



Address Validation Service  
January 2010

## **Legal and Copyright Notices**

### **Payment**

You must remit payment in accordance with the FedEx Service Guide, tariff, service agreement or other terms or instructions provided to you by FedEx from time to time. You may not withhold payment on any shipments because of equipment failure or for the failure of FedEx to repair or replace any equipment.

### **Inaccurate Invoices**

If you generate an inaccurate invoice, FedEx may bill or refund to you the difference according to the FedEx Service Guide, tariff service agreement or other terms or instructions provided to you by FedEx from time to time. A request for refund on a FedEx shipment must be made in accordance with the applicable Service Guide, or terms or instructions provided by FedEx from time to time. A shipment given to FedEx with incorrect information is not eligible for refund under any FedEx money-back guarantees. FedEx may suspend any applicable money-back guarantees in the event of equipment failure or if it becomes inoperative.

### **Confidential and Proprietary**

The information contained in this *FedEx Web Services Developer Guide* is confidential and proprietary to FedEx Corporate Services, Inc. and its affiliates (collectively "FedEx"). No part of this *FedEx Web Services Developer Guide* may be distributed or disclosed in any form to any third party without written permission of FedEx. This *FedEx Web Services Developer Guide* is provided to you under and its use is subject to the terms and conditions of the FedEx Automation Agreement. The information in this document may be changed at any time without notice. Any conflict between this *FedEx Web Services Developer Guide*, the FedEx Automation Agreement, the FedEx Freight 100-Series Rules Tariff, and the FedEx Service Guide shall be governed by the FedEx Automation Agreement and the FedEx Service Guide, in that order.

© 2010 FedEx. FedEx and the FedEx logo are registered service marks. All rights reserved.  
Unpublished.

# Contents

- About this Guide** ..... **1**
  - Document Organization ..... 1
  - Resources ..... 1
  - Support ..... 1
  
- Introduction** ..... **2**
  - Document Overview ..... 3
  - Web Services, WSDL, and SOAP Overview ..... 5
  - Implementing FedEx Web Services ..... 10
  - Understanding the XML Schema ..... 11
  - Implementation Process ..... 18
  
- Address Validation Service** ..... **21**
  - Address Validation ..... 21
  
- Schema AddressValidationService\_v2.xsd** ..... **31**

# About this Guide

This guide describes how to work with the Address Validation Service feature of FedEx® Web Services. It is written for the application developer who uses Web Services to design and deploy applications enabled by FedEx. It describes how to get started with application development and how to use the Application Programming Interface (API). It also describes the service and business logic that drives the process.

## Document Organization

Each Web Service provides access to FedEx features. The service description includes service details and a full schema listing to facilitate application development.

## Resources

The following may also be useful for FedEx Web Services developers:

- FedEx Web Services Developer Resource Center: [fedex.com/developer](https://fedex.com/developer)
- FedEx Services At-a-Glance: [fedex.com/us/services/ata glance.html](https://fedex.com/us/services/ata glance.html)
- FedEx Service Guide: [fedex.com/us/services/pdf/](https://fedex.com/us/services/pdf/)
- Web Services organization home page: [webservices.org](https://webservices.org)
- O'Reilly XML.com: [webservices.xml.com](https://webservices.xml.com)

## Support

For FedEx Web Services technical support, you can reach FedEx at [websupport@fedex.com](mailto:websupport@fedex.com) or call 1.877.339.2774 and state "FedEx Web Services" at the voice prompt. Support hours are Monday through Friday, 7 a.m. to 12 a.m. (midnight) (CST) and Saturday, 7 a.m. to 7 p.m. (CST). For international customer support, call toll free 1.800.GoFedEx.

# Introduction

FedEx Web Services gives you the tools to build custom platform- and interface-independent applications that access FedEx features. You can use FedEx Web Services in a variety of ways to create customized integration solutions for your specific shipping needs. Here are just a few of the ways a company can use Web Services to streamline operations, improve visibility, and provide more choices to clients:

- **Verify Addresses and Improve Customer Satisfaction:** Prompt customers for additional information in the event of an address discrepancy or missing information with Address Validation Service.
- **Give Customers More Options:** Help customers learn about all the available shipping options and rates with Rate Services. You can also extend this service to your shopping cart and Web site, allowing customers to access money-saving information firsthand.
- **More Convenience:** Use the Locator Service to find the FedEx pickup location nearest your customer. Or, send an e-mail to your customers with a link to this service as part of your standard order-receipt process.
- **Offer Global Shipping Options:** Create shipping labels for worldwide locations. Improve customer service by offering more shipping options to customers in more countries with the consolidated Ship Service.
- **Reduce Customer Service Costs:** Decrease phone traffic from customers checking the status of their shipments and cut customer service costs. FedEx provides online Tracking and Visibility Services that allow you to provide customers with the status of shipments, Signature Proof of Delivery (SPOD), and Shipment Notification in the Ship Request.
- **Simplify Processes and Improve Satisfaction:** Create an e-mail returns shipping label by using the Ship Service's CreatePendingShipment Request. This transaction sends an e-mail with the address (URL) of a Web site where the recipient can log in and print a return label.

Why should developers be interested in Web Services?

- **Interoperability**—Any Web Service can interact with any other Web Service and can be written in any programming language.
- **Ubiquity**—Web Services communicate using HTTP and XML. Any connected device that supports these technologies can both host and access Web Services.
- **Low Barrier to Entry**—The concepts behind Web Services are easy to understand, and developers can quickly create and deploy them using many toolkits available on the Web.
- **Industry Support**—Major content providers and vendors support the Web Services movement.

Any application running on any platform can interact with a Web Service by using the SOAP and WDSL standards for message transfer and service discovery. By following the standards, applications can seamlessly communicate with platform services.

## Document Overview

The FedEx Web Services Developer Guide provides instructions for coding the functions you need to develop FedEx-supported applications. The following chapters make up the Web Services Developer Guide:

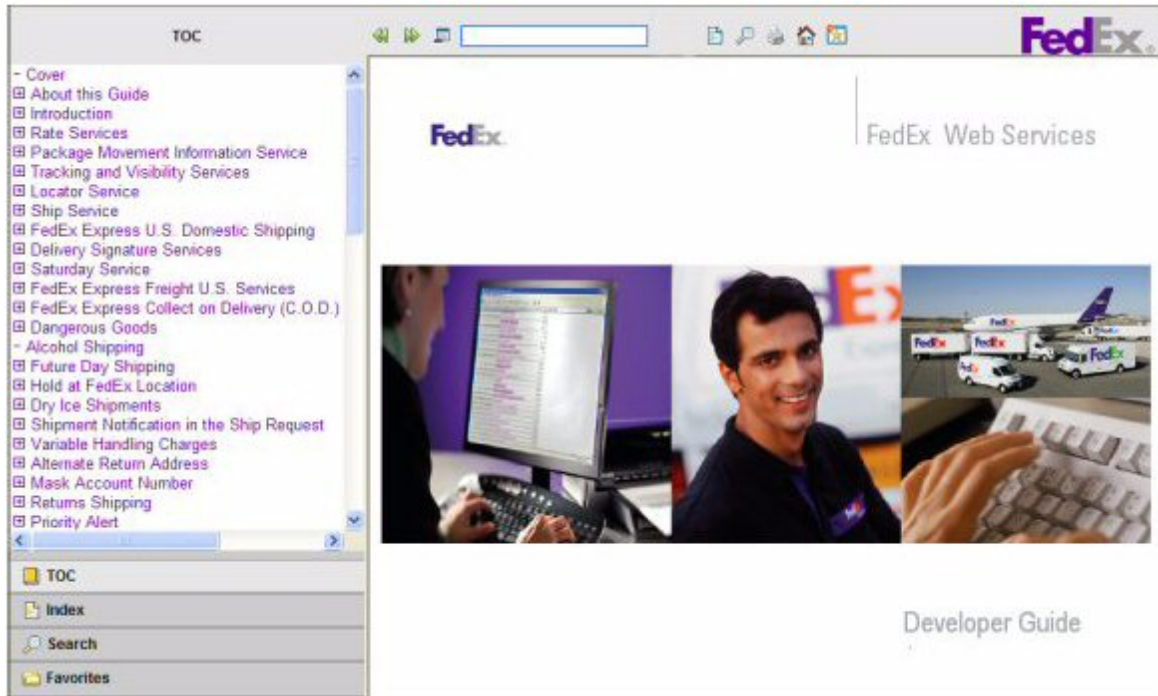
- Introduction (this chapter):
  - o Documentation overview and guidelines, including how to use the Help application and how to print this guide.
  - o Overview information about Web Services, including a high-level description of FedEx Web Services methods.
  - o Coding basics.
  - o Overview information about testing and certifying your application.
- [Address Validation Service](#) explains how to check your shipping addresses for accuracy before shipping.

This document describes Web Services coding information as follows:

- Service Details—Business rules for using the FedEx service.
- Service Options—Links to additional services that can be added to the basic Web Service.
- Coding Details—Best practices information, basic request and reply elements, and a link to error messages.
- XML Schema—A link to the layout for the service. This layout provides coding requirements for all elements in the schema.

## Using the Web Services Online Help










This guide is also available in the online help at the FedEx Developer Resource Center ([fedex.com/developer](http://fedex.com/developer)) in **Support > FedEx Web Services Developer Guide**.



Web Services Help opens in your default browser, such as Internet Explorer or Firefox. The first topic—in this case, the cover page—appears in the main window.

The Table of Contents (TOC) appears in the navigation column. Under the TOC you can choose the Index, Search, or Favorites options. Each of these features appears in the same column.

The toolbar across the top of the window displays the following elements:

-  Back—Returns you to the previously viewed topic.
-  Forward—Goes to the next topic as listed in the TOC.
-  Quick Search—Enter a search term and click  to highlight the term in the current topic. **This feature only searches the current topic.**
-  Hide Navigation—Hides the left navigation column.
-  Search—Opens the full search tool in the navigation column.
-  Print—Opens the **Print** dialog box. See [Printing This Guide or Online Help](#).
-  Home—Opens the default topic: in this case, the cover page.
-  Add Topic to Favorites—Saves the current topic to your **Favorites** list.

## Printing This Guide or Online Help

You can print all or part of this guide from both the PDF and WebHelp versions.

### ***Printing from the PDF Version***

From the PDF version you can print the complete document or a page range of the document.

Open the PDF file and click the printer icon  or click **File > Print**.

From the **Print** dialog box you can print the complete document, specify a page range, or choose from any of the available print options.

### ***Printing from the WebHelp Version***

From the WebHelp version you can print a single topic or a page range of that topic.

Open WebHelp and click the printer icon .

From the **Print** dialog box you can print the complete topic or specify a page range.

## Web Services, WSDL, and SOAP Overview

This section describes the standard coding technologies used in FedEx Web Services.

### Web Services

Web Services is a collection of programming technologies, including XML, Web Services Description Language (WSDL) and SOAP, which allow you to build programming solutions for specific messaging and application integration.

Web Services are, by definition, platform independent. FedEx Web Services allow developers to build custom applications that are independent of changes to the FedEx interface.

Note that FedEx Web Services are not offered as part of a Universal Description, Discovery, and Integration (UDDI) and must be downloaded from the FedEx Developer Resource Center ([fedex.com/developer](http://fedex.com/developer)) and stored locally for development and usage.

### WSDL

A SOAP request to, or response from, a service is generated according to the service's WSDL definition. A WSDL document describes a service. It is an XML document that provides information about what the service does, the methods that are available, their parameters, and parameter types. It describes how to communicate with the service in order to generate a request to, or decipher a response from, the service.

The purpose of a WSDL is to completely describe a Web Service to a client. A WSDL defines where the service is available and what communications protocol is used to talk to the service. It defines everything required to write a program to work with an XML Web Service. A WSDL document describes a Web Service using seven major elements. Elements can be abstract or

concrete. Abstract XML elements describe the Web Service: <types>, <message>, <operation>, <portType>.

Concrete XML elements provide connection details: <service>, <port>, <binding>.

Element	Definition								
<definitions>	The root element contains name space definitions.								
<portType>	The most important WSDL element. It is a set of all operations that a Web Service can accept and is a container for <operation> elements. This WSDL element describes a Web Service, the operations that can be performed, and the messages that are involved, and can be compared to a function library (or a module or a class) in a traditional programming language.								
<types>	Defines variable types used in the Web Service (both the parameters passed to a function and the type of the value passed back via the response). The data types are described by XML schema. This element contains user-defined data types (in the form of XML schema). For maximum platform neutrality, WSDL uses XML schema syntax to define data types.								
<message>	Defines the data elements of an operation. Each message can consist of one or more parts that can be compared to the parameters of a function call in a traditional programming language.								
<operation>	Child of the <binding> element that defines each operation that the port exposes. This element allows three messages only: <table border="1" data-bbox="391 919 1101 1066"> <thead> <tr> <th>Message</th> <th>Definition</th> </tr> </thead> <tbody> <tr> <td>Input Message</td> <td>Data Web Services receives</td> </tr> <tr> <td>Output Message</td> <td>Data Web Services sends</td> </tr> <tr> <td>Fault Message</td> <td>Error messages from Web Services</td> </tr> </tbody> </table>	Message	Definition	Input Message	Data Web Services receives	Output Message	Data Web Services sends	Fault Message	Error messages from Web Services
Message	Definition								
Input Message	Data Web Services receives								
Output Message	Data Web Services sends								
Fault Message	Error messages from Web Services								
<service>	The <service> element contains a <port> child element that describes the URL where the service is located. This is the location of the ultimate Web Service.								
<binding>	The <binding> element defines the message format and protocol details for each port. The binding element has two attributes: the name attribute and the type attribute. This element specifies how the client and the Web Service should send messages to one another.								

