section provides basic information of the process of integrating the web service into your website. Look over this section to gain a high-level understanding of requesting FraudLabs Web Service.

A WSDL is available at:

<http://v1.fraudlabs.com/fraudlabswebservice.asmx?wsdl>

For more information about FraudLabs Web Service implementation, please visit <http://www.fraudlabs.com> or contact your FraudLabs representative:

Email: sales@fraudlabs.com

2.1 Basic Parameters - Input

Field	Format	Description
IP	Required	IP address of credit card holder.
CITY	Optional	City of billing address.
REGION	Optional	Region of billing address.
COUNTRY	Optional	Country code of billing address.
POSTAL	Optional	Postal/Zip code of billing address.
DOMAIN	Optional	Email address domain name.
PHONE	Optional	Customer phone number.
BIN	Optional	First 6 digits of credit card number to identify issuing bank.
BINNAME	Optional	Name of the bank which issued the credit card.
BINPHONE	Optional	Customer service phone number listed on back of credit card.
LICENSE	Required	License key for free account or premium account.
SHIPADDR	Optional	Address of shipping address if different than billing address.
SHIPCITY	Optional	City of shipping address if different than billing address.
SHIPREGION	Optional	Region of shipping address if different than billing address.
SHIPPOSTAL	Optional	Postal/Zip code of shipping address if different than billing address.
SHIPCOUNTRY	Optional	Country of shipping address if different than billing address.
QUERYID	Optional	Additional information to identify this transaction.

2.2 Basic Parameters - Output

Field	Format	Description
COUNTRYMATCH	YES or NO	Whether country of IP address matches billing address country.
COUNTRY	Char(2)	Two-letter ISO-3166 country code.
HIGHRISKCOUNTRY	YES or NO	Whether IP address or billing address country is in Belarus, Colombia, Egypt, Indonesia, Lebanon, Macedonia, Nigeria, Pakistan, Ukraine, Vietnam or Yugoslavia.
DISTANCE	Integer	Distance from IP address to Billing Location.
IP2COUNTRY	String	Estimated Country of the IP address.
IP2REGION	String	Estimated Region of the IP address.
IP2CITY	String	Estimated City of the IP address.
IP2LATITUDE	Float	Estimated Latitude of the IP address.
IP2LONGITUDE	Float	Estimated Longitude of the IP address.
IP2ISP	String	Estimated ISP of the IP address.
ANONYMOUSPROXY	YES or NO	Whether IP address is from Anonymous Proxy Server.
FREEMAIL	YES or NO	Whether e-mail is from free e-mail provider.
BINCOUNTRYMATCH	YES or NO	Whether country of issuing bank based on BIN number matches billing address country
BINNAMEMATCH	YES or NO	Whether name of issuing bank matches inputted BIN name.
BINPHONEMATCH	YES or NO	Whether customer service phone number matches inputted BINPHONE.
BINCOUNTRY	String	No BIN information will be provided due to security reason.
BINBANKNAME	String	BIN information will be the input provided due to security reason.
BINBANKPHONE	String	BIN information will be the input provided due to security reason.
POSTALCITYMATCH	YES or NO	Whether the customer postal code is in the billing location (for US and Canada only).
PHONECITYMATCH	YES or NO	Whether the customer phone number is in the billing location (for US and Canada only).
SHIPFORWARD	YES or NO	Whether shipping address is in database of known mail drops.
CREDITSAVAILABLE	Integer	Number of queries remaining in your account, can be used to alert you when you may need to add more queries to your account.
FRAUDSCORE	Integer	Overall score between 0 and 100. 100 is the highest risk. 0 is the lowest risk.
RESPONSEID	Integer	Integer Unique ID to refer this particular response
QUERYID	Integer	Unique ID based on customer query ID
MESSAGE	String	Web Service Message Response

2.3 List of possible value for MESSAGE field

FraudLabs Web Service Error Message
• Invalid license key
• Invalid IP address
• Invalid phone number
• Invalid email address
• IP address not exist
• City not found
• Postal not exist
• Bin not exist
• No credit available
• License key was expired

3. Design Information

This section provides suggestion regarding the placing of FraudLabs Web Service into Web pages. Look over this section to get a general idea for adding this service to your website.

