

# **PostalCodeWorld™ Canada Postal Code Web Service**

<b>1. Overview</b> .....	3
1.1 Overview of the PostalCodeWorld Canada Web Service .....	4
1.2 Front End of PostalCodeWorld Canada Postal Code Web Service .....	4
1.2.1. Integration with In-house System.....	5
1.2.2. Direct Access to Hosted Web Service.....	6
1.3 Back End of PostalCodeWorld Canada Postal Code Web Service .....	7
1.4 Process Flow Overview .....	7
<b>2. Implementation</b> .....	8
2.1 Implementation Details .....	9
2.2 Basic Parameters - Input .....	12
2.3 Basic Parameters - Output .....	12
2.4 List of possible value for MESSAGE field .....	13
2.5 Structure of the Request and Response .....	13
2.5.1. Request by SOAP .....	13
2.5.2. Response by SOAP .....	14
2.5.3. Request by HTTP-GET .....	14
2.5.4. Response by HTTP-GET .....	14
2.5.5. Request by HTTP-POST.....	15
2.5.6. Response by HTTP-POST .....	15
<b>3. Design Information</b> .....	17
3.1 Placement of PostalCodeWorld Canada Postal Code Web Service .....	17
<b>Appendix I : Data File Spesification</b> .....	18
<b>Appendix II : Sample Code</b> .....	24

## **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 PostalCodeWorld Canada Postal Code Web Service.

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

Email : [sales@fraudlabs.com](mailto:sales@fraudlabs.com)

## 1.1 Overview of the PostalCodeWorld Canada Web Service

PostalCodeWorld Canada delivers scalable web services solutions that help you to obtain geographical and other information about Canada. We are able to identify the city name, province name, phone area code, timezone, daylight saving, latitude, longitude, population, total dwellings, elevation and street name in Canada, as well as other important information by using our affordable PostalCodeWorld Canada database and technology.

PostalCodeWorld Canada Postal Code Web Service is hosted, programmable XML Web Service that allows exchange of data between systems. It is hosted on redundant servers and 24x7x365 monitoring. Customers can integrate our web service regardless of their web server and other business solutions. This XML web service is used under authorized license to ZipCodeWorld.com, the global leader in postal code service industry, as well as the leading provider of postal code information to small businesses worldwide.

PostalCodeWorld Canada Postal Code Web Service is very easy to set up and use as it uses platform independent XML format. You may get precise records by postalcodes geographical location within a minute of time.

### **Key Features Include:**

- Pinpoints the precise records of location, with area name, province name, city name, latitude and longitude, and many more using Postal Code lookups
- Provides a complete XML-based Web Services API
- Contains sample code examples for ease integration

## 1.2 Front End of PostalCodeWorld Canada Postal Code 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 PostalCodeWorld Canada Postal Code 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. Our sample codes in different languages are available at: <http://www.fraudlabs.com/postalcodeworldCanadasamplecodes.aspx> . Log on 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 - VB.NET:  
[http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada\\_WebServiceClientVB.zip](http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada_WebServiceClientVB.zip)
  - ii. Microsoft ASP.Net - C#:  
[http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada\\_WebServiceClientCSharp.zip](http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada_WebServiceClientCSharp.zip)
  - iii. Microsoft ASP 3.0:  
[http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada\\_WebServiceClientAsp.zip](http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada_WebServiceClientAsp.zip)
  - iv. Java/ Apache:  
[http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada\\_WebServiceClientJava.zip](http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada_WebServiceClientJava.zip)
  - v. PHP:  
[http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada\\_WebServiceClientPHP.zip](http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada_WebServiceClientPHP.zip)
  - vi. Python:  
[http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada\\_WebServiceClientPython.zip](http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada_WebServiceClientPython.zip)
  - vii. Perl:  
[http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada\\_WebServiceClientPerl.zip](http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada_WebServiceClientPerl.zip)
  - viii. ColdFusion MX:  
[http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada\\_WebServiceClientColdFusion.zip](http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada_WebServiceClientColdFusion.zip)
  - ix. Access 2000:  
[http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada\\_WebServiceClientAccess.zip](http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada_WebServiceClientAccess.zip)
  - x. Excel 2000:  
[http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada\\_WebServiceClientExcel.zip](http://www.fraudlabs.com/samplecode/PostalCodeWorldCanada_WebServiceClientExcel.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 PostalCodeWorld™ Canada, 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 PostalCodeWorld™ Canada, 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

### **1.2.2. Direct Access to Hosted Web Service**

- I. To use our web service directly, please log on to:  
[http://ws.fraudlabs.com/PostalCodeWorldCanada\\_WebService.asmx](http://ws.fraudlabs.com/PostalCodeWorldCanada_WebService.asmx)
- II. Click on "PostalCodeWorld\_Canada"
- 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 PostalCodeWorld™ Canada, 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 PostalCodeWorld™ Canada, 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 you fill in the particular field

V. Click on “Invoke” and result is provided

### 1.3 Back End of PostalCodeWorld Canada Postal Code 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 PostalCodeWorld Canada Postal Code Web Service.

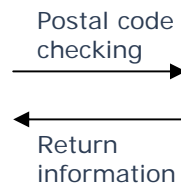
The process works as follows:

1. User enters Postal code that to be searched
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 retrieve related information from database
4. Returns and display the precise information

### 1.4 Process Flow Overview



**Step 1**  
User enters Canada  
Postal code



**Step 2**  
PostalCodeWorldCanada Web Service  
(Geographical information retrieval)

## **2. Implementation**

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 PostalCodeWorld Canada Postal Code Web Service.

The implementation section covers the following topics:

- PostalCodeWorld\_Canada Function

For those with solid SOAP programming knowledge, integration basics are included below. POST all requests to:

[http://ws.fraudlabs.com/PostalCodeWorldCanada\\_WebService.asmx](http://ws.fraudlabs.com/PostalCodeWorldCanada_WebService.asmx)

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

Email : [sales@fraudlabs.com](mailto:sales@fraudlabs.com)

## 2.1 Implementation Details

This section was created for those who wish to develop applications that make use of PostalCodeWorld Canada Postal Code Web Service.

### SOAP – PostalCodeWorld Canada Function Request

```
POST /ws.fraudlabs.com_non_ssl/PostalCodeWorldCanada_WebService.asmx
HTTP/1.1
Host: ws.fraudlabs.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://ws.fraudlabs.com/PostalCodeWorld_Canada"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <PostalCodeWorld_Canada xmlns="http://ws.fraudlabs.com/">
      <PostalCode>string</PostalCode>
      <LICENSE>string</LICENSE>
    </PostalCodeWorld_Canada>
  </soap:Body>
</soap:Envelope>
```

### SOAP – PostalCodeWorld Canada Function Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <PostalCodeWorld_CanadaResponse xmlns="http://ws.fraudlabs.com/">
      <PostalCodeWorld_CanadaResult>
        <CREDITSAVAILABLE>string</CREDITSAVAILABLE>
        <POSTAL_CODE>string</POSTAL_CODE>
        <PROVINCE>string</PROVINCE>
        <CITY>string</CITY>
        <PROVINCE_ABBR>string</PROVINCE_ABBR>
        <AREA_CODE>string</AREA_CODE>
        <CITY_FLAG>string</CITY_FLAG>
        <TIME_ZONE>float</TIME_ZONE>
        <DAY_LIGHT_SAVING>string</DAY_LIGHT_SAVING>
        <LATITUDE>float</LATITUDE>
        <LONGITUDE>float</LONGITUDE>
        <ELEVATION>float</ELEVATION>
      </PostalCodeWorld_CanadaResult>
    </PostalCodeWorld_CanadaResponse>
  </soap:Body>
</soap:Envelope>
```

```

    <POPULATION>float</POPULATION>
    <DWELLING>float</DWELLING>
    <AREA_NAME>string</AREA_NAME>
    <STREET_NAME>string</STREET_NAME>
    <STREET_TYPE_CODE>string</STREET_TYPE_CODE>
    <STREET_DIR_CODE>string</STREET_DIR_CODE>
    <STREET_SEQ_CODE>string</STREET_SEQ_CODE>
    <STREET_FROM_NO>string</STREET_FROM_NO>
    <STREET_FROM_SUFFIX>string</STREET_FROM_SUFFIX>
    <STREET_TO_NO>string</STREET_TO_NO>
    <STREET_TO_SUFFIX>string</STREET_TO_SUFFIX>
    <MESSAGE>string</MESSAGE>
  </PostalCodeWorld_CanadaResult>
</PostalCodeWorld_CanadaResponse>
</soap:Body>
</soap:Envelope>

```

## HTTP GET- PostalCodeWorld Canada Function Request