**Note:** For more information about the WSDL standard, refer to the W3C Web site at [w3.org/TR/wSDL](http://w3.org/TR/wSDL).

## SOAP

SOAP is a simple XML-based protocol that allows applications to exchange information over HTTP. SOAP is built on open standards supported by numerous development tools on various platforms. SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages. The SOAP request interface is an object in your application programming language.

SOAP enables the data to pass through layers of intermediaries and arrive at the ultimate receiver the way it was intended. It is worth noting that you may not need to actually construct the SOAP messages yourself—many development tools available today construct SOAP behind the scenes.

## SOAP Message

A SOAP message is an ordinary XML document that can be a “request” for a Web Service from a client or a “reply” from a Web Service to a client.

- Required <SOAP:Envelope>
- Optional <SOAP:Header>
- Required <SOAP:Body>

### *Example: Delete Tag Request (SOAP Message)*

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/
/XMLSchema" xmlns="http://fedex.com/ws/ship/v8">
<SOAP-ENV:Body>
<DeleteTagRequest>
<WebAuthenticationDetail>
<UserCredential>
<Key>xxxxxxxxxxxxxxxx</Key>
<Password></Password>
</UserCredential>
</WebAuthenticationDetail>
<ClientDetail>
<AccountNumber>xxxxxxxx</AccountNumber>
<MeterNumber>xxxxxxx</MeterNumber>
</ClientDetail>
<TransactionDetail>
<CustomerTransactionId>DE_Shakeout_wsvc</CustomerTransactionId>
</TransactionDetail>
<Version>
<ServiceId>ship</ServiceId>
<Major>8</Major>
<Intermediate>0</Intermediate>
<Minor>0</Minor>
</Version>
<DispatchLocationId>MQYA</DispatchLocationId>
<DispatchDate>2008-10-08</DispatchDate>
<Payment>
<PaymentType>SENDER</PaymentType>
<Payor>
<AccountNumber>xxxxxxxx</AccountNumber>
<CountryCode>US</CountryCode>
</Payor>
</Payment>
<ConfirmationNumber>xxxxxxxxxxxxxxxx</ConfirmationNumber>
</DeleteTagRequest>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

For more information about the SOAP standard, refer to the World Wide Web Consortium (W3C) Web site at [w3.org/TR/SOAP](http://w3.org/TR/SOAP).

## Non-SOAP Web Services

FedEx offers a non-SOAP Web Services solution that you can use to perform operations, without having to use tools that provide SOAP protocol support for Web Services. This may be convenient for developers using environments that do not provide support for SOAP. With this interface, XML documents are sent directly to the FedEx servers via the HTTP POST command. FedEx provides a set of specifications and examples to help with the development of this type of application.

To use the non-SOAP Web Service solution, you must meet the following requirements:

- A working knowledge of HTTPS and Secure Socket Layering encryption.
- The ability to provide a secure SSL connection to FedEx.
- The ability to code to a designated interface using XML.

The interfaces used in the SOAP and non-SOAP Web Services are defined in WSDL files. The WSDL files contain schemas that define the layout of the operations. The same WSDL file is used for both the SOAP and non-SOAP Web Service users.

Non-SOAP users are concerned only with the schema definitions and not the other WSDL components that are SOAP-specific. The XML data that is sent via the non-SOAP interface looks almost identical to the data that is sent via the SOAP interface. The only difference is that the data sent via the non-SOAP interface does not contain the wrapping Envelope and Body tags that are specific to SOAP. An example of a request using the non-SOAP interface looks like this:

```
<ns:TrackRequest xmlns:ns="http://fedex.com/ws/track/v4" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://fedex.com/ws/
track/v4 TrackService_v4.xsd ">
  <ns:WebAuthenticationDetail>
    <ns:UserCredential>
      <ns:Key>xxxxxxxxxxxxxxxx</ns:Key>
      <ns:Password>xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx</ns:Password>
    </ns:UserCredential>
  </ns:WebAuthenticationDetail>
  <ns:ClientDetail>
    <ns:AccountNumber>XXXXXXXXXX</ns:AccountNumber>
    <ns:MeterNumber>XXXXXXXXXX</ns:MeterNumber>
  </ns:ClientDetail>
  <ns:TransactionDetail>
    <ns:CustomerTransactionId>User Customizable Field</
ns:CustomerTransactionId></ns:TransactionDetail>
  <ns:Version>
    <ns:ServiceId>trck</ns:ServiceId>
    <ns:Major>4</ns:Major>
    <ns:Intermediate>0</ns:Intermediate>
    <ns:Minor>0</ns:Minor>
  </ns:Version>
  <ns:PackageIdentifier>
    <ns:Value>tttttttttttttttt</ns:Value>
    <ns:Type>TRACKING_NUMBER_OR_DOORTAG</ns:Type>
  </ns:PackageIdentifier>
```

```
<ns:IncludeDetailedScans>true</ns:IncludeDetailedScans>
</ns:TrackRequest>
```

## Error Handling

Error handling for non-SOAP transactions is different from error handling for SOAP operations. The SOAP specification provides an error handling mechanism that is not present for non-SOAP operations. For a SOAP operation, a fault is returned as a SOAP exception. For a non-SOAP request, the contents of the SOAP fault are returned as an XML document. These SOAP fault documents are returned in situations such as schema validation failures or when operation types are unrecognized. In the following example, a SOAP fault document is returned from a schema validation failure in which the “AccountNumber” element was incorrectly sent as the “AccountNumberx” element:

```
<soapenv:Fault xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <faultcode>soapenv:Server</faultcode>
  <faultstring>5: Schema validation failed for request.</faultstring>
  <detail>
    <con:fault xmlns:con="http://www.bea.com/wli/sb/context">
      <con:errorCode>5</con:errorCode>
      <con:reason>Schema validation failed for request.</con:reason>
      <con:details>
        <con1:ValidationFailureDetail xmlns:con1="http://www.bea.com/wli/sb/stages/transform/config">
          <con1:message>Expected element 'AccountNumber@http://fedex.com/ws/ship/v1' instead of 'AccountNumberx@http://fedex.com/ws/ship/v8' here in element ClientDetail@http://fedex.com/ws/ship/v8</con1:message>
          <con1:xmlLocation>
            <ship:AccountNumberx xmlns:ship="http://fedex.com/ws/ship/v8">000000000</ship:AccountNumberx>
          </con1:xmlLocation>
          <con1:message>Expected element 'AccountNumber@http://fedex.com/ws/ship/v1' before the end of the content in element ClientDetail@http://fedex.com/ws/ship/v8</con1:message>
          <con1:xmlLocation>
            <ship:ClientDetail xmlns:ship="http://fedex.com/ws/ship/v8">
              <ship:AccountNumberx>000000000000000000</ship:AccountNumberx>
              <ship:AccountNumberx>
                <ship:MeterNumber>0000000</ship:MeterNumber>
              </ship:ClientDetail>
            </con1:xmlLocation>
          </con1:ValidationFailureDetail>
        </con:details>
      <con:location>
        <con:node>Validate</con:node>
        <con:pipeline>Validate_request</con:pipeline>
        <con:stage>ValidateRequest</con:stage>
        <con:path>request-pipeline</con:path>
      </con:location>
```

```

    </con:fault>
  </detail>
</soapenv:Fault>

```

Each reply should be checked for the “Fault” element to indicate failure in processing the message. Note that the normal error processing still applies; this is an additional error check for incorrect syntax in XML documents.

Keep in mind that if you use either the SOAP or non-SOAP version of FedEx Web Services, labels are returned as Base64 encoded. To print shipping labels, you must decode labels before sending them to your printer.

### Non-SOAP HTTP POST Example

The following HTTPS POST example is a valid working example, but is not guaranteed to work for all programming languages/applications/host systems:

```

POST /xml HTTP/1.0
Referer: YourCompanyNameGoesHere
Host: gatewaybeta.fedex.com
Port: 443
Accept: image/gif, image/jpeg, image/pjpeg, text/plain, text/html, */*
Content-Type: image/gif
Content-length: %d
Your FedEx Transaction

```

Each line is followed by one new line character except Content-length and the FedEx transaction. Two new line characters follow the Content-length line. The FedEx transaction has no extra characters. The Content-length line should have the length of the FedEx transaction in place of the %d variable.

**Note:** Port 443 must be opened for bi-directional communication on your firewall.

After formatting your non-SOAP transaction and placing it in a HTTP POST request, open an SSL connection to the FedEx test server and send the request to FedEx through your SSL connection.

Next, parse the HTTPS response to determine if there were any errors. Examine the HTTP header to determine if any HTTP or Web Server errors were encountered. If you received a 200 status code, parse the reply to determine if there were any processing problems.

## Visual Basic Project Error

You may receive an error indicating that xxxxxx element is not set, even after setting it in the code. When you set a Boolean type element to True, you may also need to set the specified element to True.

## Implementing FedEx Web Services

Before you begin your implementation of FedEx Web Services, make note of the following guidelines:

- FedEx Web Services is designed for use by skilled developers who are familiar with SOAP protocol and the Web Services Description Language (WSDL).

- Unlike traditional client/server models, such as a Web server or Web page system, Web Services do not provide the user with a GUI. Instead, Web Services share business logic, data, and processes through a programmatic interface across a network.
- To perform a particular FedEx task such as tracking a package, you need to use a class, module, or function that creates your request, sends it to the FedEx platform, and handles the response.
- Web Services are designed to support any operating system and coding language. Downloadable sample code is available in Java, C++, C#, VB, .Net, and PHP languages.

## Understanding the XML Schema

The XML schema defines the messages that you can use to access the FedEx services. You create a request that contains business data and other instructions and send it to FedEx. FedEx replies with a response that contains the data resulting from the instructions you sent in. Notice that schema diagrams are conveniently linked to help you find information and child values.

The XML schema provides a means for defining the structure, content, and semantics of XML documents.

An XML schema defines:

- Elements and attributes that can appear in a document
- Elements that are child elements
- Order and number of child elements
- Whether an element is empty or can include text
- Data types, default values and fixed values for elements and attributes

Some important facts about the XML schema:

- Elements that contain sub-elements or carry attributes have complex types.
- Elements that contain numbers (and strings, and dates, etc.), but do not contain any sub-elements, have simple types. Some elements have attributes. Attributes always have simple types.
- Complex types in the instance document, and some of the simple types, are defined in the schema associated with a FedEx Web Service. Other simple types are defined as part of XML Schema's repertoire of built-in simple types.
- XML schema built-in simple types are prefixed by "xs:", which is associated with the XML Schema namespace through the declaration `xmlns:xs="http://www.w3.org/2001/XMLSchema"`, displayed in the schema element.
- The same prefix, and the same association, are also part of the names of built-in simple types, e.g. `xs:string`. This association identifies the elements and simple types as belonging to the vocabulary of the XML schema language, rather than the vocabulary of the schema author.

## Guide to the XML Schema

The XML schema for each WSDL provides details about the structure, content, and semantics of the request XML document sent to a FedEx Web Service and the XML document returned by that FedEx Web Service.

The top of each service schema includes:

- Schema location and schema file name that ends in an “.xsd” suffix.
- Alphabetical listing of complex types for the documented service.
- Alphabetical listing of schema simple types for the documented service.
- Input or request data type for the documented service.
- Output or reply data type for the documented service.

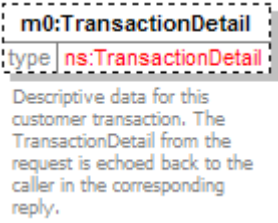

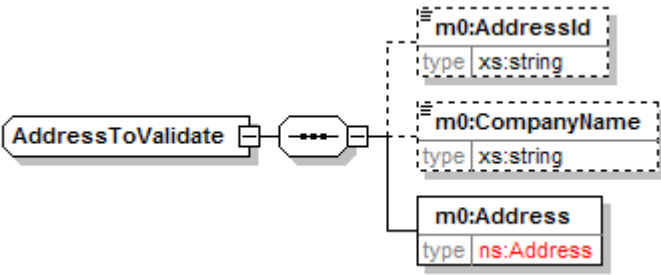
The remainder of the service schema contains tables of information about each element, complex type, and simple type.

Each table consists of some or all of the following sections: diagram, namespace, children, type, properties, used by, facets, and source.

## XML Schema Diagrams

XML schema diagrams describe the elements (usually associated with a request or reply), complex types, and simple types that make up the WSDL. The following table illustrates the relationships and behavior of elements and types.

