Introduction
The address correction service accepts a street address and returns a corrected and standardized address. This SOAP based XML service is intended to be used by any application that collects mailing or street addresses. The service corrects misspelled streets, missing components, invalid zip codes and city names. Up to five of the top candidates can also be returned using the address and city for the search. Software and data from MelissaData Inc. is updated quarterly and provides access to the USPS database for validating addresses. By using this service mailing return rates are reduced, the quality of data analysis is improved, geocoding match rates are higher, and storage of addresses in a database is enhanced.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>101 Isreal Rd</td>
<td>Address to verify with house number and all components. (required)</td>
</tr>
<tr>
<td>Address2</td>
<td>p.o. box 47904</td>
<td>Second address line or suite type additional information. (optional)</td>
</tr>
<tr>
<td>City</td>
<td>Tumwat</td>
<td>City Name (optional if zip is supplied)</td>
</tr>
<tr>
<td>Company</td>
<td>WA DOH</td>
<td>Company name, used for verifying addresses only. (optional)</td>
</tr>
<tr>
<td>Zip</td>
<td>98502</td>
<td>5 Digit Zip code (optional if city is supplied)</td>
</tr>
<tr>
<td>State</td>
<td>WA</td>
<td>2 Digit State Abbreviation (optional)</td>
</tr>
<tr>
<td>ZipPlus4</td>
<td>9656</td>
<td>Zip Plus four code (optional)</td>
</tr>
<tr>
<td>Consumer</td>
<td>LEG</td>
<td>Unique key for each consumer (required)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outputs</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeliveryAddress</td>
<td>101 Israel Rd SE</td>
<td>Corrected Address</td>
</tr>
<tr>
<td>DeliveryAddress2</td>
<td>PO Box 47904</td>
<td>Corrected Address line two</td>
</tr>
<tr>
<td>PlaceName</td>
<td>Tumwater</td>
<td>Corrected City Name</td>
</tr>
<tr>
<td>ZipCode</td>
<td>98501</td>
<td>Corrected 5-digit Zip Code</td>
</tr>
<tr>
<td>ZipPlus4</td>
<td>7904</td>
<td>Corrected Zip Plus 4 if found in USPS database</td>
</tr>
<tr>
<td>StateName</td>
<td>WA</td>
<td>Corrected State Abbreviation</td>
</tr>
<tr>
<td>County</td>
<td>Thurston</td>
<td>County the address is located in</td>
</tr>
<tr>
<td>ANSIStateCountyCode</td>
<td>053067</td>
<td>State and County FIPS Code</td>
</tr>
<tr>
<td>Company</td>
<td>WA DOH</td>
<td>Not Correct only used for verification of addresses.</td>
</tr>
<tr>
<td>OfficialStatus</td>
<td>Street Address</td>
<td>Type of Address (see possible values on next page)</td>
</tr>
<tr>
<td>StreetNamePreDirectional</td>
<td>N</td>
<td>Corrected Pre street direction</td>
</tr>
<tr>
<td>AddressNumber</td>
<td>101</td>
<td>Parsed house number of the input address</td>
</tr>
<tr>
<td>StreetName</td>
<td>Israel</td>
<td>Corrected street name of the input address</td>
</tr>
<tr>
<td>StreetNamePostType</td>
<td>Loop</td>
<td>Corrected Pre street type</td>
</tr>
<tr>
<td>StreetNamePostDirectional</td>
<td>SE</td>
<td>Corrected Post street direction</td>
</tr>
<tr>
<td>SubaddressIdentifier</td>
<td>14</td>
<td>Corrected Unit identifier</td>
</tr>
<tr>
<td>SubaddressType</td>
<td>TRLR</td>
<td>Corrected Unit type</td>
</tr>
<tr>
<td>MailableAddress</td>
<td>Yes or No</td>
<td>Was the address found in the USPS database</td>
</tr>
<tr>
<td>Quality</td>
<td>Address found in USPS database</td>
<td>Could also contain “Parsed but not found”</td>
</tr>
<tr>
<td>Results</td>
<td>AC01, AC03, AS01</td>
<td>Changes made to the input data, see Result Codes below</td>
</tr>
<tr>
<td>ErrorStatus</td>
<td>Invalid input</td>
<td>Any application error codes</td>
</tr>
<tr>
<td>Candidate1</td>
<td>101 Israel Rd SW, 98501</td>
<td>Address Candidate</td>
</tr>
<tr>
<td>Candidate2</td>
<td>101 Israel ST, 98502</td>
<td>Address Candidate</td>
</tr>
<tr>
<td>Candidate3</td>
<td></td>
<td>Address Candidate</td>
</tr>
<tr>
<td>Candidate4</td>
<td></td>
<td>Address Candidate</td>
</tr>
<tr>
<td>Candidate5</td>
<td></td>
<td>Address Candidate</td>
</tr>
<tr>
<td>Garbage</td>
<td>saddfad</td>
<td>Leftover characters after correction and parsing occurs.</td>
</tr>
<tr>
<td>MDBexpries</td>
<td>6/30/2017</td>
<td>Date the data will revert to Demo Mode if patch is not supplied.</td>
</tr>
<tr>
<td>MDRBDI</td>
<td>Residential</td>
<td>MelissaData Residential/Business Indicator</td>
</tr>
<tr>
<td>MDdatabaseExpire</td>
<td>3/15/2017</td>
<td>Date of USPS database from MelissaData</td>
</tr>
<tr>
<td>MDlicenseExpires</td>
<td>7/5/2017</td>
<td>Date the MelissaData annual license expires</td>
</tr>
</tbody>
</table>
Result Codes
AS01 Address Matched to Postal Database  AE01 Zip Unknown
AS02 Street Address Match  AE02 Unknown Street
AS03 Non-USPS Address  AE03 Component Mismatch
AS09 Foreign Zip Code  AE04 Non-Deliverable
AS10 UPS Store  AE05 Multiple Matches
AS13 LACS Conversion  AE06 EWS
AS14 Suite Appended Link  AE07 Invalid Input
AS15 Suite Appended by AP  AE08 Invalid Suite
AS16 Address is vacant  AE09 Missing Suite
AS17 Alternate Delivery  AE10 Invalid Range
AS18 DPV Error  AE11 Missing Range
AS20 No UPS or FedEx Delivery  AE12 Invalid PO, HC or RR
AS22 No Suggestions  AE13 Missing PO, HC or RR
AS23 Extra Info found  AE14 CMRA Missing
AC10 Changed the Street Name  AE15 Demo Mode
AC01 Changed the Zip Code  AE16 Expired Database
AC08 Changed Zip4  AE17 in Suite Range
AC02 Changed the State  AE19 Timed Out
AC03 Changed the City  AE20 Suggestions Disabled
AC11 Changed Suffix  AC07 Changed Company
AC12 Changed Directional  AC08 Changed Zip4
AC04 Changed to Alt. Name  AC09 Changed Urbanization
AC05 Changed Alias  AC13 Changed Suite Name
AC06 Changed Address2  AC14 Changed Suite Range