3.1 Placement of FraudLabs Web Service

As FraudLabs web service provides intensive fraud checking before an online order is being confirmed to take place, this service can be placed on ordering form or transaction form on your website, offering a seamless integration.

Description: Fraud checking is used when user make an online transaction.

Level of security: High

Site Type(s):

- e-Commerce Solutions
- Internet Retailers
- Online Credit Card Merchants
- e-Commerce Software Developers

Benefits:

- prevent fraudsters from completing transaction
- reduces chargeback
- able to capture user's information before ordering
- flags orders that may be fraudulent

Appendix I: ISO3166 Country Code

This table lists all valid ISO3166 two characters country codes that returns from component API query and describe the country names behind these country codes.

Country Code	Country Name
AD	ANDORRA
AE	UNITED ARAB EMIRATES
AF	AFGHANISTAN
AG	ANTIGUA AND BARBUDA
AI	ANGUILLA
AL	ALBANIA
AM	ARMENIA
AN	NETHERLANDS ANTILLES
AO	ANGOLA
AP	ASIA PACIFIC
AQ	ANTARCTICA
AR	ARGENTINA
AS	AMERICAN SAMOA
AT	AUSTRIA
AU	AUSTRALIA
AW	ARUBA
AZ	AZERBAIJAN
BA	BOSNIA AND HERZEGOWINA
BB	BARBADOS
BD	BANGLADESH
BE	BELGIUM
BF	BURKINA FASO
BG	BULGARIA
BH	BAHRAIN
BI	BURUNDI
BJ	BENIN
BM	BERMUDA
BN	BRUNEI DARUSSALAM
BO	BOLIVIA
BR	BRAZIL
BS	BAHAMAS
BT	BHUTAN
BV	BOUVET ISLAND
BW	BOTSWANA
BY	BELARUS
BZ	BELIZE
CA	CANADA
CC	COCOS (KEELING) ISLANDS
CD	CONGO, THE DEMOCRATIC REPUBLIC OF THE
CF	CENTRAL AFRICAN REPUBLIC
CG	CONGO

Country Code	Country Code
CH	SWITZERLAND
CI	COTE D'IVOIRE
CK	COOK ISLANDS
CL	CHILE
CM	CAMEROON
CN	CHINA
CO	COLOMBIA
CR	COSTA RICA
CS	CZECHOSLOVAKIA (FORMER)
CU	CUBA
CV	CAPE VERDE
CX	CHRISTMAS ISLAND
CY	CYPRUS
CZ	CZECH REPUBLIC
DE	GERMANY
DJ	DJIBOUTI
DK	DENMARK
DM	DOMINICA
DO	DOMINICAN REPUBLIC
DZ	ALGERIA
EC	ECUADOR
EE	ESTONIA
EG	EGYPT
EH	WESTERN SAHARA
ER	ERITREA
ES	SPAIN
ET	ETHIOPIA
EU	EUROPEAN UNION
FI	FINLAND
FJ	FIJI
FK	FALKLAND ISLANDS (MALVINAS)
FM	MICRONESIA, FEDERATED STATES OF
FO	FAROE ISLANDS
FR	FRANCE
FX	FRANCE, METROPOLITAN
GA	GABON
GB	GREAT BRITAIN
GD	GRENADA
GE	GEORGIA
GF	FRENCH GUIANA
GH	GHANA
GI	GIBRALTAR
GL	GREENLAND
GM	GAMBIA
GN	GUINEA
GP	GADELOUPE
GQ	EQUATORIAL GUINEA
GR	GREECE