```

GET
/ws.fraudlabs.com_non_ssl/postalcodeworldCanada_webservice.asmx/PostalC
odeWorld_Canada?PostalCode=string&LICENSE=string HTTP/1.1
Host: ws.fraudlabs.com

```

## HTTP GET- PostalCodeWorld Canada Function Response

```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<POSTALCODEWORLD_CANADA xmlns="http://ws.fraudlabs.com/">
  <CREDITSAVAILABLE>string</CREDITSAVAILABLE>
  <POSTAL_CODE>string</POSTAL_CODE>
  <PROVINCE>string</PROVINCE>
  <CITY>string</CITY>
  <PROVINCE_ABBR>string</PROVINCE_ABBR>
  <AREA_CODE>string</AREA_CODE>
  <CITY_FLAG>string</CITY_FLAG>
  <TIME_ZONE>float</TIME_ZONE>
  <DAY_LIGHT_SAVING>string</DAY_LIGHT_SAVING>
  <LATITUDE>float</LATITUDE>
  <LONGITUDE>float</LONGITUDE>
  <ELEVATION>float</ELEVATION>
  <POPULATION>float</POPULATION>
  <DWELLING>float</DWELLING>
  <AREA_NAME>string</AREA_NAME>
  <STREET_NAME>string</STREET_NAME>
  <STREET_TYPE_CODE>string</STREET_TYPE_CODE>
  <STREET_DIR_CODE>string</STREET_DIR_CODE>
  <STREET_SEQ_CODE>string</STREET_SEQ_CODE>
  <STREET_FROM_NO>string</STREET_FROM_NO>

```

```
<STREET_FROM_SUFFIX>string</STREET_FROM_SUFFIX>
<STREET_TO_NO>string</STREET_TO_NO>
<STREET_TO_SUFFIX>string</STREET_TO_SUFFIX>
<MESSAGE>string</MESSAGE>
</POSTALCODEWORLD_CANADA>
```

## **HTTP POST- PostalCodeWorld Canada Function Request**

```
POST
/ws.fraudlabs.com_non_ssl/postalcodeworldCanada_webservice.asmx/PostalC
odeWorld_Canada HTTP/1.1
Host: ws.fraudlabs.com
Content-Type: application/x-www-form-urlencoded
Content-Length: length

PostalCode=string&LICENSE=string
```

## **HTTP POST- PostalCodeWorld Canada Function Response**

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<POSTALCODEWORLD_CANADA xmlns="http://ws.fraudlabs.com/">
  <CREDITSAVAILABLE>string</CREDITSAVAILABLE>
  <POSTAL_CODE>string</POSTAL_CODE>
  <PROVINCE>string</PROVINCE>
  <CITY>string</CITY>
  <PROVINCE_ABBR>string</PROVINCE_ABBR>
  <AREA_CODE>string</AREA_CODE>
  <CITY_FLAG>string</CITY_FLAG>
  <TIME_ZONE>float</TIME_ZONE>
  <DAY_LIGHT_SAVING>string</DAY_LIGHT_SAVING>
  <LATITUDE>float</LATITUDE>
  <LONGITUDE>float</LONGITUDE>
  <ELEVATION>float</ELEVATION>
  <POPULATION>float</POPULATION>
  <DWELLING>float</DWELLING>
  <AREA_NAME>string</AREA_NAME>
  <STREET_NAME>string</STREET_NAME>
  <STREET_TYPE_CODE>string</STREET_TYPE_CODE>
  <STREET_DIR_CODE>string</STREET_DIR_CODE>
  <STREET_SEQ_CODE>string</STREET_SEQ_CODE>
  <STREET_FROM_NO>string</STREET_FROM_NO>
  <STREET_FROM_SUFFIX>string</STREET_FROM_SUFFIX>
  <STREET_TO_NO>string</STREET_TO_NO>
  <STREET_TO_SUFFIX>string</STREET_TO_SUFFIX>
  <MESSAGE>string</MESSAGE>
</POSTALCODEWORLD_CANADA>
```

A WSDL is available at:

[http://ws.fraudlabs.com/PostalCodeWorldCanada\\_WebService.asmx?wsdl](http://ws.fraudlabs.com/PostalCodeWorldCanada_WebService.asmx?wsdl)

A description of the PostalCodeWorldCanada\_WebService operation is available at:

[http://ws.fraudlabs.com/PostalCodeWorldCanada\\_WebService.asmx?op=PostalCodeWorld\\_Canada](http://ws.fraudlabs.com/PostalCodeWorldCanada_WebService.asmx?op=PostalCodeWorld_Canada)

## 2.2 Basic Parameters - Input

Field	Format	Description
PostalCode	Required	Canada Postal Code
LICENSE	Required	License key for free license and premium users.

## 2.3 Basic Parameters - Output

Field	Format	Description
POSTAL_CODE	string	The Postal Code of the city.
CITY	string	The name of the city of the Postal Code.
PROVINCE	string	The name of the province to that Postal Code.
PROVINCE_ABBR	string	The two-letter abbreviation for the province to that Postal Code.
AREA_CODE	string	The telephone area code belonging to the particular postal code is located.
CITY_FLAG	string	The flag to determine the Postal Code is in a City area.
TIME_ZONE	string	The time zone number represents by the hours past the Greenwich Time Zone.
DAY_LIGHT_SAVING	string	This flag indicates whether the city observe day light saving.
LATITUDE	float	The geographic coordinate of a point measured in degrees north or south of the equator.
LONGITUDE	float	The geographic coordinate of a point measured in degrees east or west of the Greenwich meridian.
ELEVATION	float	The average elevation of Postal Code in meter.
POPULATION	single	The population for Forward Sortation Area as reported by Census 2001.
DWELLING	float	The total private dwellings for Forward Sortation Area as reported by Census 2001.
AREA_NAME	string	The major community or greater municipality grouping that contains the street address or its delivery installation recognized by Canada Post.
STREET_NAME	string	The official civic name of a roadway or artery

		recognized by Canada Post.
STREET_TYPE_CODE	string	The official description used to identify the type of artery or roadway.
STREET_DIR_CODE	string	The street direction component of an official street name.
STREET_SEQ_CODE	string	This code identifies the sequence associated with the range of street numbers.
STREET_FROM_NO	string	The lowest street number in a range of municipal street addresses.
STREET_FROM_SUFFIX	string	The address suffix associated with the street address from number.
STREET_TO_NO	string	The highest street number in a range of municipal street addresses.
STREET_TO_SUFFIX	string	The address suffix associated with the street address to number.
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.
MESSAGE	String	Web Service Message Response

## 2.4 List of possible value for MESSAGE field

PostalCodeWorld Canada Web Service Error Message
• Invalid license key
• Postal Code xx not found in database. Please try again.
• Postal Code and License Key cannot be blank
• No credit available
• License key was expired

## 2.5 Structure of the Request and Response

### 2.5.1. Request by SOAP

A valid request posted from user should look something like this:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <PostalCodeWorld_Canada xmlns="http://ws.fraudlabs.com/">
      <PostalCode>A0A 1A0</ZIPCode>
      <LICENSE>xx-xxxx-xxxx</LICENSE>
    </PostalCodeWorld_Canada >
  </soap:Body>
</soap:Envelope>
```

### 2.5.2. Response by SOAP

After posting a valid request, ZIPCodeWorld\_US will return several parameters. An example of a response returned ZIPCodeWorld\_US is shown below:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <PostalCodeWorld_CanadaResponse xmlns="http://ws.fraudlabs.com/">
      <PostalCodeWorld_CanadaResult>
        <CREDITSAVAILABLE>463</CREDITSAVAILABLE>
        <PROVINCE>NEWFOUNDLAND AND LABRADOR</PROVINCE>
        <CITY>AQUAFORTE</CITY>
        <PROVINCE_ABBR>NL</PROVINCE_ABBR>
        <AREA_CODE>709</AREA_CODE>
        <CITY_FLAG>N</CITY_FLAG>
        <TIME_ZONE>3.5</TIME_ZONE>
        <DAY_LIGHT_SAVING>Y</DAY_LIGHT_SAVING>
        <LATITUDE>47.0073471</LATITUDE>
        <LONGITUDE>-52.95892</LONGITUDE>
        <ELEVATION>205</ELEVATION>
        <POPULATION>57949</POPULATION>
        <DWELLING>26014</DWELLING>
        <AREA_NAME>AQUAFORTE</AREA_NAME>
        <STREET_NAME>AQUAFORTE</STREET_NAME>
        <STREET_TYPE_CODE />
        <STREET_DIR_CODE />
        <STREET_SEQ_CODE />
        <STREET_FROM_NO />
        <STREET_FROM_SUFFIX />
        <STREET_TO_NO />
        <STREET_TO_SUFFIX />
        <MESSAGE />
      </PostalCodeWorld_CanadaResult>
    </PostalCodeWorld_CanadaResponse>
  </soap:Body>
</soap:Envelope>
```