Diagram	Description
<p>The diagram illustrates the structure of the <code>AddressValidationRequest</code> element. It is a parent element (solid border) containing several child elements:</p> <ul style="list-style-type: none"> <li><code>m0:WebAuthenticationDetail</code> (type <code>ns:WebAuthenticationDetail</code>): Descriptive data to be used in authentication of the sender's identity (and right to use FedEx web services).</li> <li><code>m0:ClientDetail</code> (type <code>ns:ClientDetail</code>): Descriptive data identifying the client submitting the transaction.</li> <li><code>m0:TransactionDetail</code> (type <code>ns:TransactionDetail</code>): Descriptive data for this customer transaction. The <code>TransactionDetail</code> from the request is echoed back to the caller in the corresponding reply. (Optional, indicated by a dotted line and dotted border).</li> <li><code>m0:Version</code> (type <code>ns:VersionId</code>): Identifies the version/level of a service operation expected by a caller (in each request) and performed by the callee (in each reply).</li> <li><code>m0:RequestTimestamp</code> (type <code>xs:dateTime</code>): A simple type.</li> <li><code>m0:Options</code> (type <code>ns:AddressValidationOptions</code>): A simple type.</li> <li><code>m0:AddressesToValidate</code> (type <code>ns:AddressToValidate</code>): A simple type with a cardinality of <code>1..100</code>.</li> </ul>	<p>Diagrams of a parent element, such as <code>AddressValidationRequest</code>, include connections to the child elements. Child elements can be simple or complex types.</p> <p>A child element connected with a solid line and surrounded by a box with a solid border represents a required type, such as <code>ClientDetail</code>.</p> <p>A child element connected by a dotted line and surrounded by a dotted border represents an optional type (<code>minOccurs="0"</code>), such as <code>TransactionDetail</code>.</p> <p><b>Note:</b> An element that is defined as <code>minOccurs="0"</code> may be required for some calls.</p> <p>Types that are documented include the documentation directly below the box.</p> <p>All children are linked by name below the diagram.</p>
<p><code>m0:Version</code> (type <code>ns:VersionId</code>)</p> <p>Identifies the version/level of a service operation expected by a caller (in each request) and performed by the callee (in each reply).</p>	<p>A box with a single solid border represents a single element that is required.</p> <p>The type can be simple or complex.</p>

Diagram	Description
 <p><b>m0:TransactionDetail</b> type ns:TransactionDetail</p> <p>Descriptive data for this customer transaction. The TransactionDetail from the request is echoed back to the caller in the corresponding reply.</p>	<p>A box with a dotted border indicates a single element that is optional.</p> <p>The type can be simple or complex.</p>
 <p><b>AddressValidationReply</b> (required multiple occurrence)</p> <ul style="list-style-type: none"> <li><b>m0:HighestSeverity</b> (required single occurrence) type ns:NotificationSeverityType</li> <li><b>m0:Notifications</b> (required multiple occurrence) type ns:Notification 1..∞</li> <li><b>m0:TransactionDetail</b> (optional single occurrence) type ns:TransactionDetail</li> <li><b>m0:ReplyTimestamp</b> (required single occurrence) type xs:dateTime</li> <li><b>m0:AddressResults</b> (required multiple occurrence) type ns:AddressValidationResult 1..100</li> <li><b>m0:Version</b> (optional single occurrence) type ns:VersionId</li> </ul> <p><b>AddressValidationResult</b> (optional multiple occurrence)</p> <ul style="list-style-type: none"> <li><b>m0:AddressId</b> (optional single occurrence) type xs:string</li> <li><b>m0:ProposedAddressDetails</b> (optional multiple occurrence) type ns:ProposedAddressDetail 0..10</li> </ul>	<p>A layered box represents a multiple occurrence element. A solid line represents a required multiple occurrence element.</p> <p>The number of possible occurrences appears below the box, as depicted by the AddressResults element.</p> <p>An unbounded number of occurrences is represented by the infinity ∞ symbol (maxOccurs="unbounded"), as depicted by the Notifications type.</p> <p>A layered box with a dotted border represents an optional multiple occurrence type (minOccurs="0"), such as ProposedAddressResults.</p> <p><b>Note:</b> An element that is defined as minOccurs="0" may be required for some calls.</p>
 <p><b>AddressToValidate</b> (required multiple occurrence)</p> <ul style="list-style-type: none"> <li><b>m0:AddressId</b> (optional single occurrence) type xs:string</li> <li><b>m0:CompanyName</b> (optional single occurrence) type xs:string</li> <li><b>m0:Address</b> (required single occurrence) type ns:Address</li> </ul>	<p>A standard type such as "string" appears in black text below element name.</p> <p>A FedEx-specific type such as "Address" appears in red text below the element name.</p>

### Required Elements

Most requests to FedEx require the following complex elements:

**Note:** These elements are common to most Web Services (see the table below for which WSDLs need which common elements) and are not documented service by service.

**WebAuthenticationDetail**—The WebAuthenticationDetail element includes user credentials issued by FedEx so that your transactions are recognized by the FedEx back-end systems. The following elements are required:

Element	Description
WebAuthenticationDetail	The descriptive data to be used in authentication of the sender's identity and right to use FedEx web services.
UserCredential	Credential used to authenticate a specific software application. This value is provided by FedEx after registration.
WebAuthenticationCredential	Two-part authentication string used to verify sender identity.
WebAuthenticationCredential/Key	Unique identifier assigned to each customer as part of authentication credential.
WebAuthenticationCredential/Password	Second part of the authentication credential which has a direct relationship with the credential key.

**Note:** Web Services now use two-factor authentication. If you do not have new credentials, the latest WSDLs will use your old authentication credentials. If you do not have a new user authentication credential, do not populate the password element.

**ClientDetail**—The ClientDetail element is required for all services that need your FedEx account number and meter number. Requirements are:

Element	Description
ClientDetail/AccountNumber	Your FedEx account number
ClientDetail/MeterNumber	Maximum of 9 characters. The associated meter number for your FedEx account number.

**TransactionDetail**—The TransactionDetail element is optional for all transactions. However, if you want to identify associated request and reply transactions, use this element.

Element	Description
TransactionDetail/CustomerTransactionId	Maximum of 40 characters. This element allows you to assign a unique identifier to your transaction. This element is returned in the reply and helps you match requests to replies.

**VersionId**—The VersionId element is required and uploads the WSDL version number to FedEx. FedEx provides the latest version number for the service you are using. This number should be updated when you implement a new version of the service.

**Sender Information**—Your sender information is required for all shipping transactions:

Element	Description
AccountNumber	If you include this element in the ship request, this entry overrides the account number in the ClientDetail element.
TIN	Tax Identification Number—this information is required for international shipments only.

Element	Description
Contact	The Contact element includes: <ul style="list-style-type: none"> <li>• PersonName</li> <li>• Title</li> <li>• CompanyName</li> <li>• Department</li> <li>• PhoneNumber</li> <li>• PagerNumber</li> <li>• FaxNumber</li> <li>• EMailAddress</li> </ul>
Address	This element includes: <ul style="list-style-type: none"> <li>• StreetLines: two StreetLines elements are allowed.</li> <li>• City</li> <li>• StateOrProvinceCode: required if your sender address is in the U.S. or Canada.</li> <li>• PostalCode: required.</li> <li>• UrbanizationCode: may be required if your sender address is in Puerto Rico.</li> <li>• CountryCode: required.</li> </ul>
Residential	Required if your sender address is considered a residential location. If you are unsure, use the Address Checker WSDL to check your address.

### **Reply Notifications**

Notifications are returned in replies. The Notification element provides the notification ranked according to their severity:

- HighestSeverity—This element ranks the level of severity of all notifications returned in the reply. Values are:
  - o FAILURE: Code/message explains that your request could not be handled at this time; please do not resubmit right now.
  - o ERROR: Code/message identifies a problem with your request data; you may fix the request data and try again.
  - o WARNING: Your request was successful. However, the code/message explains what had to be done to fulfill your request; you may need to determine whether that is what you intended, you may need to do this differently next time, or you may need to prepare for a future change. Request was completed.
  - o NOTE: Your request was successful. However, the code/message contains additional information about how your request was fulfilled; you do not need to take any special action.
  - o SUCCESS: Your request was successful. There are no NOTE or WARNING notifications.

**Note:** There is a possibility of multiple Notification objects (different severity levels) for a single request. The response notification severity values of ERROR, FAILURE and SUCCESS should never be combined in a single response.

In addition to these elements, service detail is returned in the reply. This detail is defined in each function chapter in the Developer Guide.

*Notification Examples*

For example, if you need to perform a US Address Correction, the service should accept a (domestic) Address object from its client and return that Address in a standardized form (canonical spelling and abbreviation of street name parts, elimination of redundant white space, data correction where possible, etc.). The following cases illustrate several notification types.

The example service has been assigned a NotificationSourceType value of "USACS".

**Case:**

Request to submit an Address that is valid and is already in standardized form (i.e., there is nothing to say except "OK").

Reply: Notifications:SUCCESS and Address:the original address (or copy).

**Case:**

Request to submit an Address that is valid but not in standardized form (e.g., the word "Boulevard" in a street name is replaced with the standard abbreviation "Blvd" and "Saint Louis" as a city name is replaced with "St Louis").

Reply: Notifications:{NOTE, "Standard abbreviation applied to street name"}, {NOTE, "Standard abbreviation applied to city name"} and Address:the original address, with modification made to the street name and city name.

**Case:**

Request to submit an Address that is valid but with only a 5-digit ZIP Code: the service supplies the ZIP+4 for the standardized address.

Reply: Notifications:{NOTE, "ZIP+4 suffix added"} and Address: the original address, with the four-digit suffix added to the ZIP Code.

**Case:**

Request to submit an Address that is identifiable by street data, city name, and state code, but with a 5-digit ZIP Code that does not match the other fields. The service supplies the correct ZIP+4 for the standardized address.

Reply: Notifications:{WARNING, "ZIP Code corrected to match rest of address"} and Address: the original address, with the replacement ZIP Code.

**Case:**

Request to submit an Address that has a bogus state code. The original address contains a ZIP+4 Code belonging to a city/state pair that matches the client's original city and street address. The service supplies the corresponding state code in the corrected address.

Reply: Notifications:{WARNING, "State code corrected to match city and ZIP Code"} and Address:the original address, with the revised state code.

**Case:**

Request to submit an Address that has an incorrect state code. The original address contains a ZIP+4 Code belonging to a city/state pair that matches the client's original city and street address. The service rejects the client's address.

Reply: Notifications:{ERROR, "State code is incorrect for city/ZIP combination"} and Address: empty (either all fields blank or no Address at all).

**Case:**

Request to submit an Address that contains only a single street line (no city, state or ZIP Code). The service rejects the request.

Reply: Notifications: {ERROR, "City name is missing and cannot be corrected"}, {ERROR, "State code is missing and cannot be corrected"}, {ERROR, "ZIP Code is missing and cannot be corrected"} and Address:empty (either all fields blank or no Address at all)

**Case:**

Request to submit an Address, but the address correction service's database server is down or fails.

Reply: Notifications: {FAILURE, "Service temporarily unavailable"}, Address: empty (either all fields blank or no Address at all).

## Implementation Process

Planning your integration and organizing your application data to address your shipping needs can sometimes take more time than the actual implementation of the integration. FedEx Web Services conforms to industry standards and is compatible with a comprehensive array of developer's tools. This ensures the fastest time-to-market with maximum flexibility to integrate FedEx transactions and information into your applications. FedEx WSDLs are fully interoperable with any product or developer's tool that also conforms to the WS-I Basic Profile. For details, see [ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html](http://ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html).

The process of integrating an application with FedEx Web Services includes accessing documentation, sample code, and WSDL downloads from the Developers Resource Center. In addition, all developers can obtain a test meter number and engage in real-time online testing in the FedEx hosted test environment.

## Testing

FedEx supplies a complete online operating environment with which to test your applications against live FedEx servers. In order to execute test interactions, you must first include a test account number, test meter number, authentication key, and password in your code. These credentials are provided to registered developers.

## Certification

Certification is the process of ensuring that your implementation meets a number of requirements for safe, secure, and effective operation of your solution in the FedEx production environment. Certification requirements differ based on whether you are a corporate or commercial developer, and whether you are implementing using the advanced or standard services.

## Go to Production

Once an application has passed certification, the developer must replace the test credentials with the production credentials issued by FedEx. The application connection is then directed to the production servers, and the application is live.

## **Requirements for Corporate and Non-Commercial Developers**

There are some differences in how support is provided and in the approvals required to go into production that depend on whether you are creating an application for use by your own company or you are planning to resell your solution to others.

### **Requirements and Resources for Corporate Developers**

Corporate developers are typically part of a dedicated development team at a single company. This category also includes third-party developers (consultants) hired by the company to work on its behalf. In all cases, the integration will be used by the company itself and will not be resold or distributed outside of its own footprint. In this situation, FedEx can support the customer directly.