Country Code	Country Code
GS	SOUTH GEORGIA & SOUTH SANDWICH ISLANDS
GT	GUATEMALA
GU	GUAM
GW	GUINEA-BISSAU
GY	GUYANA
HK	HONG KONG
HM	HEARD ISLAND AND MCDONALD ISLANDS
HN	HONDURAS
HR	CROATIA
HT	HAITI
HU	HUNGARY
ID	INDONESIA
IE	IRELAND
IL	ISRAEL
IN	INDIA
IO	BRITISH INDIAN OCEAN TERRITORY
IQ	IRAQ
IR	IRAN, ISLAMIC REPUBLIC OF
IS	ICELAND
IT	ITALY
JM	JAMAICA
JO	JORDAN
JP	JAPAN
KE	KENYA
KG	KYRGYZSTAN
KH	CAMBODIA
KI	KIRIBATI
KM	COMOROS
KN	SAINT KITTS AND NEVIS
KP	KOREA, DEMOCRATIC PEOPLE'S REPUBLIC OF
KR	KOREA, REPUBLIC OF
KW	KUWAIT
KY	CAYMAN ISLANDS
KZ	KAZAKSTAN
LA	LAO PEOPLE'S DEMOCRATIC REPUBLIC
LB	LEBANON
LC	SAINT LUCIA
LI	LIECHTENSTEIN
LK	SRI LANKA
LR	LIBERIA
LS	LESOTHO
LT	LITHUANIA
LU	LUXEMBOURG
LV	LATVIA
LY	LIBYAN ARAB JAMAHIRIYA
MA	MOROCCO
MC	MONACO
MD	MOLDOVA, REPUBLIC OF

Country Code	Country Code
MG	MADAGASCAR
MH	MARSHALL ISLANDS
MK	MACEDONIA, THE FORMER YUGOSLAV
ML	MALI
MM	MYANMAR
MN	MONGOLIA
MO	MACAU
MP	NORTHERN MARIANA ISLANDS
MQ	MARTINIQUE
MR	MAURITANIA
MS	MONTSERRAT
MT	MALTA
MU	MAURITIUS
MV	MALDIVES
MW	MALAWI
MX	MEXICO
MY	MALAYSIA
MZ	MOZAMBIQUE
NA	NAMIBIA
NC	NEW CALEDONIA
NE	NIGER
NF	NORFOLK ISLAND
NG	NIGERIA
NI	NICARAGUA
NL	NETHERLANDS
NO	NORWAY
NP	NEPAL
NR	NAURU
NU	NIUE
NZ	NEW ZEALAND
OM	OMAN
PA	PANAMA
PE	PERU
PF	FRENCH POLYNESIA
PG	PAPUA NEW GUINEA
PH	PHILIPPINES
PK	PAKISTAN
PL	POLAND
PM	SAINT PIERRE AND MIQUELON
PN	PITCAIRN
PR	PUERTO RICO
PS	PALESTINIAN TERRITORY, OCCUPIED
PT	PORTUGAL
PW	PALAU
PY	PARAGUAY
QA	QATAR
RE	REUNION
RO	ROMANIA

Country Code	Country Code
RU	RUSSIAN FEDERATION
RW	RWANDA
SA	SAUDI ARABIA
SB	SOLOMON ISLANDS
SC	SEYCHELLES
SD	SUDAN
SE	SWEDEN
SG	SINGAPORE
SH	SAINT HELENA
SI	SLOVENIA
SJ	SVALBARD AND JAN MAYEN
SK	SLOVAKIA
SL	SIERRA LEONE
SM	SAN MARINO
SN	SENEGAL
SO	SOMALIA
SR	SURINAME
ST	SAO TOME AND PRINCIPE
SU	RUSSIAN FEDERATION
SV	EL SALVADOR
SY	SYRIAN ARAB REPUBLIC
SZ	SWAZILAND
TC	TURKS AND CAICOS ISLANDS
TD	CHAD
TF	FRENCH SOUTHERN TERRITORIES
TG	TOGO
TH	THAILAND
TJ	TAJIKISTAN
TK	TOKELAU
TM	TURKMENISTAN
TN	TUNISIA
TO	TONGA
TP	EAST TIMOR
TR	TURKEY
TT	TRINIDAD AND TOBAGO
TV	TUVALU
TW	TAIWAN, PROVINCE OF CHINA
TZ	TANZANIA, UNITED REPUBLIC OF
UA	UKRAINE
UG	UGANDA
UK	UNITED KINGDOM
UM	UNITED STATES MINOR OUTLYING ISLANDS
US	UNITED STATES
UY	URUGUAY
UZ	UZBEKISTAN
VA	HOLY SEE (VATICAN CITY STATE)
VC	SAINT VINCENT AND THE GRENADINES
VE	VENEZUELA