### 2.5.3. Request by HTTP-GET

A valid request posted by user:

```
GET
/ws.fraudlabs.com_non_ssl/PostalCodeWorldCanada_WebService.asmx/PostalC
odeWorld_Canada?PostalCode=A0A 1A0&LICENSE=XX-XXXX-XXXX HTTP/1.1
Host: ws.fraudlabs.com
```

### 2.5.4. Response by HTTP-GET

An example of a response returned PostalCodeWorld\_Canada is shown below:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<PostalCodeWorld_Canada xmlns="http://ws.fraudlabs.com/">
  <CREDITSAVAILABLE>463</CREDITSAVAILABLE>
  <PROVINCE>NEWFOUNDLAND AND LABRADOR</PROVINCE>
  <CITY>AQUAFORTE</CITY>
  <PROVINCE_ABBR>NL</PROVINCE_ABBR>
  <AREA_CODE>709</AREA_CODE>
  <CITY_FLAG>N</CITY_FLAG>
  <TIME_ZONE>3.5</TIME_ZONE>
  <DAY_LIGHT_SAVING>Y</DAY_LIGHT_SAVING>
  <LATITUDE>47.0073471</LATITUDE>
  <LONGITUDE>-52.95892</LONGITUDE>
  <ELEVATION>205</ELEVATION>
  <POPULATION>57949</POPULATION>
  <DWELLING>26014</DWELLING>
  <AREA_NAME>AQUAFORTE</AREA_NAME>
  <STREET_NAME>AQUAFORTE</STREET_NAME>
  <STREET_TYPE_CODE />
  <STREET_DIR_CODE />
  <STREET_SEQ_CODE />
  <STREET_FROM_NO />
  <STREET_FROM_SUFFIX />
  <STREET_TO_NO />
  <STREET_TO_SUFFIX />
  <MESSAGE />
</PostalCodeWorld_Canada>
```

### 2.5.5. Request by HTTP-POST

A valid request posted by user:

```
POST
/ws.fraudlabs.com_non_ssl/PostalCodeWorldCanada_WebService.asmx/PostalC
odeWorld_Canada HTTP/1.1
Host: ws.fraudlabs.com
Content-Type: application/x-www-form-urlencoded
Content-Length: length

PostalCode=A0A 1A0&LICENSE=XX-XXXX-XXXX
```

### 2.5.6. Response by HTTP-POST

An example of a response returned by PostalCodeWorld\_Canada is shown below:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>
<PostalCodeWorld_Canada xmlns="http://ws.fraudlabs.com/">
  <CREDITSAVAILABLE>463</CREDITSAVAILABLE>
  <PROVINCE>NEWFOUNDLAND AND LABRADOR</PROVINCE>
  <CITY>AQUAFORTE</CITY>
  <PROVINCE_ABBR>NL</PROVINCE_ABBR>
  <AREA_CODE>709</AREA_CODE>
  <CITY_FLAG>N</CITY_FLAG>
  <TIME_ZONE>3.5</TIME_ZONE>
  <DAY_LIGHT_SAVING>Y</DAY_LIGHT_SAVING>
  <LATITUDE>47.0073471</LATITUDE>
  <LONGITUDE>-52.95892</LONGITUDE>
  <ELEVATION>205</ELEVATION>
  <POPULATION>57949</POPULATION>
  <DWELLING>26014</DWELLING>
  <AREA_NAME>AQUAFORTE</AREA_NAME>
  <STREET_NAME>AQUAFORTE</STREET_NAME>
  <STREET_TYPE_CODE />
  <STREET_DIR_CODE />
  <STREET_SEQ_CODE />
  <STREET_FROM_NO />
  <STREET_FROM_SUFFIX />
  <STREET_TO_NO />
  <STREET_TO_SUFFIX />
  <MESSAGE />
</PostalCodeWorld_Canada>
```

### **3. Design Information**

This section provides suggestion regarding the placing of PostalCodeWorld Canada Postal Code 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 PostalCodeWorld Canada Postal Code Web Service**

As PostalCodeWorld Canada Postal Code Web Service provides you very informative details by using Postal code, this service can be placed on any forms on your website, offering a seamless integration.

**Description:** Geographical information is needed for business purpose or even for online transaction in Canada

**Level of security:** High

**Site Type(s):**

- e-Commerce Solutions
- Internet Retailers
- Software Developers

**Benefits:**

- Get information from affordable and complete database which contains over 800,000 unique postal code records with individual latitude and longitude.
- With our PostalCodeWorld database, applications can be built easily to support business logics
- Obtain additional information on Canada addressing system and any other aspect of postal operations
- Enhance customer's information in Canada
- Improve data entry speed and accuracy

## Appendix I : Data File Spesification

### 1. POSTAL\_CODE

Canadian Postal Codes are made up of 6 digits/characters. The sequence is in AOA OAO format where A is Alphabetical character and O is a numerical character.

### 2. CITY

This is the name of city for the Postal Code. It is used at the last line of postal address. The name can represent a city, a postal entity, a community or a municipality name.

### 3. PROVINCE

This field belongs to the province name of the Postal Code.

### 4. PROVINCE\_ABBR

This is a two-character abbreviation name for the PROVINCE.

### 5. AREA\_CODE

The telephone area code belonging to the particular Postal code is located.

### 6. CITY\_FLAG

The flag to determine the Postal Code is in a City area. If "Y", the Postal Code fall into a city area; and "N" for otherwise.

### 7. LONGITUDE

The geographical coordinate of a centroid measured in degree east or west of the Greenwich meridian.

### 8. LATITUDE

The geographical coordinate of a centroid measured in degree north or south of the equator.

### 9. TIME\_ZONE

This number represents the hours past the Greenwich Meridian Time Zone (GMT).

NUMBER	TIME_ZONE
4	ATLANTIC
5	EASTERN
6	CENTRAL
7	MOUNTAIN
8	PACIFIC
9	ALASKA
10	HAWAII-ALEUTIAN
11	SAMOA

### 10. DAY\_LIGHT\_SAVING

The flag represents whether the county observes day light saving during the summer. "Y" for YES; "N" for NO.

**11. ELEVATION**

The average elevation of Postal Code centroid measured in meter.

**12. POPULATION**

The population in Forward Sortation Area as reported by Census 2001.

Note: Population is not reported by 6 digits/characters postal code. Instead, population is valid for first 3 digits/characters Forward Sortation Area (FSA).

**13. DWELLING**

The total private dwellings in Forward Sortation Area as reported by Census 2001.

Note: Total private dwellings are not being reported by 6 digits/characters postal code. Instead, the number is valid for first 3 digits/characters Forward Sortation Area (FSA).

**14. AREA\_NAME**

The major community or greater municipality grouping that contains the street address or its delivery installation recognized by Canada Post.

**15. STREET\_NAME**

The official civic name of a roadway or artery recognized by Canada Post.