<b>Requirements and Resources for Corporate Developers</b>	
Must be accepted into the FedEx Compatible Solutions Program	No
Self-certification of implementations using standard services	Yes
Self-certification of implementations using advanced services	No
Certification Assistance	Yes (WISC team)
FedEx supports the customer directly	Yes

### *Preproduction Assistance*

Preproduction assistance is available via the FedEx Web Integrated Solutions Consultation (WISC) Team. If you are in the preproduction stages of implementing a FedEx Web Integrated Solution and would like to speak with a FedEx Integration Consultant who can assist you in understanding FedEx Web Services, contact your FedEx sales executive or technical support at 1.877.339.2774 Monday-Friday, 7 a.m.-12 a.m. and Saturday 7 a.m.-7 p.m. (CST). Both your FedEx sales executive and technical support can request a WISC Team member to contact you within 3 business days.

Corporate developers may find that solutions to their needs have already been implemented by a software vendor that is part of the FedEx Compatible Solution Program. If improved time-to-market, cost containment, or specialized knowledge is needed, corporate development planners may want to review the available third-party solutions. To see a list of the solutions provided by the FedEx Compatible Solutions Program providers, go to the Available FedEx Compatible Solutions page at [fedex.allegis.com/LeadReg.asp](http://fedex.allegis.com/LeadReg.asp).

### *Requirements for Consultants*

Consultants developing on behalf of a corporate customer must ensure that their client provides their account information and a signed End User License Agreement to FedEx in order to obtain a production test meter.

### **Requirements and Resources for Commercial Developers**

Commercial developers create solutions with the intent of distributing/reselling them to their customers. Because they are deployed in a variety of situations, commercial integrations generally require a higher order of “fit and finish.” Commercial developers are responsible for supporting their products for their customers. FedEx has a dedicated team of professionals to

help developers commercialize their products and to coordinate the three-way interplay between the developer, the end customer, and FedEx.

<b>Requirements and Resources for Commercial Developers</b>	
Must be accepted into the FedEx Compatible Solutions Program	Yes (Required)
Self-certification of implementation using Standard Services	No
Self-certification of implementations using Advanced Services	No
Certification Assistance	Yes
FedEx supports the customer directly	No
FedEx supports the commercial developer's customer	Indirectly

If you are a commercial developer interested in becoming a FedEx Compatible Solution Provider, visit [fedex.com/us/compatiblesolutions/provider/](https://fedex.com/us/compatiblesolutions/provider/) for more information about the FedEx Compatible Solutions Program.

### **URL Errors**

If a VB.NET or C# project still sends transactions to the test server after changing the URL in the WSDLs to point to production, perform the following:

- Make sure permissions are already activated in the production environment.
- Copy the WSDL files to a different folder.
- Follow the directions on changing the new WSDL files to point to production as described in the Developer Resource Center in the Move to Production topic.
- Remove existing Web Services references from your project that point to old WSDLs containing the URLs to the test environment.
- Create new Web references that point to the modified WSDLs. Use the same names as the old references.
- Compile and test the project. Your new production credentials should work for Standard Web Services, such as rating or tracking without extra permissions. Advanced Web Services require permissions to be active before they will work. Old test key values will now return an error message.

# Address Validation Service

Use the Address Validation Service to validate or complete recipient addresses.

Validate recipient addresses before you ship packages, provide descriptive error details and corrected options for invalid addresses, and/or determine whether an address is business or residential to increase the accuracy of courtesy rate quotes.

## Address Validation

The AddressValidation WSDL AddressValidationRequest allows you to validate recipient address information before you ship a package. Correct addresses on the shipping label will eliminate delivery delays and additional service fees.

Use the Address Validation request to perform the following:

- Confirm the validity and completeness of U.S., Puerto Rico, and Canadian addresses.
- Complete incomplete recipient addresses.
- Correct invalid recipient addresses.
- Determine whether an address is business or residential to increase the accuracy of courtesy rate quotes. Applies to U.S. addresses only.

## Address Validation Service Details

The following service details apply to Address Validation:

- Provides street level matches.
- Contains a database of company listing to improve your results (not all companies are listed).
- Receives monthly updates to its address database.
- Checks addresses within the United States, Puerto Rico, and Canada.
- Can distinguish between U.S. business and U.S. residential addresses if an exact match is found.
- Does not currently verify suite or apartment numbers.
- Does not match addresses based upon individual/personal names, but may check an address by matching company names that correspond to street addresses.
- CASS certified (Coding Accuracy Support System). A service and rating system for mailers that measures and helps to improve address accuracy.

- FedEx does not normally deliver to P.O. box addresses for U.S. or U.S. inbound shipments. However, FedEx may deliver to post office boxes in some rural locations if the P.O. box is associated with an address. You may also use P.O. box addresses for certain international locations, including shipments to Puerto Rico, but you must include a valid phone, fax or telex number on the label. FedEx cannot deliver to U.S. military P.O. box addresses such as APO and FPO.

For more detailed information about the services offered by FedEx, see the electronic [FedEx Service Guide](#).

## How FedEx Address Validation Works

- Checks if the street exists in the city, state or province, or postal code entered.
- Checks if the street number is within a valid range for the street entered.
- Provides an exact match or possible alternatives when an exact match cannot be found based on the street number, street name, city, state or province, and postal code entered.
- Informs you if no possible alternatives can be found based on the street number, street name, city, state or province, or postal code entered.
- Up to 100 addresses can be checked in one Web Service request.

## Tips on Getting Accurate Address Matches

**Use correct spacing:** Make sure spaces are placed correctly and avoid unnecessary spaces.

**Use correct spelling:** Eliminate spelling and typographic errors. Make sure you have the correct usage of the number zero (0) and letter O.

**Avoid special characters:** Refrain from using special characters not required for the address, such as periods after abbreviations (Ave vs. Ave.)

**Provide additional address and street information:** Providing additional address information can increase the accuracy of address results. For example:

- Building or house number such as 1, 1A, One
- Street name such as Main, George Washington, 42nd
- Street Suffix such as Road, Avenue, Rd, Ave

**Enter city, state/province and postal code:** Providing all address information will increase the accuracy of your results. The ZIP+4 portion of the postal code is not necessary to check an address.

**Use correct abbreviations:** The United States Postal Service and postal authorities in other countries define standard abbreviations for state/province, street suffix, and apartment/unit designations. A nonstandard abbreviation may cause poor search results. If you are unsure about an abbreviation, do not use it.

**Company:** Providing a company name may improve your results. Some addresses have specific company names assigned to them. By including the company name in your transaction, Address Validation can search for that company and address.

## Multiple Address Results

You receive multiple address results when an exact match was not found. You should confirm an address for accuracy before using it to ship a package. To narrow your results, you can provide more specific address information and check the address again.

Urbanization (Puerto Rico only): This descriptor, commonly used in urban areas of Puerto Rico, is an important part of the address format as it describes the location of a given street. In Puerto Rico, repeated street names and address number ranges can be found within the same postal code. These streets can have the same house number ranges. In these cases, the urbanization name is needed to correctly identify the location of a particular address.

For example:

Sr Pedro Rivera  
 Urb Hermosillo  
 123 Calle 1  
 Bayamon, PR 00961-1212

## Address Validation Coding Details

The following information is the minimum required to check an address:

- Address
- City and State or Province or postal code

### ***AddressValidationRequest Elements***

Element	Required	Description
AddressValidationRequest/ RequestTimestamp	Required	Time of request based on shipper's time zone. Defaults to CDT. The date format must be YYYY-MM-DDTHH:MM:SS-xx:xx. The time must be in the format: HH:MM:SS using a 24-hour clock. The date and time are separated by the letter T (e.g., 2009-06-26T17:00:00). The UTC offset indicates the number of hours/minutes (e.g. xx:xx) from UTC (e.g 2009-06-26T17:00:00-04:00 is defined as June 26, 2009 5:00 p.m. Eastern Time).
AddressValidationRequest/ AddressToValidate	Required	This element contains basic address information for validation, including: <ul style="list-style-type: none"> <li>• Company</li> <li>• StreetLines</li> <li>• City</li> <li>• StateorProvinceCode</li> <li>• PostalCode</li> <li>• UrbanizationCode</li> <li>• CountryCode</li> <li>• Residential</li> </ul> <b>Note:</b> Up to 100 addresses can be validated in one request.

Element	Required	Description
AddressValidationRequest/ AddressValidationOptions	Optional	<p>In addition to address information, you can include the following elements to further identify the type of address validation information or formatting you want in the reply:</p> <p>VerifyAddresses to validate all address elements and return in the reply.</p> <p>CheckResidentialStatus check addresses for residential status only.</p> <p>MaximumNumberOfMatches allows you to configure the number of possible matches returned. Maximum is 10.</p> <p>StreetAccuracy: Values are: EXACT, TIGHT, MEDIUM, and LOOSE.</p> <p>DirectionalAccuracy: Values are EXACT, TIGHT, MEDIUM and LOOSE.</p> <p>CompanyNameAccuracy: Values are EXACT, TIGHT, MEDIUM and LOOSE.</p> <p>For U.S. addresses only, you can control the algorithm to use when determining if an input address matches an address in the postal database. Valid values are:</p> <ul style="list-style-type: none"> <li>• EXACT: input must match the database exactly.</li> <li>• TIGHT: matching of address is allowed for slight variance</li> <li>• MEDIUM: matching of address allows for more variance of address and provides corrections [default]</li> <li>• LOOSE: matching of address is minimal</li> </ul> <p><b>Warning:</b> Selecting EXACT means that every part of the address must match the postal database exactly, and no correction will be made to the address for you. It is recommended to use the MEDIUM setting to get better results.</p> <p>ConvertToUpperCase element controls whether addresses are returned in upper case text.</p> <p>RecognizeAlternateCityNames recognizes alternate city names. For example, if you have an address whose city is Hollywood, if the address can be verified as in Los Angeles, address verification will be performed instead of returning an error.</p> <p>ReturnParsedElements returns the address validation elements in the reply, as verified by the system before validation.</p>

## AddressValidationReply Elements

Any error conditions or address-checking issues are returned in the Address Validation reply. The following table describes Address Validation reply elements:

Element	Description
AddressID	Every verified address is assigned an ID to help you match submitted addresses with verified information.
ProposedAddressDetails/Score	The Score element is used to rate the submitted address. If the Score is too low, the service returns the "Address Not Validated" message. The Score is an integer ranging from 0 to 100, with 100 being the highest and zero indicating failure.
ProposedAddressDetails/Changes	<p>Returned values are:</p> <ul style="list-style-type: none"> <li>• APARTMENT_NUMBER_NOT_FOUND</li> <li>• APARTMENT_NUMBER_REQUIRED</li> <li>• NORMALIZED</li> <li>• REMOVED_DATA</li> <li>• BOX_NUMBER_REQUIRED</li> <li>• NO_CHANGES</li> <li>• MODIFIED_TO_ACHIEVE_MATCH</li> <li>• STREET_RANGE_MATCH</li> <li>• BOX_NUMBER_MATCH**</li> <li>• RR_OR_HC_MATCH</li> <li>• CITY_MATCH</li> <li>• POSTAL_CODE_MATCH</li> <li>• RR_OR_HC_BOX_NUMBER_NEEDED</li> <li>• FORMATTED_FOR_COUNTRY</li> <li>• APO_OR_FPO_MATCH</li> <li>• GENERAL_DELIVERY_MATCH</li> <li>• FIELD_TRUNCATED</li> <li>• UNABLE_TO_APPEND_NON_ADDRESS_DATA</li> <li>• INSUFFICIENT_DATA</li> <li>• HOUSE_OR_BOX_NUMBER_NOT_FOUND</li> <li>• POSTAL_CODE_NOT_FOUND</li> <li>• INVALID_COUNTRY</li> <li>• SERVICE_UNAVAILABLE_FOR_ADDRESS</li> </ul> <p>**If BOX_NUMBER_MATCH is returned in the reply, remember FedEx does not normally deliver to P.O. box addresses for U.S. addresses or for U.S. inbound shipments.</p>
ProposedAddressDetails/ResidentialStatus	<p>Returned values are:</p> <ul style="list-style-type: none"> <li>• UNDETERMINED</li> <li>• BUSINESS</li> <li>• RESIDENTIAL</li> <li>• INSUFFICIENT_DATA</li> <li>• UNAVAILABLE</li> <li>• NOT_APPLICABLE_TO_COUNTRY</li> </ul>

Element	Description
ProposedAddressDetails/ DeliveryPointValidation	Returned values are: <ul style="list-style-type: none"> <li>• CONFIRMED</li> <li>• UNCONFIRMED</li> <li>• UNAVAILABLE</li> </ul>
ProposedAddressDetails/CompanyName	The company name as submitted for validation.
ProposedAddressDetails/Address	The address as submitted for validation.
ProposedAddressDetails/ ParsedCompanyName	The verified company name.
ProposedAddressDetails/ParsedAddress	The verified address.
ProposedAddressDetails/ RemovedNon-AddressData	Any information removed from the submitted address before validation.

### **XML Schema**

See [Schema AddressValidationService\\_v2.xsd](#).