Country Code	Country Code
VG	VIRGIN ISLANDS, BRITISH
VI	VIRGIN ISLANDS, U.S.
VN	VIET NAM
VU	VANUATU
WF	WALLIS AND FUTUNA
WS	SAMOA
YE	YEMEN
YT	MAYOTTE
YU	YUGOSLAVIA
ZA	SOUTH AFRICA
ZM	ZAMBIA
ZW	ZIMBABWE

Appendix II: Sample Code

FraudLabs Web Service sample code is available in several different programming languages. Below are some examples, for more different programming languages please log on to:

<http://www.fraudlabs.com/samplecodes.aspx>

i. ASP.NET – VB.NET (SOAP)

```
Private Sub FraudLabsWebService()  
    Dim x_FraudLabs As New FraudLabsWebService.FraudLabsWebService  
    Dim oFraudLabs As New FraudLabsWebService.FraudLabsOutput  
    Dim iFraudLabs As New FraudLabsWebService.FraudLabsInput  
  
    Try  
        iFraudLabs.IP = Me.txtIP.Text  
        iFraudLabs.CITY = Me.txtCity.Text  
        iFraudLabs.REGION = Me.txtRegion.Text  
        iFraudLabs.POSTAL = Me.txtPostal.Text  
        iFraudLabs.COUNTRY = Me.txtCountry.Text  
        iFraudLabs.DOMAIN = Me.txtDomain.Text  
        iFraudLabs.PHONE = Me.txtPhone.Text  
        iFraudLabs.BIN = Me.txtBIN.Text  
        iFraudLabs.BINNAME = Me.txtBINName.Text  
        iFraudLabs.BINPHONE = Me.txtBINPhone.Text  
        iFraudLabs.LICENSE = Me.txtLicense.Text  
        iFraudLabs.SHIPADDR = Me.txtShipAddr.Text  
        iFraudLabs.SHIPCITY = Me.txtShipCity.Text  
        iFraudLabs.SHIPREGION = Me.txtShipRegion.Text  
        iFraudLabs.SHIPPOSTAL = Me.txtShipPostal.Text  
        iFraudLabs.SHIPCOUNTRY = Me.txtShipCountry.Text  
        iFraudLabs.QUERYID = Me.txtQueryID.Text  
  
        oFraudLabs = x_FraudLabs.FraudLabs(iFraudLabs)  
  
        Me.txtResult.Text = "Anonymous Proxy:" &  
oFraudLabs.ANONYMOUSPROXY & vbNewLine  
        Me.txtResult.Text += "BIN Bank Name:" & oFraudLabs.BINBANKNAME &  
vbNewLine  
        Me.txtResult.Text += "BIN Bank Phone:" & oFraudLabs.BINBANKPHONE  
& vbNewLine  
        Me.txtResult.Text += "BIN Country:" & oFraudLabs.BINCOUNTRY &  
vbNewLine  
        Me.txtResult.Text += "BIN Country Match:" &  
oFraudLabs.BINCOUNTRYMATCH & vbNewLine  
        Me.txtResult.Text += "BIN Name Match:" & oFraudLabs.BINNAMEMATCH  
& vbNewLine  
        Me.txtResult.Text += "BIN Phone Match:" &  
oFraudLabs.BINPHONEMATCH & vbNewLine  
        Me.txtResult.Text += "Country:" & oFraudLabs.COUNTRY & vbNewLine  
        Me.txtResult.Text += "Country Match:" & oFraudLabs.COUNTRYMATCH &  
vbNewLine  
        Me.txtResult.Text += "Credits Available:" &  
oFraudLabs.CREDITSAVAILABLE & vbNewLine
```

```

        Me.txtResult.Text += "Distance:" & oFraudLabs.DISTANCE & vbNewLine
        Me.txtResult.Text += "Fraud Score:" & oFraudLabs.FRAUDSCORE &
vbNewLine
        Me.txtResult.Text += "Free Email:" & oFraudLabs.FREEMAIL &
vbNewLine
        Me.txtResult.Text += "High Risk Country:" &
oFraudLabs.HIGHRISKCOUNTRY & vbNewLine
        Me.txtResult.Text += "IP to City:" & oFraudLabs.IP2CITY &
vbNewLine
        Me.txtResult.Text += "IP to Country:" & oFraudLabs.IP2COUNTRY &
vbNewLine
        Me.txtResult.Text += "IP to ISP:" & oFraudLabs.IP2ISP & vbNewLine
        Me.txtResult.Text += "IP to Latitude:" & oFraudLabs.IP2LATITUDE &
vbNewLine
        Me.txtResult.Text += "IP to Longitude:" & oFraudLabs.IP2LONGITUDE
& vbNewLine
        Me.txtResult.Text += "IP to Region:" & oFraudLabs.IP2REGION &
vbNewLine
        Me.txtResult.Text += "Message:" & oFraudLabs.MESSAGE & vbNewLine
        Me.txtResult.Text += "Phone City Match:" &
oFraudLabs.PHONECITYMATCH & vbNewLine
        Me.txtResult.Text += "Postal City Match:" &
oFraudLabs.POSTALCITYMATCH & vbNewLine
        Me.txtResult.Text += "Query ID:" & oFraudLabs.QUERYID & vbNewLine
        Me.txtResult.Text += "Ship Forward:" & oFraudLabs.SHIPFORWARD &
vbNewLine
    Catch ex As Exception
        Response.Write(ex.Message)
    End Try
End Sub