**16. STREET\_TYPE\_CODE**

The official description used to identify the type of artery or roadway. The valid values are space, or:

TYPE	ABBREVIATION
Abbey	ABBEY
Acres	ACRES
Allée	ALLÉE
Alley	ALLEY
Autoroute	AUT
Avenue (English)	AVE
Avenue (French)	AV
Bay	BAY
Beach	BEACH
Bend	BEND
Boulevard (English)	BLVD
Boulevard (French)	BOUL
By-Pass	BYPASS
Byway	BYWAY
Campus	CAMPUS
Cape	CAPE
Carré	CAR
Carrefour	CARREF
Centre (English)	CTR
Centre (French)	C
Cercle	CERCLE
Chase	CHASE
Chemin	CH
Circle	CIR

Circuit	CIRCT
Close	CLOSE
Common	COMMON
Concession	CONC
Corners	CRNRS
Côte	CÔTE
Cour	COUR
Cours	COURS
Court	CRT
Cove	COVE
Crescent	CRES
Croissant	CROIS
Crossing	CROSS
Cul-de-sac	CDS
Dale	DALE
Dell	DELL
Diversion	DIVERS
Downs	DOWNS
Drive	DR
Échangeur	ÉCH
End	END
Esplanade	ESPL
Estates	ESTATE
Expressway	EXPY
Extension	EXTEN
Farm	FARM
Field	FIELD
Forest	FOREST
Freeway	FWY
Front	FRONT
Gardens	GDNS
Gate	GATE
Glade	GLADE
Glen	GLEN
Green	GREEN
Grounds	GRNDS
Grove	GROVE
Harbour	HARBR
Heath	HEATH
Heights	HTS
Highlands	HGHLDS
Highway	HWY
Hill	HILL
Hollow	HOLLOW
Île	ÎLE
Impasse	IMP
Inlet	INLET
Island	ISLAND
Key	KEY
Knoll	KNOLL
Landing	LANDNG

Lane	LANE
Limits	LMTS
Line	LINE
Link	LINK
Lookout	LKOUT
Loop	LOOP
Mall	MALL
Manor	MANOR
Maze	MAZE
Meadow	MEADOW
Mews	MEWS
Montée	MONTÉE
Moor	MOOR
Mount	MOUNT
Mountain	MTN
Orchard	ORCH
Parade	PARADE
Parc	PARC
Park	PK
Parkway	PKY
Passage	PASS
Path	PATH
Pathway	PTWAY
Pines	PINES
Place (English)	PL
Place (French)	PLACE
Plateau	PLAT
Plaza	PLAZA
Point	PT
Pointe	POINTE
Port	PORT
Private	PVT
Promenade	PROM
Quai	QUAI
Quay	QUAY
Ramp	RAMP
Rang	RANG
Range	RG
Ridge	RIDGE
Rise	RISE
Road	RD
Rond Point	RDPT
Route	RTE
Row	ROW
Rue	RUE
Ruelle	RLE
Run	RUN
Sentier	SENT
Square	SQ
Street	ST
Subdivision	SUBDIV

Terrace	TERR
Terrasse	TSSE
Thicket	THICK
Towers	TOWERS
Townline	TLINE
Trail	TRAIL
Turnabout	TRNABT
Vale	VALE
Via	VIA
View	VIEW
Village	VILLGE
Villas	VILLAS
Vista	VISTA
Voie	VOIE
Walk	WALK
Way	WAY
Wharf	WHARF
Wood	WOOD
Wynd	WYND

## 17. STREET\_DIR\_CODE

The street direction component of an official street name. The valid Street Direction Code values are:

CODE	DIRECTION
N	NORTH/NORD
S	SOUTH/SUD
E	EAST/EST
W	WEST
O	OUEST
NE	NORTHWEST
NO	NORD-OUEST
SE	SOUTHEAST/SUD-EST
SW	SOUTHWEST
SO	SUD-OUEST

## 18. STREET\_SEQ\_CODE

This code identifies the sequence associated with the range of street numbers. The valid Street Address Sequence Code values are:

CODE	SEQUENCE
1	Odd
2	Even
3	Consecutive
4	Consecutive (route service box – streets served by route only)

**19. STREET\_FROM\_NO**

The lowest street number in a range of municipal street addresses.

**20. STREET\_FROM\_SUFFIX**

The address suffix associated with the street address from number. Examples of street numbers with suffixes are 14½ and 22B. A numeric value denotes a fractional suffix. Fractional code valid values:

CODE	SUFFIX
1	¼
2	½
3	¾
A-Z	A-Z respectively

**21. STREET\_TO\_NO**

The highest street number in a range of municipal street addresses.

**22. STREET\_TO\_SUFFIX**

The address suffix associated with the street address to number. Examples of street numbers with suffixes are 14½ and 22B. A numeric value denotes a fractional suffix. Fractional code valid values:

CODE	SUFFIX
1	¼
2	½
3	¾
A-Z	A-Z respectively

## Appendix II : Sample Code

PostalCodeWorldCanada 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/postalcodeworldCanadasamplecodes.aspx>

### i. ASP.NET – VB.NET (SOAP)

```
Dim x_PostalCodeWorld As New
PostalCodeWorldCanada_WebService.PostalCodeWorldCanada_WebService
Dim oPostalCodeWorld As New POSTALCODEWORLD_CANADA

Try
oPostalCodeWorld =
x_PostalCodeWorld.PostalCodeWorld_Canada(Me.txtPostalCode.Text,
Me.txtLicense.Text)

Me.txtResult.Text = "Postal Code:" + oPostalCodeWorld.POSTAL_CODE &
vbNewLine
Me.txtResult.Text += "City:" + oPostalCodeWorld.CITY & vbNewLine
Me.txtResult.Text += "Province:" + oPostalCodeWorld.PROVINCE &
vbNewLine
Me.txtResult.Text += "Province ABBR:" + oPostalCodeWorld.PROVINCE_ABBR
& vbNewLine
Me.txtResult.Text += "City Flag:" + oPostalCodeWorld.CITY_FLAG &
vbNewLine
Me.txtResult.Text += "Latitude:" + oPostalCodeWorld.LATITUDE.ToString &
vbNewLine
Me.txtResult.Text += "Longitude:" + oPostalCodeWorld.LONGITUDE.ToString
& vbNewLine
Me.txtResult.Text += "Time Zone:" + oPostalCodeWorld.TIME_ZONE.ToString
& vbNewLine
Me.txtResult.Text += "Day Light Saving:" +
oPostalCodeWorld.DAY_LIGHT_SAVING & vbNewLine
Me.txtResult.Text += "Elevation:" + oPostalCodeWorld.ELEVATION.ToString
& vbNewLine
Me.txtResult.Text += "Population:" +
oPostalCodeWorld.POPULATION.ToString & vbNewLine
Me.txtResult.Text += "Dwelling:" + oPostalCodeWorld.DWELLING.ToString &
vbNewLine
Me.txtResult.Text += "Area Name:" + oPostalCodeWorld.AREA_CODE &
vbNewLine
Me.txtResult.Text += "Street Name:" + oPostalCodeWorld.STREET_NAME &
vbNewLine
Me.txtResult.Text += "Street Type Code:" +
oPostalCodeWorld.STREET_TYPE_CODE & vbNewLine
Me.txtResult.Text += "Street Dir Code:" +
oPostalCodeWorld.STREET_DIR_CODE & vbNewLine
Me.txtResult.Text += "Street Seq Code:" +
oPostalCodeWorld.STREET_SEQ_CODE.ToString & vbNewLine
Me.txtResult.Text += "Street From No:" +
oPostalCodeWorld.STREET_FROM_NO.ToString & vbNewLine
Me.txtResult.Text += "Street From Suffix:" +
```

```

oPostalCodeWorld.STREET_FROM_SUFFIX.ToString & vbNewLine
Me.txtResult.Text += "Street To No:" +
oPostalCodeWorld.STREET_TO_NO.ToString & vbNewLine
Me.txtResult.Text += "Street To Suffix:" +
oPostalCodeWorld.STREET_TO_SUFFIX.ToString & vbNewLine
Me.txtResult.Text += "Credits Available:" +
oPostalCodeWorld.CREDITSAVAILABLE & vbNewLine
Me.txtResult.Text += "Message:" + oPostalCodeWorld.MESSAGE & vbNewLine

Catch ex As Exception
Response.Write(ex.Message)
End Try
End Sub