### **Samples**

You can download sample service requests and replies with the WSDLs from the FedEx Developer Resource Center Technical Resources.

## Mapping EAS CheckAddressResponse Code to AddressValidationReply

### Notification

If EAS returns a nonzero systemStatus, the AddressValidationReply will contain a severity notification of FAILURE and a code equal to that systemStatus.

### Changes

EAS addressIndicator (indicator attribute)	Changes Element
100	APARTMENT_NUMBER_NOT_FOUND
101	APARTMENT_NUMBER_REQUIRED
102	NORMALIZED
103	REMOVED_DATA
104	BOX_NUMBER_REQUIRED
200	NO_CHANGES
201	MODIFIED_TO_ACHIEVE_MATCH
202	STREET_RANGE_MATCH
203	BOX_NUMBER_MATCH
204	RR_OR_HC_MATCH
205	CITY_MATCH
206	POSTAL_CODE_MATCH
207	RR_OR_HC_BOX_NUMBER_NEEDED
208	FORMATTED_FOR_COUNTRY
209	APO_OR_FPO_MATCH
210	GENERAL_DELIVERY_MATCH
211	FIELD_TRUNCATED
212	UNABLE_TO_APPEND_NON_ADDRESS_DATA
300	INSUFFICIENT_DATA
301	HOUSE_OR_BOX_NUMBER_NOT_FOUND
303	POSTAL_CODE_NOT_FOUND
305	INVALID_COUNTRY
400	SERVICE_UNAVAILABLE_FOR_ADDRESS

## Residential Status

EAS businessResidentialIndicator	Residential Status Element
1	UNDETERMINED
2	BUSINESS
3	RESIDENTIAL
4	INSUFFICIENT_DATA
5	UNAVAILABLE
6	NOT_APPLICABLE_TO_COUNTRY

## DeliveryPointValidation

EAS dpvIndicator	DeliveryPointValidation Element
1	CONFIRMED
2	UNCONFIRMED
3	UNAVAILABLE

## Known Service Issue

The Address Validation Web Service schema contains nested nodes that have the maxOccurs attribute set. The Web Services Description Language Tool (WSDL.exe), when used to generate the client information, creates multidimensional arrays in the generated Reference.vb / Reference.cs file. Therefore, the generated Reference file contains incorrect types for the nested nodes.

### Solution

- Search for string “>()()” in Reference.vb or for string “[][]” in Reference.cs file; you’ll see Class ParsedAddress.

```
'''<remarks/>
    <System.CodeDom.Compiler.GeneratedCodeAttribute("System.Xml",
"2.0.50727.42"),
    System.SerializableAttribute(),
    System.Diagnostics.DebuggerStepThroughAttribute(),
    System.ComponentModel.DesignerCategoryAttribute("code"),
    System.Xml.Serialization.XmlTypeAttribute([Namespace]:="http://fedex.com/
ws/addressvalidation")>
    Partial Public Class ParsedAddress
```

```
        Private parsedUrbanizationCodeField() As ParsedElement
```

```
        Private parsedStreetLineField() () As ParsedElement
```

```
        Private parsedCityField() As ParsedElement
```

- Remove extra “()” for VB.NET and “[]” for C# in front of parsedStreetLineField member.

```
/// <remarks/>
    [System.CodeDom.Compiler.GeneratedCodeAttribute("System.Xml",
"2.0.50727.42")]
```

```
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlTypeAttribute(Namespace="http://fedex.com/ws/
addressvalidation")]
public partial class ParsedAddress {
```

```
    private ParsedElement[] parsedUrbanizationCodeField;
```

```
    private ParsedElement[][] parsedStreetLineField;
```

```
    private ParsedElement[] parsedCityField;
```

- Search for the next “>()” in Reference.vb or “[[]]” in Reference.cs file; you’ll find Property ParsedStreetLine.

```
'''<remarks/>
    <System.Xml.Serialization.XmlArrayItemAttribute("Elements",
GetType(ParsedElement), IsNullable:=false)> _
    Public Property ParsedStreetLine() As ParsedElement() ()
        Get
            Return Me.parsedStreetLineField
        End Get
        Set
```

- Remove extra “()” for VB.Net and “[[]]” for C# in front of the ParsedElement.

```
/// <remarks/>
    [System.Xml.Serialization.XmlArrayItemAttribute("Elements",
typeof(ParsedElement), IsNullable=false)]
    public ParsedElement[][] ParsedStreetLine {
        get {
            return this.parsedStreetLineField;
        }
        set
```

**Note:** Web reference changes will be lost and need to be made manually.

- Search for ParsedElement[ ][ ] in ParsedAddress.java; you will find first reference. Remove extra “[[]]” from ParsedElement[[]].

```
public class ParsedAddress implements java.io.Serializable {
    private com.fedex.addressvalidation.stub.ParsedElement[]
parsedUrbanizationCode;
    private com.fedex.addressvalidation.stub.ParsedElement[][]
parsedStreetLine;
    private com.fedex.addressvalidation.stub.ParsedElement[] parsedCity;
```

- Continue the search for ParsedElement[ ][ ] in ParsedAddress.java; you will find another reference to ParsedElement[[]]. Remove extra “[[]]” from ParsedElement[[]].

```
public ParsedAddress(
    com.fedex.addressvalidation.stub.ParsedElement[]
parsedUrbanizationCode,
    com.fedex.addressvalidation.stub.ParsedElement[][] parsedStreetLine,
    com.fedex.addressvalidation.stub.ParsedElement[] parsedCity,
```

- Continue the search for `ParsedElement[][]` in `ParsedAddress.java`; you will find another reference to `ParsedElement[][]`. Remove extra “`[]`” from `ParsedElement[][]`.

```

    public com.fedex.addressvalidation.stub.ParsedElement[][]
getParsedStreetLine() {
    return parsedStreetLine;
}

```

- Continue the search for `ParsedElement[][]` in `ParsedAddress.java`; you will find another reference to `ParsedElement[][]`. Remove extra “`[]`” from `ParsedElement[][]`.

```

public void
setParsedStreetLine(com.fedex.addressvalidation.stub.ParsedElement[][]
parsedStreetLine) {
    this.parsedStreetLine = parsedStreetLine;
}

```

- Comment the code out as mentioned below for the following get/set methods:

```

/*
    public com.fedex.addressvalidation.stub.ParsedElement[]
getParsedStreetLine(int i) {
    return this.parsedStreetLine[i];
}
    public void setParsedStreetLine(int i,
com.fedex.addressvalidation.stub.ParsedElement[] _value) {
    this.parsedStreetLine[i] = _value;
}
*/

```

# Schema

## AddressValidationService\_v2.xsd

The following pages describe the AddressValidationService schema in detail.

# Schema

## AddressValidationService\_v2.xsd

targetNamespace: <http://fedex.com/ws/addressvalidation/v2>

### Elements

[AddressValidationReply](#)  
[AddressValidationRequest](#)

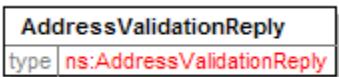
### Complex types

[Address](#)  
[AddressToValidate](#)  
[AddressValidationOptions](#)  
[AddressValidationReply](#)  
[AddressValidationRequest](#)  
[AddressValidationResult](#)  
[ClientDetail](#)  
[Localization](#)  
[Notification](#)  
[NotificationParameter](#)  
[ParsedAddress](#)  
[ParsedAddressPart](#)  
[ParsedElement](#)  
[ProposedAddressDetail](#)  
[TransactionDetail](#)  
[VersionId](#)  
[WebAuthenticationCredential](#)  
[WebAuthenticationDetail](#)

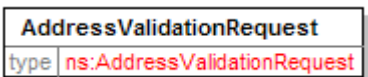
### Simple types

[AddressValidationAccuracyType](#)  
[AddressValidationChangeType](#)  
[DeliveryPointValidationType](#)  
[NotificationSeverityType](#)  
[ResidentialStatusType](#)

### element AddressValidationReply

diagram	
type	ns:AddressValidationReply
source	<code>&lt;xs:element name="AddressValidationReply" type="ns:AddressValidationReply"/&gt;</code>

### element AddressValidationRequest

diagram	
type	ns:AddressValidationRequest
source	<code>&lt;xs:element name="AddressValidationRequest" type="ns:AddressValidationRequest"/&gt;</code>

complexType Address

<p>diagram</p>	<div style="border: 1px dashed black; padding: 5px;"> <p><b>m0:StreetLines</b> type xs:string 0..4 Combination of number, street name, etc. At least one line is required for a valid physical address; empty lines should not be included.</p> <p><b>m0:City</b> type xs:string Name of city, town, etc.</p> <p><b>m0:StateOrProvinceCode</b> type xs:string Identifying abbreviation for US state, Canada province, etc. Format and presence of this field will vary, depending on country.</p> <p><b>m0:PostalCode</b> type xs:string Identification of a region (usually small) for mail/package delivery. Format and presence of this field will vary, depending on country. This element is required if both the City and StateOrProvinceCode are not present.</p> <p><b>m0:UrbanizationCode</b> type xs:string Relevant only to addresses in Puerto Rico. In Puerto Rico, multiple addresses within the same ZIP code can have the same house number and street name. When this is the case, the urbanization code is needed to distinguish them.</p> <p><b>m0:CountryCode</b> type xs:string Identification of a country.</p> <p><b>m0:Residential</b> type xs:boolean Indicates whether this address is residential (as opposed to commercial).</p> </div>
<p>children</p>	<p><a href="#">m0:StreetLines</a> <a href="#">m0:City</a> <a href="#">m0:StateOrProvinceCode</a> <a href="#">m0:PostalCode</a> <a href="#">m0:UrbanizationCode</a> <a href="#">m0:CountryCode</a> <a href="#">m0:Residential</a></p>
<p>source</p>	<pre>&lt;xs:complexType name="Address"&gt;</pre>

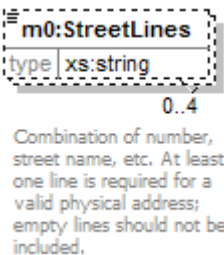
```

<xs:annotation>
  <xs:documentation>The descriptive data for a physical location.</xs:documentation>
</xs:annotation>
<xs:sequence>
  <xs:element name="StreetLines" type="xs:string" minOccurs="0" maxOccurs="4">
    <xs:annotation>
      <xs:documentation>Combination of number, street name, etc. At least one line is required for
a valid physical address; empty lines should not be included.</xs:documentation>
      <xs:appinfo>
        <MaxLength>35</MaxLength>
      </xs:appinfo>
    </xs:annotation>
  </xs:element>
  <xs:element name="City" type="xs:string" minOccurs="0">
    <xs:annotation>
      <xs:documentation>Name of city, town, etc.</xs:documentation>
      <xs:appinfo>
        <MaxLength>
          <Express>35</Express>
          <Ground>20</Ground>
        </MaxLength>
      </xs:appinfo>
    </xs:annotation>
  </xs:element>
  <xs:element name="StateOrProvinceCode" type="xs:string" minOccurs="0">
    <xs:annotation>
      <xs:documentation>Identifying abbreviation for US state, Canada province, etc. Format and
presence of this field will vary, depending on country.</xs:documentation>
      <xs:appinfo>
        <MaxLength>14</MaxLength>
      </xs:appinfo>
    </xs:annotation>
  </xs:element>
  <xs:element name="PostalCode" type="xs:string" minOccurs="0">
    <xs:annotation>
      <xs:documentation>Identification of a region (usually small) for mail/package delivery. Format
and presence of this field will vary, depending on country. This element is required if both the City
and StateOrProvinceCode are not present.</xs:documentation>
      <xs:appinfo>
        <MaxLength>16</MaxLength>
      </xs:appinfo>
    </xs:annotation>
  </xs:element>
  <xs:element name="UrbanizationCode" type="xs:string" minOccurs="0">
    <xs:annotation>
      <xs:documentation>Relevant only to addresses in Puerto Rico. In Puerto Rico, multiple
addresses within the same ZIP code can have the same house number and street name. When this
is the case, the urbanization code is needed to distinguish them.</xs:documentation>
      <xs:appinfo>
        <MaxLength>100</MaxLength>
      </xs:appinfo>
    </xs:annotation>
  </xs:element>
  <xs:element name="CountryCode" type="xs:string" minOccurs="0">
    <xs:annotation>
      <xs:documentation>Identification of a country.</xs:documentation>

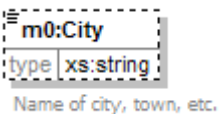
```

	<pre> &lt;xs:appinfo&gt;   &lt;MaxLength&gt;2&lt;/MaxLength&gt; &lt;/xs:appinfo&gt; &lt;/xs:annotation&gt; &lt;/xs:element&gt; &lt;xs:element name="Residential" type="xs:boolean" minOccurs="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Indicates whether this address is residential (as opposed to commercial).&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt; &lt;/xs:sequence&gt; &lt;/xs:complexType&gt; </pre>
--	--