```

ii. ASP.NET – C#.NET (SOAP)

```

private void FraudLabsWebService()
{
    FraudLabsWebService x_FraudLabs = new FraudLabsWebService();
    FraudLabsOutput oFraudLabs = new FraudLabsOutput();
    FraudLabsInput iFraudLabs = new FraudLabsInput();
    try
    {
        iFraudLabs.IP = this.txtIP.Text;
        iFraudLabs.CITY = this.txtCity.Text;
        iFraudLabs.REGION = this.txtRegion.Text;
        iFraudLabs.POSTAL = this.txtPostal.Text;
        iFraudLabs.COUNTRY = this.txtCountry.Text;
        iFraudLabs.DOMAIN = this.txtDomain.Text;
        iFraudLabs.PHONE = this.txtPhone.Text;
        iFraudLabs.BIN = this.txtBIN.Text;
        iFraudLabs.BINNAME = this.txtBINName.Text;
        iFraudLabs.BINPHONE = this.txtBINPhone.Text;
        iFraudLabs.LICENSE = this.txtLicense.Text;
        iFraudLabs.SHIPADDR = this.txtShipAddr.Text;
        iFraudLabs.SHIPCITY = this.txtShipCity.Text;
        iFraudLabs.SHIPREGION = this.txtShipRegion.Text;
    }
}

```

```
        iFraudLabs.SHIPPOSTAL = this.txtShipPostal.Text;
        iFraudLabs.SHIPCOUNTRY = this.txtShipCountry.Text;
        iFraudLabs.QUERYID = this.txtQueryID.Text;
        oFraudLabs = x_FraudLabs.FraudLabs(iFraudLabs);

        this.txtResult.Text = "Anonymous Proxy:" +
oFraudLabs.ANONYMOUSPROXY + "\n";
        this.txtResult.Text += "BIN Bank Name:" +
oFraudLabs.BINBANKNAME + "\n";
        this.txtResult.Text += "BIN Bank Phone:" +
oFraudLabs.BINBANKPHONE + "\n";
        this.txtResult.Text += "BIN Country:" +
oFraudLabs.BINCOUNTRY + "\n";
        this.txtResult.Text += "BIN Country Match:" +
oFraudLabs.BINCOUNTRYMATCH + "\n";
        this.txtResult.Text += "BIN Name Match:" +
oFraudLabs.BINNAMEMATCH + "\n";
        this.txtResult.Text += "BIN Phone Match:" +
oFraudLabs.BINPHONEMATCH + "\n";
        this.txtResult.Text += "Country:" + oFraudLabs.COUNTRY +
"\n";
        this.txtResult.Text += "Country Match:" +
oFraudLabs.COUNTRYMATCH + "\n";
        this.txtResult.Text += "Credits Available:" +
oFraudLabs.CREDITSAVAILABLE + "\n";
        this.txtResult.Text += "Distance:" + oFraudLabs.DISTANCE +
"\n";
        this.txtResult.Text += "Fraud Score:" +
oFraudLabs.FRAUDSCORE + "\n";
        this.txtResult.Text += "Free Email:" + oFraudLabs.FREEMAIL
+ "\n";
        this.txtResult.Text += "High Risk Country:" +
oFraudLabs.HIGHRISKCOUNTRY + "\n";
        this.txtResult.Text += "IP to City:" + oFraudLabs.IP2CITY +
"\n";
        this.txtResult.Text += "IP to Country:" +
oFraudLabs.IP2COUNTRY + "\n";
        this.txtResult.Text += "IP to ISP:" + oFraudLabs.IP2ISP +
"\n";
        this.txtResult.Text += "IP to Latitude:" +
oFraudLabs.IP2LATITUDE + "\n";
        this.txtResult.Text += "IP to Longitude:" +
oFraudLabs.IP2LONGITUDE + "\n";
        this.txtResult.Text += "IP to Region:" +
oFraudLabs.IP2REGION + "\n";
        this.txtResult.Text += "Message:" + oFraudLabs.MESSAGE +
"\n";
        this.txtResult.Text += "Phone City Match:" +
oFraudLabs.PHONECITYMATCH + "\n";
        this.txtResult.Text += "Postal City Match:" +
oFraudLabs.POSTALCITYMATCH + "\n";
        this.txtResult.Text += "Query ID:" + oFraudLabs.QUERYID +
"\n";
        this.txtResult.Text += "Ship Forward:" +
oFraudLabs.SHIPFORWARD + "\n";
    }
    catch (Exception ex)
```

```

    {
        Response.Write(ex.Message);
    }
}