```

## ii. ASP.NET – VB.NET (HTTP)

```

Private Sub Lookup()
Dim x_PostalCodeWorld As New
PostalCodeWorldCanada_WebService.PostalCodeWorldCanada_WebService
Dim oPostalCodeWorld As New POSTALCODEWORLD_CANADA

Try
' create the request URL
Dim x_builder As String
' add the host element of the URL
x_builder =
"http://ws.fraudlabs.com/PostalCodeWorldCanada_WebService.asmx/PostalCo
deWorld_Canada?" & _
"PostalCode=" + Me.txtPostalCode.Text & _
"&LICENSE=" + Me.txtLicense.Text
' create the web client and obtain the response data as a byte array
Dim x_web_client As WebClient = New WebClient
Dim x_response() As Byte =
x_web_client.DownloadData(x_builder.ToString())
' process the string result to obtain a validation result
processResultString(Encoding.Default.GetString(x_response))
Catch ex As Exception
Response.Write(ex.Message)
End Try
End Sub

Private Sub processResultString(ByVal p_result As String)
Dim indexOpen As Integer
Dim indexClose As Integer
Dim strParam As String

strParam = "<POSTAL_CODE>"
indexOpen = p_result.IndexOf(strParam) + strParam.Length
indexClose = p_result.IndexOf("</POSTAL_CODE>")
Me.txtResult.Text = "Postal Code: "
If indexClose <> -1 Then
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen)
End If
strParam = "<CITY>"

```

```
indexOpen = p_result.IndexOf(strParam) + strParam.Length
indexClose = p_result.IndexOf("</CITY>")
Me.txtResult.Text += vbNewLine & "CITY: "
If indexClose <> -1 Then
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen)
End If
strParam = "<PROVINCE>"
indexOpen = p_result.IndexOf(strParam) + strParam.Length
indexClose = p_result.IndexOf("</PROVINCE>")
Me.txtResult.Text += vbNewLine & "PROVINCE: "
If indexClose <> -1 Then
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen)
End If
strParam = "<PROVINCE_ABBR>"
indexOpen = p_result.IndexOf(strParam) + strParam.Length
indexClose = p_result.IndexOf("</PROVINCE_ABBR>")
Me.txtResult.Text += vbNewLine & "PROVINCE_ABBR: "
If indexClose <> -1 Then
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen)
End If
strParam = "<AREA_CODE>"
indexOpen = p_result.IndexOf(strParam) + strParam.Length
indexClose = p_result.IndexOf("</AREA_CODE>")
Me.txtResult.Text += vbNewLine & "AREA_CODE: "
If indexClose <> -1 Then
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen)
End If
strParam = "<CITY_FLAG>"
indexOpen = p_result.IndexOf(strParam) + strParam.Length
indexClose = p_result.IndexOf("</CITY_FLAG>")
Me.txtResult.Text += vbNewLine & "CITY_FLAG: "
If indexClose <> -1 Then
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen)
End If
strParam = "<LATITUDE>"
indexOpen = p_result.IndexOf(strParam) + strParam.Length
indexClose = p_result.IndexOf("</LATITUDE>")
Me.txtResult.Text += vbNewLine & "Latitude: "
If indexClose <> -1 Then
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen)
End If
strParam = "<LONGITUDE>"
indexOpen = p_result.IndexOf(strParam) + strParam.Length
indexClose = p_result.IndexOf("</LONGITUDE>")
Me.txtResult.Text += vbNewLine & "Longitude: "
If indexClose <> -1 Then
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen)
End If
strParam = "<TIME_ZONE>"
indexOpen = p_result.IndexOf(strParam) + strParam.Length
```

```
indexClose = p_result.IndexOf("</TIME_ZONE>")
Me.txtResult.Text += vbNewLine & "Time Zone: "
If indexClose <> -1 Then
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen)
End If
strParam = "<DAY_LIGHT_SAVING>"
indexOpen = p_result.IndexOf(strParam) + strParam.Length
indexClose = p_result.IndexOf("</DAY_LIGHT_SAVING>")
Me.txtResult.Text += vbNewLine & "Day Light Saving: "
If indexClose <> -1 Then
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen)
End If
strParam = "<ELEVATION>"
indexOpen = p_result.IndexOf(strParam) + strParam.Length
indexClose = p_result.IndexOf("</ELEVATION>")
Me.txtResult.Text += vbNewLine & "ELEVATION: "
If indexClose <> -1 Then
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen)
End If
strParam = "<POPULATION>"
indexOpen = p_result.IndexOf(strParam) + strParam.Length
indexClose = p_result.IndexOf("</POPULATION>")
Me.txtResult.Text += vbNewLine & "POPULATION: "
If indexClose <> -1 Then
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen)
End If
strParam = "<DWELLING>"
indexOpen = p_result.IndexOf(strParam) + strParam.Length
indexClose = p_result.IndexOf("</DWELLING>")
Me.txtResult.Text += vbNewLine & "DWELLING: "
If indexClose <> -1 Then
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen)
End If
strParam = "<AREA_NAME>"
indexOpen = p_result.IndexOf(strParam) + strParam.Length
indexClose = p_result.IndexOf("</AREA_NAME>")
Me.txtResult.Text += vbNewLine & "AREA_NAME: "
If indexClose <> -1 Then
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen)
End If
strParam = "<STREET_NAME>"
indexOpen = p_result.IndexOf(strParam) + strParam.Length
indexClose = p_result.IndexOf("</STREET_NAME>")
Me.txtResult.Text += vbNewLine & "STREET_NAME: "
If indexClose <> -1 Then
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen)
End If
strParam = "<STREET_TYPE_CODE>"
indexOpen = p_result.IndexOf(strParam) + strParam.Length
indexClose = p_result.IndexOf("</STREET_TYPE_CODE>")
```

```
Me.txtResult.Text += vbNewLine & "STREET_TYPE_CODE: "  
If indexClose <> -1 Then  
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -  
indexOpen)  
End If  
strParam = "<STREET_DIR_CODE>"  
indexOpen = p_result.IndexOf(strParam) + strParam.Length  
indexClose = p_result.IndexOf("</STREET_DIR_CODE>")  
Me.txtResult.Text += vbNewLine & "STREET_DIR_CODE: "  
If indexClose <> -1 Then  
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -  
indexOpen)  
End If  
strParam = "<STREET_SEQ_CODE>"  
indexOpen = p_result.IndexOf(strParam) + strParam.Length  
indexClose = p_result.IndexOf("</STREET_SEQ_CODE>")  
Me.txtResult.Text += vbNewLine & "STREET_SEQ_CODE: "  
If indexClose <> -1 Then  
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -  
indexOpen)  
End If  
strParam = "<STREET_FROM_NO>"  
indexOpen = p_result.IndexOf(strParam) + strParam.Length  
indexClose = p_result.IndexOf("</STREET_FROM_NO>")  
Me.txtResult.Text += vbNewLine & "STREET_FROM_NO: "  
If indexClose <> -1 Then  
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -  
indexOpen)  
End If  
strParam = "<STREET_FROM_SUFFIX>"  
indexOpen = p_result.IndexOf(strParam) + strParam.Length  
indexClose = p_result.IndexOf("</STREET_FROM_SUFFIX>")  
Me.txtResult.Text += vbNewLine & "STREET_FROM_SUFFIX: "  
If indexClose <> -1 Then  
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -  
indexOpen)  
End If  
strParam = "<STREET_TO_NO>"  
indexOpen = p_result.IndexOf(strParam) + strParam.Length  
indexClose = p_result.IndexOf("</STREET_TO_NO>")  
Me.txtResult.Text += vbNewLine & "STREET_TO_NO: "  
If indexClose <> -1 Then  
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -  
indexOpen)  
End If  
strParam = "<STREET_TO_SUFFIX>"  
indexOpen = p_result.IndexOf(strParam) + strParam.Length  
indexClose = p_result.IndexOf("</STREET_TO_SUFFIX>")  
Me.txtResult.Text += vbNewLine & "STREET_TO_SUFFIX: "  
If indexClose <> -1 Then  
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -  
indexOpen)  
End If  
strParam = "<CREDITSAVAILABLE>"  
indexOpen = p_result.IndexOf(strParam) + strParam.Length  
indexClose = p_result.IndexOf("</CREDITSAVAILABLE>")  
Me.txtResult.Text += vbNewLine & "Credits Available:"
```

```

If indexClose <> -1 Then
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen)
End If
strParam = "<MESSAGE>"
indexOpen = p_result.IndexOf(strParam) + strParam.Length
indexClose = p_result.IndexOf("</MESSAGE>")
Me.txtResult.Text += vbNewLine & "Message: "
If indexClose <> -1 Then
Me.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen)
End If
End Sub