Address Types
Alias
Firm or Company address
General Delivery address
High Rise or Business complex
PO Box address
Rural Route address
Street or Residential address

Consumption Example
This assumes your Visual Studio 2010 web reference is named "Correction" and VB.NET is the language.

Dim addr_obj As New Correction.Service
Dim addr_corr As New Correction.Address_Correction
addr_obj.Url = "https://fortress..."  (Contact DOH for URL)
addr_corr = addr_obj.Getstandardizedaddress("101 is real rd se","po box 47904","Tumwat","WA DOH","98502","WA","5570","LEG")

The addr_corr object will have the input and output values as properties.

Notes
Arguments for the getstandardizedaddress method are "address","address2","city","company","zipcode","state","zip4","consumer".
Either a city or zip is required and consumer.
Zip should be in the 5 digit format.
State should be in the 2 digit format and defaults to "WA" if not specified.
Transmission of the address is in clear text without encryption inside the State Government Network
If address line 1 is not found then address line 2 is used. Addresses will be switched in the output. Address line 1 will always contain the address used.
ASP.NET Test Page
Enter an address into this page and it calls three Web services and one mapping application.

Getting Started
Contact gis.dirm@doh.wa.gov for the Fortress production URL and a unique key.
(360)236-4271

Revisions
8/7/2013
Added the MelissaData Unique Address Key to the output XML Schema.

9/30/2013
Added Address2 and Company as inputs and outputs.
Added Garbage and AddressType to the output XML.
XML Schema has been modified.