```

iii. PHP

```

<?php
if (!isset($_POST['submit'])) {} // if page is not submitted to itself
echo the form
else
{
    $ip = $_POST["ip"];
    $license = $_POST["license"];
    $city = $_POST["city"];
    $region = $_POST["region"];
    $postal = $_POST["postal"];
    $country = $_POST["country"];
    $emailedomain = $_POST["emailedomain"];
    $phone = $_POST["phone"];
    $bin = $_POST["bin"];
    $binname = $_POST["binname"];
    $binphone = $_POST["binphone"];
    $shipadd = $_POST["shipadd"];
    $shipcity = $_POST["shipcity"];
    $shipregion = $_POST["shipregion"];
    $shippostal = $_POST["shippostal"];
    $shipcountry = $_POST["shipcountry"];
    $queryid = $_POST["queryid"];

    if ($license == "<Enter License Key>" || $license == "")
    {
        echo "license key is a required field." ;
    }
    else
    {
        $wsdl = "http://v1.fraudlabs.com/fraudlabswebservice.asmx?wsdl";
        $client = new SoapClient($wsdl);
        $parms = array("IP" => $ip,"CITY" => $city,"REGION" =>
$region,"POSTAL" => $postal,"COUNTRY" => $country,"DOMAIN" =>
$emailedomain,"PHONE" => $phone,"BIN" => $bin,"BINNAME" =>
$binname,"BINPHONE" => $binphone,"LICENSE" => $license,"SHIPADDR" =>
$shipadd,"SHIPCITY" => $shipcity,"SHIPREGION" =>
$shipregion,"SHIPPOSTAL" => $shippostal,"SHIPCOUNTRY" =>
$shipcountry,"QUERYID" => $queryid);

        $result = $client->FraudLabs($parms);
        echo "COUNTRYMATCH = " . $result->COUNTRYMATCH . "<br>";
        echo "COUNTRY = " . $result->COUNTRY . "<br>";
        echo "HIGHRISKCOUNTRY = " . $result->HIGHRISKCOUNTRY . "<br>";
        echo "DISTANCE = " . $result->DISTANCE . "<br>";
        echo "IP2COUNTRY = " . $result->IP2COUNTRY . "<br>";
        echo "IP2REGION = " . $result->IP2REGION . "<br>";
        echo "IP2CITY = " . $result->IP2CITY . "<br>";
        echo "IP2LATITUDE = " . $result->IP2LATITUDE . "<br>";
    }
}