### element Address/StreetLines

diagram									
type	<b>xs:string</b>								
properties	<table border="0"> <tr><td>isRef</td><td>0</td></tr> <tr><td>minOcc</td><td>0</td></tr> <tr><td>maxOcc</td><td>4</td></tr> <tr><td>content</td><td>simple</td></tr> </table>	isRef	0	minOcc	0	maxOcc	4	content	simple
isRef	0								
minOcc	0								
maxOcc	4								
content	simple								
source	<pre> &lt;xs:element name="StreetLines" type="xs:string" minOccurs="0" maxOccurs="4"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Combination of number, street name, etc. At least one line is required for a valid physical address; empty lines should not be included.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt;   &lt;xs:appinfo&gt;     &lt;MaxLength&gt;35&lt;/MaxLength&gt;   &lt;/xs:appinfo&gt; &lt;/xs:annotation&gt; &lt;/xs:element&gt; </pre>								

### element Address/City


diagram									
type	<b>xs:string</b>								
properties	<table border="0"> <tr><td>isRef</td><td>0</td></tr> <tr><td>minOcc</td><td>0</td></tr> <tr><td>maxOcc</td><td>1</td></tr> <tr><td>content</td><td>simple</td></tr> </table>	isRef	0	minOcc	0	maxOcc	1	content	simple
isRef	0								
minOcc	0								
maxOcc	1								
content	simple								
source	<pre> &lt;xs:element name="City" type="xs:string" minOccurs="0"&gt;   &lt;xs:annotation&gt; </pre>								

	<pre> &lt;xs:documentation&gt;Name of city, town, etc.&lt;/xs:documentation&gt; &lt;xs:appinfo&gt;   &lt;MaxLength&gt;     &lt;Express&gt;35&lt;/Express&gt;     &lt;Ground&gt;20&lt;/Ground&gt;   &lt;/MaxLength&gt; &lt;/xs:appinfo&gt; &lt;/xs:annotation&gt; &lt;/xs:element&gt; </pre>
--	---

**element Address/StateOrProvinceCode**

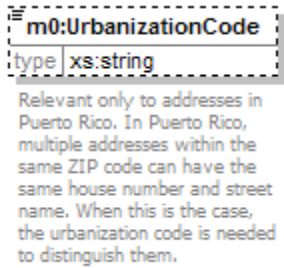
diagram	
type	<b>xs:string</b>
properties	<pre> isRef 0 minOcc 0 maxOcc 1 content simple </pre>
source	<pre> &lt;xs:element name="StateOrProvinceCode" type="xs:string" minOccurs="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Identifying abbreviation for US state, Canada province, etc. Format and     presence of this field will vary, depending on country.&lt;/xs:documentation&gt;     &lt;xs:appinfo&gt;       &lt;MaxLength&gt;14&lt;/MaxLength&gt;     &lt;/xs:appinfo&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt; </pre>

**element Address/PostalCode**

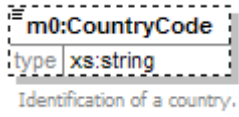
diagram	
type	<b>xs:string</b>
properties	<pre> isRef 0 minOcc 0 maxOcc 1 content simple </pre>
source	<pre> &lt;xs:element name="PostalCode" type="xs:string" minOccurs="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Identification of a region (usually small) for mail/package delivery. Format     and presence of this field will vary, depending on country. This element is required if both the City </pre>

	<p>and StateOrProvinceCode are not present.&lt;/xs:documentation&gt;</p> <pre>&lt;xs:appinfo&gt;   &lt;MaxLength&gt;16&lt;/MaxLength&gt; &lt;/xs:appinfo&gt; &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>
--	---

**element Address/UrbanizationCode**

diagram	 <p>Relevant only to addresses in Puerto Rico. In Puerto Rico, multiple addresses within the same ZIP code can have the same house number and street name. When this is the case, the urbanization code is needed to distinguish them.</p>
type	<b>xs:string</b>
properties	<pre>isRef 0 minOcc 0 maxOcc 1 content simple</pre>
source	<pre>&lt;xs:element name="UrbanizationCode" type="xs:string" minOccurs="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Relevant only to addresses in Puerto Rico. In Puerto Rico, multiple     addresses within the same ZIP code can have the same house number and street name. When this     is the case, the urbanization code is needed to distinguish them.&lt;/xs:documentation&gt;     &lt;xs:appinfo&gt;       &lt;MaxLength&gt;100&lt;/MaxLength&gt;     &lt;/xs:appinfo&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>

**element Address/CountryCode**

diagram	 <p>Identification of a country.</p>
type	<b>xs:string</b>
properties	<pre>isRef 0 minOcc 0 maxOcc 1 content simple</pre>
source	<pre>&lt;xs:element name="CountryCode" type="xs:string" minOccurs="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Identification of a country.&lt;/xs:documentation&gt;     &lt;xs:appinfo&gt;       &lt;MaxLength&gt;2&lt;/MaxLength&gt;     &lt;/xs:appinfo&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>

### element Address/Residential

diagram	
type	<b>xs:boolean</b>
properties	isRef 0 minOcc 0 maxOcc 1 content simple
source	<pre>&lt;xs:element name="Residential" type="xs:boolean" minOccurs="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Indicates whether this address is residential (as opposed to     commercial).&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>

### complexType AddressToValidate

diagram	
children	<a href="#">m0:AddressId</a> <a href="#">m0:CompanyName</a> <a href="#">m0:Address</a>
source	<pre>&lt;xs:complexType name="AddressToValidate"&gt;   &lt;xs:sequence&gt;     &lt;xs:element name="AddressId" type="xs:string" minOccurs="0"/&gt;     &lt;xs:element name="CompanyName" type="xs:string" minOccurs="0"/&gt;     &lt;xs:element name="Address" type="ns:Address"/&gt;   &lt;/xs:sequence&gt; &lt;/xs:complexType&gt;</pre>

### element AddressToValidate/AddressId

diagram	
type	<b>xs:string</b>
properties	isRef 0 minOcc 0 maxOcc 1 content simple
source	<pre>&lt;xs:element name="AddressId" type="xs:string" minOccurs="0"/&gt;</pre>

element **AddressToValidate/CompanyName**

diagram	
type	<b>xs:string</b>
properties	isRef 0 minOcc 0 maxOcc 1 content simple
source	<code>&lt;xs:element name="CompanyName" type="xs:string" minOccurs="0"/&gt;</code>

element **AddressToValidate/Address**

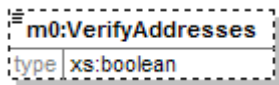
diagram	
type	<b>ns:Address</b>
properties	isRef 0
source	<code>&lt;xs:element name="Address" type="ns:Address"/&gt;</code>

complexType **AddressValidationOptions**

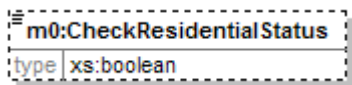
diagram	<p>The diagram shows a complex type <b>AddressValidationOptions</b> with a sequence of elements:</p> <ul style="list-style-type: none"> <li><b>m0:VerifyAddresses</b>: type <code>xs:boolean</code></li> <li><b>m0:CheckResidentialStatus</b>: type <code>xs:boolean</code></li> <li><b>m0:MaximumNumberOfMatches</b>: type <code>xs:positiveInteger</code>, derivedBy <code>restriction</code>, min/maxIncl <code>10</code></li> <li><b>m0:StreetAccuracy</b>: type <code>ns:AddressValidationAccuracy...</code></li> <li><b>m0:DirectionalAccuracy</b>: type <code>ns:AddressValidationAccuracy...</code></li> <li><b>m0:CompanyNameAccuracy</b>: type <code>ns:AddressValidationAccuracy...</code></li> <li><b>m0:ConvertToUpperCase</b>: type <code>xs:boolean</code></li> <li><b>m0:RecognizeAlternateCityNames</b>: type <code>xs:boolean</code></li> <li><b>m0:ReturnParsedElements</b>: type <code>xs:boolean</code></li> </ul>
---------	---

children	<a href="#">m0:VerifyAddresses</a> <a href="#">m0:CheckResidentialStatus</a> <a href="#">m0:MaximumNumberOfMatches</a> <a href="#">m0:StreetAccuracy</a> <a href="#">m0:DirectionalAccuracy</a> <a href="#">m0:CompanyNameAccuracy</a> <a href="#">m0:ConvertToUpperCase</a> <a href="#">m0:RecognizeAlternateCityNames</a> <a href="#">m0:ReturnParsedElements</a>
source	<pre> &lt;xs:complexType name="AddressValidationOptions"&gt;   &lt;xs:sequence&gt;     &lt;xs:element name="VerifyAddresses" type="xs:boolean" minOccurs="0"/&gt;     &lt;xs:element name="CheckResidentialStatus" type="xs:boolean" minOccurs="0"/&gt;     &lt;xs:element name="MaximumNumberOfMatches" minOccurs="0"&gt;       &lt;xs:simpleType&gt;         &lt;xs:restriction base="xs:positiveInteger"&gt;           &lt;xs:maxInclusive value="10"/&gt;         &lt;/xs:restriction&gt;       &lt;/xs:simpleType&gt;     &lt;/xs:element&gt;     &lt;xs:element name="StreetAccuracy" type="ns:AddressValidationAccuracyType" minOccurs="0"/&gt;     &lt;xs:element name="DirectionalAccuracy" type="ns:AddressValidationAccuracyType" minOccurs="0"/&gt;     &lt;xs:element name="CompanyNameAccuracy" type="ns:AddressValidationAccuracyType" minOccurs="0"/&gt;     &lt;xs:element name="ConvertToUpperCase" type="xs:boolean" minOccurs="0"/&gt;     &lt;xs:element name="RecognizeAlternateCityNames" type="xs:boolean" minOccurs="0"/&gt;     &lt;xs:element name="ReturnParsedElements" type="xs:boolean" minOccurs="0"/&gt;   &lt;/xs:sequence&gt; &lt;/xs:complexType&gt; </pre>

**element AddressValidationOptions/VerifyAddresses**

diagram									
type	<b>xs:boolean</b>								
properties	<table border="0"> <tr><td>isRef</td><td>0</td></tr> <tr><td>minOcc</td><td>0</td></tr> <tr><td>maxOcc</td><td>1</td></tr> <tr><td>content</td><td>simple</td></tr> </table>	isRef	0	minOcc	0	maxOcc	1	content	simple
isRef	0								
minOcc	0								
maxOcc	1								
content	simple								
source	<code>&lt;xs:element name="VerifyAddresses" type="xs:boolean" minOccurs="0"/&gt;</code>								

**element AddressValidationOptions/CheckResidentialStatus**

diagram									
type	<b>xs:boolean</b>								
properties	<table border="0"> <tr><td>isRef</td><td>0</td></tr> <tr><td>minOcc</td><td>0</td></tr> <tr><td>maxOcc</td><td>1</td></tr> <tr><td>content</td><td>simple</td></tr> </table>	isRef	0	minOcc	0	maxOcc	1	content	simple
isRef	0								
minOcc	0								
maxOcc	1								
content	simple								
source	<code>&lt;xs:element name="CheckResidentialStatus" type="xs:boolean" minOccurs="0"/&gt;</code>								

element **AddressValidationOptions/MaximumNumberOfMatches**

diagram	<pre> m0:MaximumNumberOfMatches ├── type: xs:positiveInteger ├── derivedBy: restriction └── min/maxIncl: 10                     </pre>
type	restriction of <b>xs:positiveInteger</b>
properties	isRef 0 minOcc 0 maxOcc 1 content simple
source	<pre> &lt;xs:element name="MaximumNumberOfMatches" minOccurs="0"&gt;   &lt;xs:simpleType&gt;     &lt;xs:restriction base="xs:positiveInteger"&gt;       &lt;xs:maxInclusive value="10"/&gt;     &lt;/xs:restriction&gt;   &lt;/xs:simpleType&gt; &lt;/xs:element&gt;                     </pre>

element **AddressValidationOptions/StreetAccuracy**

diagram	<pre> m0:StreetAccuracy ├── type: ns:AddressValidationAccuracy...                     </pre>
type	<b>ns:AddressValidationAccuracyType</b>
properties	isRef 0 minOcc 0 maxOcc 1
source	<pre> &lt;xs:element name="StreetAccuracy" type="ns:AddressValidationAccuracyType" minOccurs="0"/&gt;                     </pre>

element **AddressValidationOptions/DirectionalAccuracy**

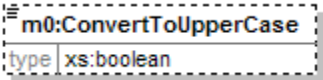
diagram	<pre> m0:DirectionalAccuracy ├── type: ns:AddressValidationAccuracy...                     </pre>
type	<b>ns:AddressValidationAccuracyType</b>
properties	isRef 0 minOcc 0 maxOcc 1
source	<pre> &lt;xs:element name="DirectionalAccuracy" type="ns:AddressValidationAccuracyType" minOccurs="0"/&gt;                     </pre>

element **AddressValidationOptions/CompanyNameAccuracy**

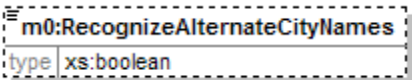
diagram	<pre> m0:CompanyNameAccuracy ├── type: ns:AddressValidationAccuracy...                     </pre>
type	<b>ns:AddressValidationAccuracyType</b>

properties	isRef 0 minOcc 0 maxOcc 1
source	<xs:element name="CompanyNameAccuracy" type="ns:AddressValidationAccuracyType" minOccurs="0"/>