```

### iii. ASP.NET – C#.NET (SOAP)

```

private void PostalCodeWorldCanadaWebService()
{
PostalCodeWorldCanada_WebService.PostalCodeWorldCanada_WebService
xPostalCodeWorld = new
PostalCodeWorldCanada_WebService.PostalCodeWorldCanada_WebService();
POSTALCODEWORLD_CANADA oPostalCodeWorld = new POSTALCODEWORLD_CANADA();

try
{
oPostalCodeWorld =
xPostalCodeWorld.PostalCodeWorld_Canada(this.txtPostalCode.Text,
this.txtLicense.Text);

this.txtResult.Text = "Postal Code:" + oPostalCodeWorld.POSTAL_CODE +
"\n";
this.txtResult.Text += "City:" + oPostalCodeWorld.CITY + "\n";
this.txtResult.Text += "Province:" + oPostalCodeWorld.PROVINCE + "\n";
this.txtResult.Text += "Province ABBR:" +
oPostalCodeWorld.PROVINCE_ABBR + "\n";
this.txtResult.Text += "Area Code:" + oPostalCodeWorld.AREA_CODE +
"\n";
this.txtResult.Text += "City Flag:" + oPostalCodeWorld.CITY_FLAG +
"\n";
this.txtResult.Text += "Latitude:" + oPostalCodeWorld.LATITUDE + "\n";
this.txtResult.Text += "Longitude:" + oPostalCodeWorld.LONGITUDE +
"\n";
this.txtResult.Text += "Time Zone:" + oPostalCodeWorld.TIME_ZONE +
"\n";
this.txtResult.Text += "Day Light Saving:" +
oPostalCodeWorld.DAY_LIGHT_SAVING + "\n";
this.txtResult.Text += "Elevation:" + oPostalCodeWorld.ELEVATION +
"\n";
this.txtResult.Text += "Population:" + oPostalCodeWorld.POPULATION +
"\n";
this.txtResult.Text += "Dwelling:" + oPostalCodeWorld.DWELLING + "\n";
this.txtResult.Text += "Area Name:" + oPostalCodeWorld.AREA_NAME +
"\n";
this.txtResult.Text += "Street Name:" + oPostalCodeWorld.STREET_NAME +
"\n";
}
}

```

```
this.txtResult.Text += "Street Type Code:" +
oPostalCodeWorld.STREET_TYPE_CODE + "\n";
this.txtResult.Text += "Street Dir Code:" +
oPostalCodeWorld.STREET_DIR_CODE + "\n";
this.txtResult.Text += "Street Seq Code:" +
oPostalCodeWorld.STREET_SEQ_CODE + "\n";
this.txtResult.Text += "Street From No:" +
oPostalCodeWorld.STREET_FROM_NO + "\n";
this.txtResult.Text += "Street From Suffix:" +
oPostalCodeWorld.STREET_FROM_SUFFIX + "\n";
this.txtResult.Text += "Street To No:" + oPostalCodeWorld.STREET_TO_NO
+ "\n";
this.txtResult.Text += "Street To Suffix:" +
oPostalCodeWorld.STREET_TO_SUFFIX + "\n";
this.txtResult.Text += "Credits Available:" +
oPostalCodeWorld.CREDITSAVAILABLE + "\n";
this.txtResult.Text += "Message:" + oPostalCodeWorld.MESSAGE + "\n";
}
catch (Exception ex)
{
Response.Write(ex.Message);
}
}
```

#### iv. ASP.NET – C#.NET (HTTP)

```
private void Lookup()
{
PostalCodeWorldCanada_WebService.PostalCodeWorldCanada_WebService
xPostalCodeWorld = new
PostalCodeWorldCanada_WebService.PostalCodeWorldCanada_WebService();
POSTALCODEWORLD_CANADA oPostalCodeWorld = new POSTALCODEWORLD_CANADA();

try
{
//create the request URL
string x_builder;
//add the host element of the URL
x_builder =
"http://ws.fraudlabs.com/PostalCodeWorldCanada_WebService.asmx/PostalCo
deWorld_Canada?";
x_builder += "POSTALCODE=" + this.txtPostalCode.Text;
x_builder += "&LICENSE=" + this.txtLicense.Text;
// create the web client and obtain the response data as a byte array
WebClient x_web_client = new WebClient();
byte [] x_response = x_web_client.DownloadData(x_builder.ToString());
// process the string result to obtain a validation result
processResultString(Encoding.Default.GetString(x_response));
}
catch (Exception ex)
{
Response.Write(ex.Message);
}
}
```

```
private void processResultString(String p_result)
{
    int indexOpen;
    int indexClose;
    string strParam;
    strParam = "<POSTAL_CODE>";
    indexOpen = p_result.IndexOf(strParam) + strParam.Length;
    indexClose = p_result.IndexOf("</POSTAL_CODE>");
    this.txtResult.Text = "Postal Code: ";
    if (indexClose != -1)
    {
        this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
    }
    strParam = "<CITY>";
    indexOpen = p_result.IndexOf(strParam) + strParam.Length;
    indexClose = p_result.IndexOf("</CITY>");
    this.txtResult.Text += "\rCITY: ";
    if (indexClose != -1)
    {
        this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
    }
    strParam = "<PROVINCE>";
    indexOpen = p_result.IndexOf(strParam) + strParam.Length;
    indexClose = p_result.IndexOf("</PROVINCE>");
    this.txtResult.Text += "\rPROVINCE: ";
    if (indexClose != -1)
    {
        this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
    }
    strParam = "<PROVINCE_ABBR>";
    indexOpen = p_result.IndexOf(strParam) + strParam.Length;
    indexClose = p_result.IndexOf("</PROVINCE_ABBR>");
    this.txtResult.Text += "\rPROVINCE ABBR: ";
    if (indexClose != -1)
    {
        this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
    }
    strParam = "<AREA_CODE>";
    indexOpen = p_result.IndexOf(strParam) + strParam.Length;
    indexClose = p_result.IndexOf("</AREA_CODE>");
    this.txtResult.Text += "\rAREA CODE: ";
    if (indexClose != -1)
    {
        this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
    }
    strParam = "<CITY_FLAG>";
    indexOpen = p_result.IndexOf(strParam) + strParam.Length;
    indexClose = p_result.IndexOf("</CITY_FLAG>");
    this.txtResult.Text += "\rCITY FLAG: ";
    if (indexClose != -1)
    {
        this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
```

```
indexOpen);
}
strParam = "<LATITUDE>";
indexOpen = p_result.IndexOf(strParam) + strParam.Length;
indexClose = p_result.IndexOf("</LATITUDE>");
this.txtResult.Text += "\rLatitude: ";
if (indexClose != -1)
{
    this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
}
strParam = "<LONGITUDE>";
indexOpen = p_result.IndexOf(strParam) + strParam.Length;
indexClose = p_result.IndexOf("</LONGITUDE>");
this.txtResult.Text += "\rLongitude: ";
if (indexClose != -1)
{
    this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
}
strParam = "<TIME_ZONE>";
indexOpen = p_result.IndexOf(strParam) + strParam.Length;
indexClose = p_result.IndexOf("</TIME_ZONE>");
this.txtResult.Text += "\rTime Zone: ";
if (indexClose != -1)
{
    this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
}
strParam = "<DAY_LIGHT_SAVING>";
indexOpen = p_result.IndexOf(strParam) + strParam.Length;
indexClose = p_result.IndexOf("</DAY_LIGHT_SAVING>");
this.txtResult.Text += "\rDay Light Saving: ";
if (indexClose != -1)
{
    this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
}
strParam = "<ELEVATION>";
indexOpen = p_result.IndexOf(strParam) + strParam.Length;
indexClose = p_result.IndexOf("</ELEVATION>");
this.txtResult.Text += "\rELEVATION: ";
if (indexClose != -1)
{
    this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
}
strParam = "<POPULATION>";
indexOpen = p_result.IndexOf(strParam) + strParam.Length;
indexClose = p_result.IndexOf("</POPULATION>");
this.txtResult.Text += "\rPOPULATION: ";
if (indexClose != -1)
{
    this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
}
strParam = "<AREA_NAME>";
```

```
indexOpen = p_result.IndexOf(strParam) + strParam.Length;
indexClose = p_result.IndexOf("</AREA_NAME>");
this.txtResult.Text += "\rAREA NAME: ";
if (indexClose != -1)
{
    this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
}
strParam = "<DWELLING>";
indexOpen = p_result.IndexOf(strParam) + strParam.Length;
indexClose = p_result.IndexOf("</DWELLING>");
this.txtResult.Text += "\rDWELLING: ";
if (indexClose != -1)
{
    this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
}
strParam = "<STREET_NAME>";
indexOpen = p_result.IndexOf(strParam) + strParam.Length;
indexClose = p_result.IndexOf("</STREET_NAME>");
this.txtResult.Text += "\rSTREET NAME: ";
if (indexClose != -1)
{
    this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
}
strParam = "<STREET_TYPE_CODE>";
indexOpen = p_result.IndexOf(strParam) + strParam.Length;
indexClose = p_result.IndexOf("</STREET_TYPE_CODE>");
this.txtResult.Text += "\rSTREET TYPE CODE: ";
if (indexClose != -1)
{
    this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
}
strParam = "<STREET_DIR_CODE>";
indexOpen = p_result.IndexOf(strParam) + strParam.Length;
indexClose = p_result.IndexOf("</STREET_DIR_CODE>");
this.txtResult.Text += "\rSTREET DIR CODE: ";
if (indexClose != -1)
{
    this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
}
strParam = "<STREET_SEQ_CODE>";
indexOpen = p_result.IndexOf(strParam) + strParam.Length;
indexClose = p_result.IndexOf("</STREET_SEQ_CODE>");
this.txtResult.Text += "\rSTREET SEQ CODE: ";
if (indexClose != -1)
{
    this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
}
strParam = "<STREET_FROM_NO>";
indexOpen = p_result.IndexOf(strParam) + strParam.Length;
indexClose = p_result.IndexOf("</STREET_FROM_NO>");
this.txtResult.Text += "\rSTREET FROM NO: ";
```

```

if (indexClose != -1)
{
    this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
}
strParam = "<STREET_FROM_SUFFIX>";
indexOpen = p_result.IndexOf(strParam) + strParam.Length;
indexClose = p_result.IndexOf("</STREET_FROM_SUFFIX>");
this.txtResult.Text += "\rSTREET FROM SUFFIX: ";
if (indexClose != -1)
{
    this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
}
strParam = "<STREET_TO_NO >";
indexOpen = p_result.IndexOf(strParam) + strParam.Length;
indexClose = p_result.IndexOf("</STREET_TO_NO >");
this.txtResult.Text += "\rSTREET TO NO: ";
if (indexClose != -1)
{
    this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
}
strParam = "<STREET_TO_SUFFIX>";
indexOpen = p_result.IndexOf(strParam) + strParam.Length;
indexClose = p_result.IndexOf("</STREET_TO_SUFFIX>");
this.txtResult.Text += "\rSTREET TO SUFFIX: ";
if (indexClose != -1)
{
    this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
}
strParam = "<CREDITSAVAILABLE>";
indexOpen = p_result.IndexOf(strParam) + strParam.Length;
indexClose = p_result.IndexOf("</CREDITSAVAILABLE>");
this.txtResult.Text += "\rCredits Available:";
if (indexClose != -1)
{
    this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
}
strParam = "<MESSAGE>";
indexOpen = p_result.IndexOf(strParam) + strParam.Length;
indexClose = p_result.IndexOf("</MESSAGE>");
this.txtResult.Text += "\rMessage: ";
if (indexClose != -1)
{
    this.txtResult.Text += p_result.Substring(indexOpen, indexClose -
indexOpen);
}
}
}