12/15/2014
Added Consumer to the required input. Users should standardize values and at least provide the agency.
Renamed output XML elements to align with FGDC standards.
Added the MDexpires output element that contains the date the MelissaData will time out if the quarterly update is not applied.
Added the RBDI output element that contains the MelissaData residential/business indicator.
Removed the Addresskey element from the output.

4/25/2017
Added the ability to verify the consumer’s key that gets passed in. This will help to limit access.
Introduction
The geocoding service accepts a street address and returns location information. Addresses are matched against various sources including the Master Address File (MAF), County parcels and street centerlines. This service geocodes to zip code, post office, city and zip+4 if a street level match isn’t possible. The latitude, longitude, accuracy and source values are also returned. This service uses the ArcGIS Server software from Environmental Systems Research Institute to perform the geocoding.

Inputs
<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>101 Israel Rd SE Address to geocode with house number and all components.   (optional)</td>
</tr>
<tr>
<td>Zip</td>
<td>98501 5 Digit Zip Code                                                     (optional)</td>
</tr>
<tr>
<td>City</td>
<td>Tumwater City Name                                                         (optional)</td>
</tr>
<tr>
<td>Zip4</td>
<td>5570 Zip Plus 4                                                            (optional)</td>
</tr>
<tr>
<td>Consumer</td>
<td>DOH Unique key for each consumer                                           (required)</td>
</tr>
</tbody>
</table>

Outputs
<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Matched (M) or Unmatched (U)</td>
</tr>
<tr>
<td>Score</td>
<td>Numeric score returned from ESRI’s address locator</td>
</tr>
<tr>
<td>Source</td>
<td>MAF Source data used for geocoding</td>
</tr>
<tr>
<td>Locator</td>
<td>URL for the REST endpoint of the address locator</td>
</tr>
<tr>
<td>Accuracy</td>
<td>Close, Approximate or Very Approximate</td>
</tr>
<tr>
<td>Longitude</td>
<td>WGS84 in decimal degrees</td>
</tr>
<tr>
<td>Latitude</td>
<td>WGS84 in decimal degrees</td>
</tr>
<tr>
<td>Av_date</td>
<td>Date and time geocoding was attempted</td>
</tr>
<tr>
<td>Error_status</td>
<td>Any application messages</td>
</tr>
<tr>
<td>Matched_address</td>
<td>101 Israel Rd SE, 98501 Matched address returned from the ESRI address locator used</td>
</tr>
<tr>
<td>Candidate1 – 5</td>
<td>101 Israel Rd SW, 98501 1 – 5 candidates if found</td>
</tr>
</tbody>
</table>

Consumption Example
This assumes your Visual Studio 2010 web reference is named “Geocoder” and VB.NET is the language.

```vbnet
Dim geo_obj As New Geocoder.Geocoder
Dim geo As New Geocoder.Geocode
geo_obj.Url = "https://fortress..." ' (Contact DOH for URL)
geo = geo_obj.FindAddress("101 Israel Rd SE", "98501", "Tumwater", "DOE")
The geo object will have the input and output values as properties.
```

Locators Used
<table>
<thead>
<tr>
<th>Reference Data</th>
<th>Status</th>
<th>Score</th>
<th>Accuracy</th>
<th>~Horizontal Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAF Address Points</td>
<td>M</td>
<td>100 - 80</td>
<td>High</td>
<td>+ - 100 ft.</td>
</tr>
<tr>
<td>NavTeq Points</td>
<td>M</td>
<td>100 - 80</td>
<td>High</td>
<td>+ - 100 ft.</td>
</tr>
<tr>
<td>County Parcel Points</td>
<td>M</td>
<td>100 - 80</td>
<td>Medium</td>
<td>+ - 300 ft.</td>
</tr>
<tr>
<td>NavTeq Streets</td>
<td>M</td>
<td>100 - 80</td>
<td>Medium</td>
<td>+ - 300 ft.</td>
</tr>
<tr>
<td>TIGER 2010 Streets</td>
<td>M</td>
<td>100 - 80</td>
<td>Medium</td>
<td>+ - 300 ft.</td>
</tr>
<tr>
<td>TIGER 2006 Streets</td>
<td>M</td>
<td>100 - 80</td>
<td>Medium</td>
<td>+ - 300 ft.</td>
</tr>
<tr>
<td>Zip4 Points</td>
<td>M</td>
<td>60</td>
<td>Low</td>
<td>+ - 1000 ft.</td>
</tr>
<tr>
<td>NavTeq Street Points</td>
<td>M</td>
<td>50</td>
<td>Low</td>
<td>+ - 1000 ft.</td>
</tr>
<tr>
<td>GNIS Points</td>
<td>U</td>
<td>40</td>
<td>Low</td>
<td>+ - 5 Miles</td>
</tr>
<tr>
<td>Zip 5 Points</td>
<td>U</td>
<td>30</td>
<td>Low</td>
<td>+ - 5 miles</td>
</tr>
<tr>
<td>Post Office Points</td>
<td>U</td>
<td>30</td>
<td>Low</td>
<td>+ - 10 Miles</td>
</tr>
</tbody>
</table>