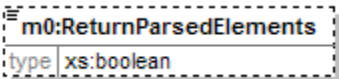
#### element AddressValidationOptions/ConvertToUpperCase

diagram	
type	xs:boolean
properties	isRef 0 minOcc 0 maxOcc 1 content simple
source	<xs:element name="ConvertToUpperCase" type="xs:boolean" minOccurs="0"/>

#### element AddressValidationOptions/RecognizeAlternateCityNames

diagram	
type	xs:boolean
properties	isRef 0 minOcc 0 maxOcc 1 content simple
source	<xs:element name="RecognizeAlternateCityNames" type="xs:boolean" minOccurs="0"/>

#### element AddressValidationOptions/ReturnParsedElements

diagram	
type	xs:boolean
properties	isRef 0 minOcc 0 maxOcc 1 content simple
source	<xs:element name="ReturnParsedElements" type="xs:boolean" minOccurs="0"/>

complexType **AddressValidationReply**

<p>diagram</p>	
<p>children</p>	<p><a href="#">m0:HighestSeverity</a> <a href="#">m0:Notifications</a> <a href="#">m0:TransactionDetail</a> <a href="#">m0:ReplyTimestamp</a> <a href="#">m0:AddressResults</a> <a href="#">m0:Version</a></p>
<p>source</p>	<pre>&lt;xs:complexType name="AddressValidationReply"&gt;   &lt;xs:sequence&gt;     &lt;xs:element name="HighestSeverity" type="ns:NotificationSeverityType"/&gt;     &lt;xs:element name="Notifications" type="ns:Notification" maxOccurs="unbounded"/&gt;     &lt;xs:element name="TransactionDetail" type="ns:TransactionDetail" minOccurs="0"/&gt;     &lt;xs:element name="ReplyTimestamp" type="xs:dateTime"/&gt;     &lt;xs:element name="AddressResults" type="ns:AddressValidationResult" maxOccurs="100"/&gt;     &lt;xs:element name="Version" type="ns:VersionId" minOccurs="0"/&gt;   &lt;/xs:sequence&gt; &lt;/xs:complexType&gt;</pre>

element **AddressValidationReply/HighestSeverity**

<p>diagram</p>	
<p>type</p>	<p>ns:NotificationSeverityType</p>
<p>properties</p>	<p>isRef 0</p>
<p>source</p>	<pre>&lt;xs:element name="HighestSeverity" type="ns:NotificationSeverityType"/&gt;</pre>

element **AddressValidationReply/Notifications**

<p>diagram</p>	
----------------	--

type	<b>ns:Notification</b>
properties	isRef 0 minOcc 1 maxOcc unbounded
source	<code>&lt;xs:element name="Notifications" type="ns:Notification" maxOccurs="unbounded"/&gt;</code>

#### element AddressValidationReply/TransactionDetail

diagram	
type	<b>ns:TransactionDetail</b>
properties	isRef 0 minOcc 0 maxOcc 1
source	<code>&lt;xs:element name="TransactionDetail" type="ns:TransactionDetail" minOccurs="0"/&gt;</code>

#### element AddressValidationReply/ReplyTimestamp

diagram	
type	<b>xs:dateTime</b>
properties	isRef 0 content simple
source	<code>&lt;xs:element name="ReplyTimestamp" type="xs:dateTime"/&gt;</code>

#### element AddressValidationReply/AddressResults

diagram	
type	<b>ns:AddressValidationResult</b>
properties	isRef 0 minOcc 1 maxOcc 100
source	<code>&lt;xs:element name="AddressResults" type="ns:AddressValidationResult" maxOccurs="100"/&gt;</code>

#### element AddressValidationReply/Version


diagram	
type	<b>ns:VersionId</b>
properties	isRef 0 minOcc 0 maxOcc 1
source	<code>&lt;xs:element name="Version" type="ns:VersionId" minOccurs="0"/&gt;</code>

## complexType AddressValidationRequest


<p>diagram</p>	<div style="display: flex; flex-direction: column; align-items: flex-end;"> <div style="border: 1px solid black; padding: 2px; margin-bottom: 5px;"> <b>m0:WebAuthenticationDetail</b>              type   ns:WebAuthenticationDetail              The descriptive data to be used in authentication of the sender's identity (and right to use FedEx web services).         </div> <div style="border: 1px solid black; padding: 2px; margin-bottom: 5px;"> <b>m0:ClientDetail</b>              type   ns:ClientDetail              Descriptive data identifying the client submitting the transaction.         </div> <div style="border: 1px dashed black; padding: 2px; margin-bottom: 5px;"> <b>m0:TransactionDetail</b>              type   ns:TransactionDetail              Descriptive data for this customer transaction. The TransactionDetail from the request is echoed back to the caller in the corresponding reply.         </div> <div style="border: 1px solid black; padding: 2px; margin-bottom: 5px;"> <b>m0:Version</b>              type   ns:VersionId              Identifies the version/level of a service operation expected by a caller (in each request) and performed by the callee (in each reply).         </div> <div style="border: 1px solid black; padding: 2px; margin-bottom: 5px;"> <b>m0:RequestTimestamp</b>              type   xs:dateTime         </div> <div style="border: 1px solid black; padding: 2px; margin-bottom: 5px;"> <b>m0:Options</b>              type   ns:AddressValidationOptions         </div> <div style="border: 1px solid black; padding: 2px; margin-bottom: 5px;"> <b>m0:AddressesToValidate</b>              type   ns:AddressToValidate              1..100         </div> </div>
<p>children</p>	<p><a href="#">m0:WebAuthenticationDetail</a> <a href="#">m0:ClientDetail</a> <a href="#">m0:TransactionDetail</a> <a href="#">m0:Version</a> <a href="#">m0:RequestTimestamp</a> <a href="#">m0:Options</a> <a href="#">m0:AddressesToValidate</a></p>
<p>source</p>	<pre> &lt;xs:complexType name="AddressValidationRequest"&gt;   &lt;xs:sequence&gt;     &lt;xs:element name="WebAuthenticationDetail" type="ns:WebAuthenticationDetail"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;The descriptive data to be used in authentication of the sender's identity         (and right to use FedEx web services).&lt;/xs:documentation&gt;       &lt;/xs:annotation&gt;     &lt;/xs:element&gt;     &lt;xs:element name="ClientDetail" type="ns:ClientDetail"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;Descriptive data identifying the client submitting the         transaction.&lt;/xs:documentation&gt;       &lt;/xs:annotation&gt;     &lt;/xs:element&gt;   &lt;/xs:sequence&gt; &lt;/xs:complexType&gt; </pre>

	<pre> &lt;xs:element name="TransactionDetail" type="ns:TransactionDetail" minOccurs="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Descriptive data for this customer transaction. The TransactionDetail from the request is echoed back to the caller in the corresponding reply.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt; &lt;xs:element name="Version" type="ns:VersionId"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Identifies the version/level of a service operation expected by a caller (in each request) and performed by the callee (in each reply).&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt; &lt;xs:element name="RequestTimestamp" type="xs:date"/&gt; &lt;xs:element name="Options" type="ns:AddressValidationOptions"/&gt; &lt;xs:element name="AddressesToValidate" type="ns:AddressToValidate" maxOccurs="100"/&gt; &lt;/xs:sequence&gt; &lt;/xs:complexType&gt; </pre>
--	---


#### element AddressValidationRequest/WebAuthenticationDetail

diagram	
type	ns:WebAuthenticationDetail
properties	isRef 0
source	<pre> &lt;xs:element name="WebAuthenticationDetail" type="ns:WebAuthenticationDetail"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;The descriptive data to be used in authentication of the sender's identity (and right to use FedEx web services).&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt; </pre>


#### element AddressValidationRequest/ClientDetail

diagram	
type	ns:ClientDetail
properties	isRef 0
source	<pre> &lt;xs:element name="ClientDetail" type="ns:ClientDetail"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Descriptive data identifying the client submitting the transaction.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt; </pre>


### element AddressValidationRequest/TransactionDetail

diagram	 <p>Descriptive data for this customer transaction. The TransactionDetail from the request is echoed back to the caller in the corresponding reply.</p>
type	ns:TransactionDetail
properties	isRef 0 minOcc 0 maxOcc 1
source	<pre>&lt;xs:element name="TransactionDetail" type="ns:TransactionDetail" minOccurs="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Descriptive data for this customer transaction. The TransactionDetail from the request is echoed back to the caller in the corresponding reply.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>


### element AddressValidationRequest/Version

diagram	 <p>Identifies the version/level of a service operation expected by a caller (in each request) and performed by the callee (in each reply).</p>
type	ns:VersionId
properties	isRef 0
source	<pre>&lt;xs:element name="Version" type="ns:VersionId"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Identifies the version/level of a service operation expected by a caller (in each request) and performed by the callee (in each reply).&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>


### element AddressValidationRequest/RequestTimestamp

diagram	
type	xs:dateTime
properties	isRef 0 content simple
source	<pre>&lt;xs:element name="RequestTimestamp" type="xs:dateTime"/&gt;</pre>

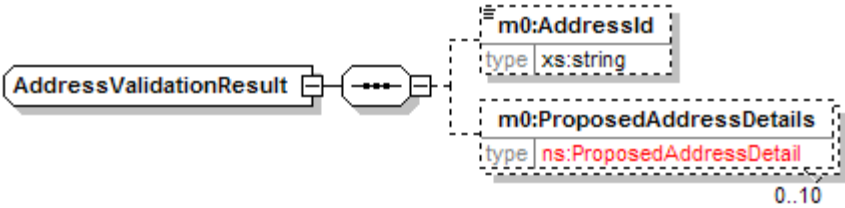
**element AddressValidationRequest/Options**

diagram	
type	ns:AddressValidationOptions
properties	isRef 0
source	<xs:element name="Options" type="ns:AddressValidationOptions"/>

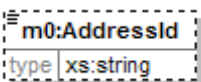
**element AddressValidationRequest/AddressesToValidate**

diagram	
type	ns:AddressToValidate
properties	isRef 0 minOcc 1 maxOcc 100
source	<xs:element name="AddressesToValidate" type="ns:AddressToValidate" maxOccurs="100"/>

**complexType AddressValidationResult**

diagram	
children	<a href="#">m0:AddressId</a> <a href="#">m0:ProposedAddressDetails</a>
source	<pre>&lt;xs:complexType name="AddressValidationResult"&gt;   &lt;xs:sequence&gt;     &lt;xs:element name="AddressId" type="xs:string" minOccurs="0"/&gt;     &lt;xs:element name="ProposedAddressDetails" type="ns:ProposedAddressDetail" minOccurs="0" maxOccurs="10"/&gt;   &lt;/xs:sequence&gt; &lt;/xs:complexType&gt;</pre>

**element AddressValidationResult/AddressId**

diagram	
type	xs:string

properties	isRef 0 minOcc 0 maxOcc 1 content simple
source	<xs:element name="AddressId" type="xs:string" minOccurs="0"/>

**element AddressValidationResult/ProposedAddressDetails**

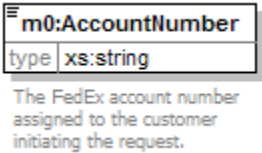
diagram	<p>The diagram shows a dashed box containing the text: <b>m0:ProposedAddressDetails</b>, type ns:ProposedAddressDetail, and 0..10 below it.</p>
type	ns:ProposedAddressDetail
properties	isRef 0 minOcc 0 maxOcc 10
source	<xs:element name="ProposedAddressDetails" type="ns:ProposedAddressDetail" minOccurs="0" maxOccurs="10"/>

**complexType ClientDetail**

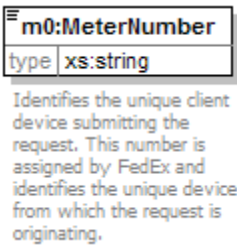
diagram	<p>The diagram shows a complex type <b>ClientDetail</b> with a description: "The descriptive data identifying the client submitting the transaction." It is connected to three child elements: <b>m0:AccountNumber</b> (type xs:string, description: "The FedEx account number assigned to the customer initiating the request."), <b>m0:MeterNumber</b> (type xs:string, description: "Identifies the unique client device submitting the request. This number is assigned by FedEx and identifies the unique device from which the request is originating."), and <b>m0:Localization</b> (type ns:Localization, description: "Governs any future language/translations used for human-readable Notification.localizedMessages in responses to the request containing this ClientDetail object. Different requests from the same client may contain different Localization data. (Contrast with TransactionDetail.localization, which governs data payload language/translation.)").</p>
children	<a href="#">m0:AccountNumber</a> <a href="#">m0:MeterNumber</a> <a href="#">m0:Localization</a>
source	<xs:complexType name="ClientDetail"> <xs:annotation>