```

## v. ASP 3.0

```
Sub sendRequest(strPostalCode, strLICENSE)
```

```
dim SoapBody
if((strPostalCode <> "") AND (strLICENSE <> "") AND (strLICENSE <>
"<Your License Key>")) then
SoapBody = xmlSoap(strPostalCode, strLICENSE)
end if
%>
<table border="0" cellspacing="1" cellpadding="3" rules="rows">
<%
if SoapBody = "" then
%><tr><td><b>Empty Soap Body Response</b></td></tr><%
else
dim xml
on error resume next
set xml = Server.CreateObject("Microsoft.XMLDOM")
if(err.number = 0) then
xml.async = False
xml.loadxml(SoapBody)
dim oNode : set oNode = xml.selectSingleNode("soap:Envelope/soap:Body/"
& SoapAction & "Response/" & SoapAction & "Result")
if
(oNode.selectSingleNode("Error").nodeTypedValue = "") then
%>
<tr>
<th colspan="2" align="left"><b>Result</b></th>
</tr>
<% if(TypeName(oNode.selectSingleNode("POSTAL_CODE")) =
"IXMLDOMEElement") then %>
<tr>
<td align="left">Postal Code: </td>
<td><%=oNode.selectSingleNode("POSTAL_CODE").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("CITY")) = "IXMLDOMEElement") then
%>
<tr>
<td align="left">City: </td>
<td><%=oNode.selectSingleNode("CITY").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("PROVINCE")) = "IXMLDOMEElement")
then %>
<tr>
<td align="left">Province: </td>
<td><%=oNode.selectSingleNode("PROVINCE").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("PROVINCE_ABBR")) =
"IXMLDOMEElement") then %>
<tr>
<td align="left">Province ABBR </td>
<td><%=oNode.selectSingleNode("PROVINCE_ABBR").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("CITY_FLAG")) = "IXMLDOMEElement")
then %>
<tr>
<td align="left">City Flag: </td>
```

```
<td><%=oNode.selectSingleNode("CITY_FLAG").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("LATITUDE")) = "IXMLDOMEElement")
then %>
<tr>
<td align="left">Latitude: </td>
<td><%=oNode.selectSingleNode("LATITUDE").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("LONGITUDE")) = "IXMLDOMEElement")
then %>
<tr>
<td align="left">Longitude: </td>
<td><%=oNode.selectSingleNode("LONGITUDE").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("TIME_ZONE")) = "IXMLDOMEElement")
then %>
<tr>
<td align="left">Time Zone: </td>
<td><%=oNode.selectSingleNode("TIME_ZONE").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("DAY_LIGHT_SAVING")) =
"IXMLDOMEElement") then %>
<tr>
<td align="left">Day Light Saving: </td>
<td><%=oNode.selectSingleNode("DAY_LIGHT_SAVING").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("AREA_NAME")) = "IXMLDOMEElement")
then %>
<tr>
<td align="left">Area Name: </td>
<td><%=oNode.selectSingleNode("AREA_NAME").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("AREA_CODE")) = "IXMLDOMEElement")
then %>
<tr>
<td align="left">Area Code: </td>
<td><%=oNode.selectSingleNode("AREA_CODE").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("ELEVATION")) = "IXMLDOMEElement")
then %>
<tr>
<td align="left">Elevation: </td>
<td><%=oNode.selectSingleNode("ELEVATION").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("POPULATION")) =
"IXMLDOMEElement") then %>
<tr>
<td align="left">Population: </td>
<td><%=oNode.selectSingleNode("POPULATION").nodeTypedValue%></td>
```

```
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("DWELLING")) = "IXMLDOMEElement")
then %>
<tr>
<td align="left">Dwelling: </td>
<td><%=oNode.selectSingleNode("DWELLING").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("AREA_NAME")) = "IXMLDOMEElement")
then %>
<tr>
<td align="left">Area Name: </td>
<td><%=oNode.selectSingleNode("AREA_NAME").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("AREA_CODE")) = "IXMLDOMEElement")
then %>
<tr>
<td align="left">Area Code: </td>
<td><%=oNode.selectSingleNode("AREA_CODE").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("STREET_NAME")) =
"IXMLDOMEElement") then %>
<tr>
<td align="left">Street Name: </td>
<td><%=oNode.selectSingleNode("STREET_NAME").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("STREET_TYPE_CODE")) =
"IXMLDOMEElement") then %>
<tr>
<td align="left">Street Type Code: </td>
<td><%=oNode.selectSingleNode("STREET_TYPE_CODE").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("STREET_DIR_CODE")) =
"IXMLDOMEElement") then %>
<tr>
<td align="left">Street Dir Code: </td>
<td><%=oNode.selectSingleNode("STREET_DIR_CODE").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("STREET_SEQ_CODE")) =
"IXMLDOMEElement") then %>
<tr>
<td align="left">Street Seq Code: </td>
<td><%=oNode.selectSingleNode("STREET_SEQ_CODE").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("STREET_FROM_NO")) =
"IXMLDOMEElement") then %>
<tr>
<td align="left">Street Form No: </td>
<td><%=oNode.selectSingleNode("STREET_FROM_NO").nodeTypedValue%></td>
</tr>
```

```
<% end if %>
<% if(TypeName(oNode.selectSingleNode("STREET_FROM_SUFFIX")) =
"IXMLDOMEElement") then %>
<tr>
<td align="left">STREET From Suffix: </td>
<td><%=oNode.selectSingleNode("STREET_FROM_SUFFIX").nodeTypedValue%></t
d>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("STREET_TO_NO")) =
"IXMLDOMEElement") then %>
<tr>
<td align="left">Street To No: </td>
<td><%=oNode.selectSingleNode("STREET_TO_NO").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("STREET_TO_SUFFIX")) =
"IXMLDOMEElement") then %>
<tr>
<td align="left">STREET To Suffix: </td>
<td><%=oNode.selectSingleNode("STREET_TO_SUFFIX").nodeTypedValue%></td>
</tr>
<% end if %>
<% if(TypeName(oNode.selectSingleNode("CREDITSAVAILABLE")) =
"IXMLDOMEElement") then %>
<tr>
<td align="left">Credits Available: </td>
<td><%=oNode.selectSingleNode("CREDITSAVAILABLE").nodeTypedValue%></td>
</tr>
<% end if %>

<% if(TypeName(oNode.selectSingleNode("MESSAGE")) = "IXMLDOMEElement")
then %>
<tr>
<td align="left">Message: </td>
<td><%=oNode.selectSingleNode("MESSAGE").nodeTypedValue%></td>
</tr>
<% end if %>
<%
else ' error element
%>
<tr>
<td align="left" colspan="2"><b>Error</b></td>
</tr>
<tr>
<td align="left">Description:</td>
<td><%=oNode.selectSingleNode("MESSAGE").nodeTypedValue%></td>
</tr>
<%
end if 'error element does not exist
set xml = nothing
else
%>
<tr>
```

```
<td align="left">This objects requires Microsofts XML Parser 3.0 SP2 or
greater.</td>
</tr>
<tr>
<td>Download here: <a
href="http://download.microsoft.com/download/xml/SP/3.20/W9X2KMeXP/EN-
US/msxml3sp2Setup.exe">http://download.microsoft.com/download/xml/SP/3.
20/W9X2KMeXP/EN-US/msxml3sp2Setup.exe</a></td>
</tr>
<%
Response.Write("Error: " & err.number)
end if 'DOM object is valid
end if 'soapbody is <> "" empty string
%>
</table>
<%
end sub

function xmlSoap(strPostalCode, strLICENSE)
' Instantiate objects to hold the XML DOM and the HTTP/XML
communication:
' Instantiate object for HTTP/XML communication:
Dim xmlhttp, strSoap
Set xmlhttp = Server.CreateObject("Msxml2.ServerXMLHTTP")

' Build XML document:
strSoap = "<?xml version=""1.0"" encoding=""utf-8""?>" & _
"<soap:Envelope xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
xmlns:xsd=""http://www.w3.org/2001/XMLSchema""
xmlns:soap=""http://schemas.xmlsoap.org/soap/envelope/">" & _
"<soap:Body>" & _
"<" & SoapAction & " xmlns=""& SoapNamespace & "">" & _
"<PostalCode>" & strPostalCode & "</PostalCode>" & _
"<LICENSE>" & strLICENSE & "</LICENSE>" & _
"</" & SoapAction & ">" & _
"</soap:Body>" & _
"</soap:Envelope>"
'Build custom HTTP header:
xmlhttp.Open "POST", "http://" & SoapServer & SoapPath, False
False = Do not respond immediately
xmlhttp.setRequestHeader "Man", "POST " & SoapPath & " HTTP/1.1"
xmlhttp.setRequestHeader "Host", SoapServer
xmlhttp.setRequestHeader "Content-Type", "text/xml; charset=utf-8"
xmlhttp.setRequestHeader "SOAPAction", SoapNamespace & SoapAction
'Send it using the header generated above:
xmlhttp.send(strSoap)
if xmlhttp.Status = 200 then ' Response from server was success
xmlSoap = xmlhttp.responseText ' note xmlSoap is the function name
hence the return value is a string Variant
else ' Response from server failed
xmlSoap = ""
' Tell administrator what went wrong - maybe not users though
Response.Write("Server Error.<br>")
Response.Write("status = " & xmlhttp.status)
Response.Write("<br>" & xmlhttp.statusText & "<br>" &
xmlhttp.responseText)
Response.Write("<br><pre>" & Request.ServerVariables("ALL_HTTP") &
```

```

"</pre>")
end If
Set xmlhttp = nothing
end function
%>