```

```

echo "IP2LONGITUDE = " . $result->IP2LONGITUDE . "<br>";
echo "IP2ISP = " . $result->IP2ISP . "<br>";
echo "ANONYMOUSPROXY = " . $result->ANONYMOUSPROXY . "<br>";
echo "FREEMAIL = " . $result->FREEMAIL . "<br>";
echo "BINCOUNTRYMATCH = " . $result->BINCOUNTRYMATCH . "<br>";
echo "BINNAMEMATCH = " . $result->BINNAMEMATCH . "<br>";
echo "BINPHONEMATCH = " . $result->BINPHONEMATCH . "<br>";
echo "BINCOUNTRY = " . $result->BINCOUNTRY . "<br>";
echo "BINBANKNAME = " . $result->BINBANKNAME . "<br>";
echo "BINBANKPHONE = " . $result->BINBANKPHONE . "<br>";
echo "POSTALCITYMATCH = " . $result->POSTALCITYMATCH . "<br>";
echo "PHONECITYMATCH = " . $result->PHONECITYMATCH . "<br>";
echo "SHIPFORWARD = " . $result->SHIPFORWARD . "<br>";
echo "CREDITSAVAILABLE = " . $result->CREDITSAVAILABLE . "<br>";
echo "FRAUDSCORE = " . $result->FRAUDSCORE . "<br>";
echo "QUERYID = " . $result->QUERYID . "<br>";
echo "MESSAGE = " . $result->MESSAGE . "<br>";
}
}
?>

```

iv. JAVA / APACHE

```

public synchronized String FraudLabs(String strIP, String strCITY,
String strREGION, String strPOSTAL, String strCOUNTRY, String
strDOMAIN, String strPHONE, String strBIN, String strBINNAME, String
strBINPHONE, String strLICENSE, String strSHIPADDR, String strSHIPCITY,
String strSHIPREGION, String strSHIPPOSTAL, String strSHIPCOUNTRY,
String strQUERYID) throws SOAPException
{
    String returnValue = "";
    if (this.url_ == null)
    {
        throw new SOAPException (Constants.FAULT_CODE_CLIENT, "A URL must
be specified through ApacheSoapProxy.setEndPoint(URL)");
    }
    // Get this from the soapAction attribute on the
    // soap:operation element that is found within the SOAP
    // binding information in the WSDL
    this.soapActionUri_ = "http://v1.fraudlabs.com/";
    ApacheMessageBody ourBody = new ApacheMessageBody ();
    // Set the argument
    ourBody.strIP = strIP;
    ourBody.strCITY = strCITY;
    ourBody.strREGION = strREGION;
    ourBody.strPOSTAL = strPOSTAL;
    ourBody.strCOUNTRY = strCOUNTRY;
    ourBody.strDOMAIN = strDOMAIN;
    ourBody.strPHONE = strPHONE;
    ourBody.strBIN = strBIN;
    ourBody.strBINNAME = strBINNAME;
    ourBody.strBINPHONE = strBINPHONE;
    ourBody.strLICENSE = strLICENSE;
    ourBody.strSHIPADDR = strSHIPADDR;
}