Horizontal Accuracy
The horizontal accuracy values in the table above represent a general guideline. Horizontal accuracy is generally higher in urban areas and less accurate in rural areas. Based on the reference data used, the position of a
geocode can range from exact to miles from the correct position. The accuracy is also relative to the target location of an address (e.g. mailbox, condo, driveway, rooftop of structure). For geocodes using parcel points (centroids) the size and shape of the parcel has a direct impact on the accuracy.

Search Order
Matching continues until a suitable match is found (score above 80 for point or street matches).

(1) Post Offices

(2) Master Address File (MAF)
Washington State Department of Health
Geocoder Web Service

(3) Navteq Address Points, (4) County Parcel Points

(5) Navteq Streets, (6) TIGER 2010 Streets, (7) TIGER 2006 Streets, (8) TIGER 2006 Streets
(11) City Points

(12) 5 Digit Zip Code Points
ASP.NET Test Page
Enter an address into this page and it calls five Web services and one mapping application.

Getting Started
Contact gis.dirm@doh.wa.gov for the Fortress production URL and a unique key.
(360)236-4271

Revisions
8/7/2013
Added input parameter for zip+4.
Added input parameter for the MelissaData Unique AddressKey from the Correction Web service.
Added the MelissaData Geocoder/GeoPoint matches for rooftops and streets.

12/15/2014
Removed the Addresskey output XML element.
Added the NavTeq Zip+4 locator service.
Added Consumer to the required input. Users should standardize values and at least provide the agency.

4/25/2017
Added the ability to verify the consumer’s key that gets passed in. This will help to limit access.
Identify Example

This example describes how to leverage the REST endpoint for the LocationFinder_GPService service, found at: http://geoservices.wa.gov/ArcGIS/rest/services/MapServices.

The list of data layers available to identify can be found by viewing the ArcGIS Service Directory for the service: http://geoservices.wa.gov/ArcGIS/rest/services/MapServices/LocationFinder_GPService/MapServer.

Below is an example of performing an identify task using the ArcGIS JavaScript API. Detailed explanation of the ArcGIS JavaScript API can be viewed at: http://help.arcgis.com/en/webapi/javascript/arcgis/jsapi/#overview_jsapi_reference