```

## vi. PHP

```

<?php
if (!isset($_POST['submit'])) {} // if page is not submitted to itself
echo the form
else
{
$PostalCode = $_POST["PostalCode"];
$License = $_POST["License"];

if ($License == "<Enter License Key>" || $License == "")
{
    echo "License key are required field." ;
}

else
{
require_once('./lib/nusoap.php');
// Call the service with this message
$msg = '<PostalCodeWorld_Canada xmlns="http://ws.fraudlabs.com/">
<PostalCode>'.$PostalCode.'</PostalCode>
<LICENSE>'.$License.'</LICENSE>
</PostalCodeWorld_Canada>';

// Create a new SOAP object
$soapclient = new
soapclient_nusoap('http://ws.fraudlabs.com/PostalCodeWorldCanada_WebSer
vice.asmx?wsdl','wsdl');
if (!$soapclient->fault) {
$result = $soapclient->call('PostalCodeWorld_Canada', array($msg));
if(!isset($result['Error'])) {
// No Error. Retrieve results.
echo "POSTALCODE = " . $result["POSTAL_CODE"] . "<br>";
echo "CITY = " . $result["CITY"] . "<br>";
echo "PROVINCE = " . $result["PROVINCE"] . "<br>";
echo "PROVINCE ABBR = " . $result["PROVINCE_ABBR"] . "<br>";
echo "LATITUDE = " . $result["LATITUDE"] . "<br>";
echo "LONGITUDE = " . $result["LONGITUDE"] . "<br>";
echo "TIME_ZONE = " . $result["TIME_ZONE"] . "<br>";
echo "DAY LIGHT SAVING = " . $result["DAY_LIGHT_SAVING"] . "<br>";
echo "CITY FLAG = " . $result["CITY_FLAG"] . "<br>";
echo "ELEVATION = " . $result["ELEVATION"] . "<br>";
echo "POPULATION = " . $result["POPULATION"] . "<br>";
echo "AREA CODE = " . $result["AREA_CODE"] . "<br>";
echo "AREA NAME = " . $result["AREA_NAME"] . "<br>";
echo "DWELLING = " . $result["DWELLING"] . "<br>";
echo "STREET NAME = " . $result["STREET_NAME"] . "<br>";
echo "STREET TYPE CODE = " . $result["STREET_TYPE_CODE"] . "<br>";
echo "STREET DIR CODE = " . $result["STREET_DIR_CODE"] . "<br>";

```

```

echo "STREET SEQ CODE = " . $result["STREET_SEQ_CODE"] . "<br>";
echo "STREET FROM NO = " . $result["STREET_FROM_NO"] . "<br>";
echo "STREET FROM SUFFIX = " . $result["STREET_FROM_SUFFIX"] . "<br>";
echo "STREET TO NO = " . $result["STREET_TO_NO"] . "<br>";
echo "STREET TO SUFFIX = " . $result["STREET_TO_SUFFIX"] . "<br>";
echo "CREDITSAVAILABLE = " . $result["CREDITSAVAILABLE"] . "<br>";
echo "MESSAGE = " . $result["MESSAGE"] . "<br>";
}
else {
// Error
echo "Error Description = " . $result["Error"]["Desc"] . "<br>";
echo "Error Number = " . $result["Error"]["Number"] . "<br>";
}
}
else {
echo "Error = {$soapclient->faultcode}<br>";
echo "String = {$soapclient->faultstring}";
} // end if $soapclient->call is success
}
}
?>

```

## vii. JAVA / APACHE

```

import java.io.*;

//Web Service Client Class
public class Client
{
//Entry Point to this Application
public static void main(String[] args)
{
try
{
//Create a proxy
ApacheSoapProxy proxy = new ApacheSoapProxy ();

String result = proxy.PostalCodeWorld_Canada("A0A 1A0", "<Enter
License Key>");

//Print out the value
System.out.println (result);
}
catch (java.net.MalformedURLException exception)
{
exception.printStackTrace ();
}
catch (org.apache.soap.SOAPException exception)
{
exception.printStackTrace ();
}
}
}
}

```