```

```
ourBody.strSHIPCITY = strSHIPCITY;
ourBody.strSHIPREGION = strSHIPREGION;
ourBody.strSHIPPOSTAL = strSHIPPOSTAL;
ourBody.strSHIPCOUNTRY = strSHIPCOUNTRY;
ourBody.strQUERYID = strQUERYID;
//Replace the default body with our own
this.envelope_.setBody (ourBody);
message_.send (this.getEndPoint(), this.soapActionUri_,
this.envelope_);
try
{
    //Since the Body.unmarshall() handler is static, we can't
    //replace the basic machinery easily. Instead, we must obtain
and parse the message on our own.
    this.soapMessage_ = this.message_.receive();
    XMLReader reader =
(XMLReader)Class.forName("org.apache.xerces.parsers.SAXParser").newInst
ance();
    SAXHandler handler = new SAXHandler();
    String strResult = "";
    handler.setElementToSearchFor ("COUNTRYMATCH");
    //Set the Content Handler
    reader.setContentHandler (handler);
    //Parse the file
    reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
    //If we reached here, the result has been parsed and
    //stored in the handler instance.
    strResult = "Country Match: " + handler.getResult();
    handler.setElementToSearchFor ("COUNTRY");
    reader.setContentHandler (handler);
    reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
    strResult += "\nCountry: " + handler.getResult();
    handler.setElementToSearchFor ("HIGHRISKCOUNTRY");
    reader.setContentHandler (handler);
    reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
    strResult += "\nHigh Risk Country: " + handler.getResult();
    handler.setElementToSearchFor ("DISTANCE");
    reader.setContentHandler (handler);
    reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
    strResult += "\nDistance: " + handler.getResult();
    handler.setElementToSearchFor ("IP2COUNTRY");
    reader.setContentHandler (handler);
    reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
    strResult += "\nIp2Country: " + handler.getResult();
    handler.setElementToSearchFor ("IP2REGION");
    reader.setContentHandler (handler);
    reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
    strResult += "\nIp2Region: " + handler.getResult();
    handler.setElementToSearchFor ("IP2CITY");
    reader.setContentHandler (handler);
    reader.parse ( new InputSource (new StringReader
```

```

(this.soapMessage_.getContent().toString() ));
    strResult += "\nIp2City: " + handler.getResult();
    handler.setElementToSearchFor ("IP2LATITUDE");
    reader.setContentHandler (handler);
    reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
    strResult += "\nIp2Latitude: " + handler.getResult();
    handler.setElementToSearchFor ("IP2LONGITUDE");
    reader.setContentHandler (handler);
    reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
    strResult += "\nIp2Longitude: " + handler.getResult();
    handler.setElementToSearchFor ("IP2ISP");
    reader.setContentHandler (handler);
    reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
    strResult += "\nIp2ISP: " + handler.getResult();
    handler.setElementToSearchFor ("ANONYMOUSPROXY");
    reader.setContentHandler (handler);
    reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
    strResult += "\nAnonymous Proxy: " + handler.getResult();
    handler.setElementToSearchFor ("FREEMAIL");
    reader.setContentHandler (handler);
    reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
    strResult += "\nFree Email: " + handler.getResult();
    handler.setElementToSearchFor ("BINCOUNTRYMATCH");
    reader.setContentHandler (handler);
    reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
    strResult += "\nBIN Country Match: " + handler.getResult();
    handler.setElementToSearchFor ("BINNAMEMATCH");
    reader.setContentHandler (handler);
    reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
    strResult += "\nBIN Name Match: " + handler.getResult();
    handler.setElementToSearchFor ("BINPHONEMATCH");
    reader.setContentHandler (handler);
    reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
    strResult += "\nBIN Phone Match: " + handler.getResult();
    handler.setElementToSearchFor ("BINCOUNTRY");
    reader.setContentHandler (handler);
    reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
    strResult += "\nBIN Country: " + handler.getResult();
    handler.setElementToSearchFor ("BINBANKNAME");
    reader.setContentHandler (handler);
    reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
    strResult += "\nBIN Bank Name: " + handler.getResult();
    handler.setElementToSearchFor ("BINBANKPHONE");
    reader.setContentHandler (handler);
    reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
    strResult += "\nBIN Bank Phone: " + handler.getResult();

```

```

        handler.setElementToSearchFor ("POSTALCITYMATCH");
        reader.setContentHandler (handler);
        reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
        strResult += "\nPostal City Match: " + handler.getResult();
        handler.setElementToSearchFor ("PHONECITYMATCH");
        reader.setContentHandler (handler);
        reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
        strResult += "\nPhone City Match: " + handler.getResult();
        handler.setElementToSearchFor ("SHIPFORWARD");
        reader.setContentHandler (handler);
        reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
        strResult += "\nShip Forward: " + handler.getResult();
        handler.setElementToSearchFor ("CREDITSAVAILABLE");
        reader.setContentHandler (handler);
        reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
        strResult += "\nCredits Available: " + handler.getResult();
        handler.setElementToSearchFor ("FRAUDSCORE");
        reader.setContentHandler (handler);
        reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
        strResult += "\nFraud Score: " + handler.getResult();
        handler.setElementToSearchFor ("QUERYID");
        reader.setContentHandler (handler);
        reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
        strResult += "\nReference ID: " + handler.getResult();
        handler.setElementToSearchFor ("MESSAGE");
        reader.setContentHandler (handler);
        reader.parse ( new InputSource (new StringReader
(this.soapMessage_.getContent().toString() ));
        strResult += "\nMessage: " + handler.getResult();
        returnValue = strResult;
    }
    catch (Exception exception)
    {
        exception.printStackTrace ();
    }
    return returnValue;
}

```