Example – Identify City and County Name:

```html
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=7,IE=9" />
<!--The viewport meta tag is used to improve the presentation and behavior of the samples on iOS devices-->
<meta name="viewport" content="initial-scale=1, maximum-scale=1, user-scalable=no" />
<title>Identify Sample</title>
<link rel="stylesheet" type="text/css" href="http://serverapi.arcgisonline.com/jsapi/arcgis/2.8/js/dojo/dijit/themes/tundra/tundra.css" />
<style type="text/css">
html, body
{
    height: 98%;
    width: 98%;
    margin: 0;
    padding: 5px;
}
</style>
<script type="text/javascript">var djConfig = { parseOnLoad: true };</script>
<script type="text/javascript" src="http://serverapi.arcgisonline.com/jsapi/arcgis/2.8/"/></script>
<script type="text/javascript">
    dojo.require("esri.map");
    dojo.require("esri.tasks.identify");
    dojo.require("dijit.layout.ContentPane");
    dojo.require("dijit.layout.TabControl");
    dojo.require("dijit.form.Button");

    var map, identifyTask, identifyParams, symbol;
    var cityResults, countyResults;

    function init() {
        var initExtent = new esri.geometry.Extent(-14221820, 5512725, -12682073, 6459321, new esri.SpatialReference({ "wkid": 102113 }));
        map = new esri.Map("mapDiv", { extent: initExtent, lods: lods });
        dojo.connect(map, "onLoad", initFunctionality);
```
function initFunctionality(map) {
  dojo.connect(map, "onClick", doIdentify);
  identifyTask = new esri.tasks.IdentifyTask("http://198.238.214.59/ArcGIS/rest/services/MapServices/LocationFinder_GPService/MapServer");
  identifyParams = new esri.tasks.IdentifyParameters();
  identifyParams.tolerance = 1;
  identifyParams.returnGeometry = false;
  identifyParams.layerIds = [4, 6]; // city/uga(4) and county(6) layers used for this example
  identifyParams.layerOption = esri.tasks.IdentifyParameters.LAYER_OPTION_ALL;
  identifyParams.width = map.width;
  identifyParams.height = map.height;
  map.infoWindow.resize(415, 200);
  map.infoWindow.setContent(dijit.byId("tabs").domNode);
  map.infoWindow.setTitle("Identify Results");
  symbol = new esri.symbol.SimpleFillSymbol(esri.symbol.SimpleFillSymbol.STYLE_SOLID, new esri.symbol.SimpleLineSymbol(esri.symbol.SimpleLineSymbol.STYLE_SOLID, new dojo.Color([255, 0, 0]), 2), new dojo.Color([255, 255, 0, 0.25]));
}

function doIdentify(evt) {
  map.graphics.clear();
  identifyParams.geometry = evt.mapPoint;
  identifyParams.mapExtent = map.extent;
  identifyTask.execute(identifyParams, function (idResults) { addToMap(idResults, evt); });
}

function addToMap(idResults, evt) {
  cityResults = { displayFieldName: null, features: []);
  countyResults = { displayFieldName: null, features: []});
  for (var i = 0, il = idResults.length; i < il; i++) {
    var idResult = idResults[i];
    if (idResult.layerId === 4) {
      if (!cityResults.displayFieldName) {
        cityResults.displayFieldName = idResult.displayFieldName
      }
      cityResults.features.push(idResult.feature);
    } else if (idResult.layerId === 6) {
      if (!countyResults.displayFieldName) {
        countyResults.displayFieldName = idResult.displayFieldName
      }
      countyResults.features.push(idResult.feature);
    }
  }
  dijit.byId("cityTab").setContent(layerTabContent(cityResults, "cityResults"));
  dijit.byId("countyTab").setContent(layerTabContent(countyResults, "countyResults"));
  map.infoWindow.show(evt.screenPoint, map.getInfoWindowAnchor(evt.screenPoint));
}

function layerTabContent(layerResults, layerName) {
  var content = "";
  switch (layerName) {
    case "cityResults": content += "Total features returned: " + layerResults.features.length + "<i>/i>";
      content += "<table border="1">";
      for (var i = 0, il = layerResults.features.length; i < il; i++) {
        content += "<tr><td>City Name</td><th>City Name</th>";
        for (var i = 0, il = layerResults.features.length; i < il; i++) {
          content += "<td>" + layerResults.features[i].attributes["CITY_NM"] + "</td>";
        }
      }
      content += "</tr></table>";
      break;
    case "countyResults": content += "Total features returned: " + layerResults.features.length + "<i>/i>";
content += "<table border='1'><tr><th>County Name</th>";
for (var i = 0, il = layerResults.features.length; i < il; i++) {
  content += "<tr><td>" + layerResults.features[i].attributes['COUNTY_NM'] + "</td>";
}
content += "</tr></table>";
break);

return content;

//show map on load
dojo.addOnLoad(init);
</script>
</head>
<body class="tundra">
  Click the map to identify city and county information.
  <div id="mapDiv" style="width: 800px; height: 600px; border: 1px solid #000;">
  </div>
  <!-- info window tabs -->
  <div id="tabs" dojotype="dijit.layout.TabContainer" style="width: 385px; height: 150px;">
    <div id="cityTab" dojotype="dijit.layout.ContentPane" title="Cities">
    </div>
    <div id="countyTab" dojotype="dijit.layout.ContentPane" title="Counties">
    </div>
  </div>
</body>
</html>