	<pre> &lt;xs:documentation&gt;The descriptive data identifying the client submitting the transaction.&lt;/xs:documentation&gt; &lt;/xs:annotation&gt; &lt;xs:sequence&gt;   &lt;xs:element name="AccountNumber" type="xs:string"&gt;     &lt;xs:annotation&gt;       &lt;xs:documentation&gt;The FedEx account number assigned to the customer initiating the request.&lt;/xs:documentation&gt;       &lt;xs:appinfo&gt;         &lt;MaxLength&gt;12&lt;/MaxLength&gt;       &lt;/xs:appinfo&gt;     &lt;/xs:annotation&gt;   &lt;/xs:element&gt;   &lt;xs:element name="MeterNumber" type="xs:string"&gt;     &lt;xs:annotation&gt;       &lt;xs:documentation&gt;Identifies the unique client device submitting the request. This number is assigned by FedEx and identifies the unique device from which the request is originating.&lt;/xs:documentation&gt;       &lt;xs:appinfo&gt;         &lt;MaxLength&gt;10&lt;/MaxLength&gt;       &lt;/xs:appinfo&gt;     &lt;/xs:annotation&gt;   &lt;/xs:element&gt;   &lt;xs:element name="Localization" type="ns:Localization" minOccurs="0"&gt;     &lt;xs:annotation&gt;       &lt;xs:documentation&gt;Governs any future language/translations used for human-readable Notification.localizedMessages in responses to the request containing this ClientDetail object. Different requests from the same client may contain different Localization data. (Contrast with TransactionDetail.localization, which governs data payload language/translation.)&lt;/xs:documentation&gt;     &lt;/xs:annotation&gt;   &lt;/xs:element&gt; &lt;/xs:sequence&gt; &lt;/xs:complexType&gt; </pre>
--	---

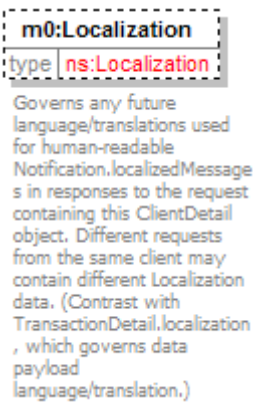
### element ClientDetail/AccountNumber

diagram	
type	<b>xs:string</b>
properties	isRef 0 content simple
source	<pre> &lt;xs:element name="AccountNumber" type="xs:string"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;The FedEx account number assigned to the customer initiating the request.&lt;/xs:documentation&gt;     &lt;xs:appinfo&gt;       &lt;MaxLength&gt;12&lt;/MaxLength&gt;     &lt;/xs:appinfo&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt; </pre>

### element ClientDetail/MeterNumber

diagram	 <p>Identifies the unique client device submitting the request. This number is assigned by FedEx and identifies the unique device from which the request is originating.</p>
type	<b>xs:string</b>
properties	isRef 0 content simple
source	<pre>&lt;xs:element name="MeterNumber" type="xs:string"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Identifies the unique client device submitting the request. This number is assigned by FedEx and identifies the unique device from which the request is originating.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt;   &lt;xs:appinfo&gt;     &lt;MaxLength&gt;10&lt;/MaxLength&gt;   &lt;/xs:appinfo&gt; &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>

### element ClientDetail/Localization

diagram	 <p>Governs any future language/translations used for human-readable Notification.localizedMessages in responses to the request containing this ClientDetail object. Different requests from the same client may contain different Localization data. (Contrast with TransactionDetail.localization, which governs data payload language/translation.)</p>
type	<b>ns:Localization</b>
properties	isRef 0 minOcc 0 maxOcc 1
source	<pre>&lt;xs:element name="Localization" type="ns:Localization" minOccurs="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Governs any future language/translations used for human-readable Notification.localizedMessages in responses to the request containing this ClientDetail object. Different requests from the same client may contain different Localization data. (Contrast with TransactionDetail.localization, which governs data payload language/translation.)&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>

## complexType Localization

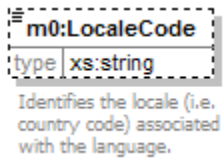
diagram	
children	<a href="#">m0:LanguageCode</a> <a href="#">m0:LocaleCode</a>
source	<pre> &lt;xs:complexType name="Localization"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Governs any future language/translations used for human-readable text.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt;   &lt;xs:sequence&gt;     &lt;xs:element name="LanguageCode" type="xs:string"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;Identifies the language to use for human-readable messages.&lt;/xs:documentation&gt;         &lt;xs:appinfo&gt;           &lt;MaxLength&gt;2&lt;/MaxLength&gt;         &lt;/xs:appinfo&gt;       &lt;/xs:annotation&gt;     &lt;/xs:element&gt;     &lt;xs:element name="LocaleCode" type="xs:string" minOccurs="0"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;Identifies the locale (i.e. country code) associated with the language.&lt;/xs:documentation&gt;         &lt;xs:appinfo&gt;           &lt;MaxLength&gt;2&lt;/MaxLength&gt;         &lt;/xs:appinfo&gt;       &lt;/xs:annotation&gt;     &lt;/xs:element&gt;   &lt;/xs:sequence&gt; &lt;/xs:complexType&gt; </pre>

## element Localization/LanguageCode

diagram	
type	<b>xs:string</b>
properties	isRef 0 content simple

source	<pre> &lt;xs:element name="LanguageCode" type="xs:string"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Identifies the language to use for human-readable messages.&lt;/xs:documentation&gt;     &lt;xs:appinfo&gt;       &lt;MaxLength&gt;2&lt;/MaxLength&gt;     &lt;/xs:appinfo&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt; </pre>
--------	--

### element Localization/LocaleCode

diagram	
type	<b>xs:string</b>
properties	<pre> isRef 0 minOcc 0 maxOcc 1 content simple </pre>
source	<pre> &lt;xs:element name="LocaleCode" type="xs:string" minOccurs="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Identifies the locale (i.e. country code) associated with the language.&lt;/xs:documentation&gt;     &lt;xs:appinfo&gt;       &lt;MaxLength&gt;2&lt;/MaxLength&gt;     &lt;/xs:appinfo&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt; </pre>

## complexType Notification

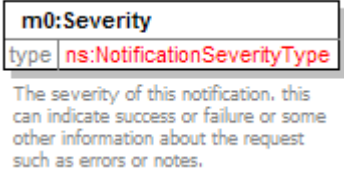
<p>diagram</p>	<p><b>Notification</b> The descriptive data regarding the results of the submitted transaction.</p> <p><b>m0:Severity</b> type <code>ns:NotificationSeverityType</code> The severity of this notification. this can indicate success or failure or some other information about the request such as errors or notes.</p> <p><b>m0:Source</b> type <code>xs:string</code> Indicates the source of the notification. Combined with Code, it uniquely identifies this message.</p> <p><b>m0:Code</b> type <code>xs:string</code> A code that represents this notification. Combined with Source, it uniquely identifies this message.</p> <p><b>m0:Message</b> type <code>xs:string</code> Text that explains this notification.</p> <p><b>m0:LocalizedMessage</b> type <code>xs:string</code> A translated message. The translation is based on the Localization element of the ClientDetail element of the request.</p> <p><b>m0:MessageParameters</b> type <code>ns:NotificationParameter</code> 0..∞ If the message used parameter replacement to be specific as to the meaning of the message, this is the list of parameters that were used.</p>
<p>children</p>	<p><a href="#">m0:Severity</a> <a href="#">m0:Source</a> <a href="#">m0:Code</a> <a href="#">m0:Message</a> <a href="#">m0:LocalizedMessage</a> <a href="#">m0:MessageParameters</a></p>
<p>source</p>	<pre>&lt;xs:complexType name="Notification"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;The descriptive data regarding the results of the submitted transaction.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt;   &lt;xs:sequence&gt;     &lt;xs:element name="Severity" type="ns:NotificationSeverityType"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;The severity of this notification. this can indicate success or failure or some other information about the request such as errors or notes.&lt;/xs:documentation&gt;       &lt;/xs:annotation&gt;     &lt;/xs:element&gt;     &lt;xs:element name="Source" type="xs:string"&gt;       &lt;xs:annotation&gt;</pre>

```

<xs:documentation>Indicates the source of the notification. Combined with Code, it uniquely
identifies this message.</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="Code" type="xs:string" minOccurs="0">
  <xs:annotation>
    <xs:documentation>A code that represents this notification. Combined with Source, it uniquely
identifies this message.</xs:documentation>
    <xs:appinfo>
      <MaxLength>8</MaxLength>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
<xs:element name="Message" type="xs:string" minOccurs="0">
  <xs:annotation>
    <xs:documentation>Text that explains this notification.</xs:documentation>
    <xs:appinfo>
      <MaxLength>255</MaxLength>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
<xs:element name="LocalizedMessage" type="xs:string" minOccurs="0">
  <xs:annotation>
    <xs:documentation>A translated message. The translation is based on the Localization
element of the ClientDetail element of the request.</xs:documentation>
    <xs:appinfo>
      <MaxLength>TBD</MaxLength>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
<xs:element name="MessageParameters" type="ns:NotificationParameter" minOccurs="0"
maxOccurs="unbounded">
  <xs:annotation>
    <xs:documentation>If the message used parameter replacement to be specific as to the
meaning of the message, this is the list of parameters that were used.</xs:documentation>
    <xs:appinfo>
      <MaxLength>TBD</MaxLength>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>

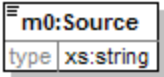
```

**element Notification/Severity**

diagram	
type	<b>ns:NotificationSeverityType</b>
properties	isRef 0

source	<pre>&lt;xs:element name="Severity" type="ns:NotificationSeverityType"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;The severity of this notification. this can indicate success or failure or some other information about the request such as errors or notes.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>
--------	---

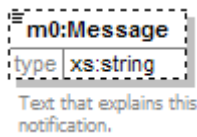
### element Notification/Source

diagram	 <p>Indicates the source of the notification. Combined with Code, it uniquely identifies this message.</p>
type	<b>xs:string</b>
properties	isRef 0 content simple
source	<pre>&lt;xs:element name="Source" type="xs:string"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Indicates the source of the notification. Combined with Code, it uniquely identifies this message.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>

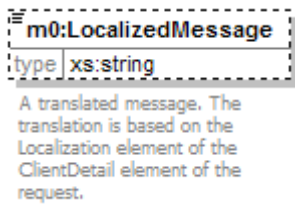
### element Notification/Code

diagram	 <p>A code that represents this notification. Combined with Source, it uniquely identifies this message.</p>
type	<b>xs:string</b>
properties	isRef 0 minOcc 0 maxOcc 1 content simple
source	<pre>&lt;xs:element name="Code" type="xs:string" minOccurs="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;A code that represents this notification. Combined with Source, it uniquely identifies this message.&lt;/xs:documentation&gt;   &lt;xs:appinfo&gt;     &lt;MaxLength&gt;8&lt;/MaxLength&gt;   &lt;/xs:appinfo&gt; &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>

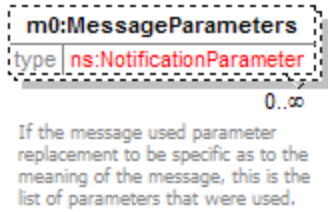
### element Notification/Message

diagram	
type	<b>xs:string</b>
properties	isRef 0 minOcc 0 maxOcc 1 content simple
source	<pre>&lt;xs:element name="Message" type="xs:string" minOccurs="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Text that explains this notification.&lt;/xs:documentation&gt;     &lt;xs:appinfo&gt;       &lt;MaxLength&gt;255&lt;/MaxLength&gt;     &lt;/xs:appinfo&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>

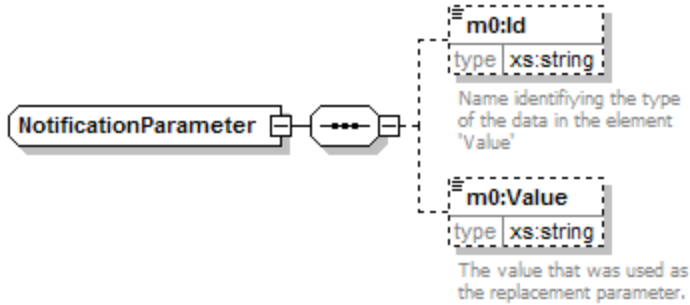
### element Notification/LocalizedMessage

diagram	
type	<b>xs:string</b>
properties	isRef 0 minOcc 0 maxOcc 1 content simple
source	<pre>&lt;xs:element name="LocalizedMessage" type="xs:string" minOccurs="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;A translated message. The translation is based on the Localization element of the ClientDetail element of the request.&lt;/xs:documentation&gt;     &lt;xs:appinfo&gt;       &lt;MaxLength&gt;TBD&lt;/MaxLength&gt;     &lt;/xs:appinfo&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>

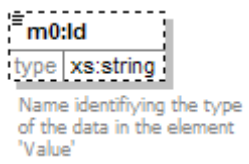
## element Notification/MessageParameters

diagram	 <p>If the message used parameter replacement to be specific as to the meaning of the message, this is the list of parameters that were used.</p>
type	ns:NotificationParameter
properties	isRef 0 minOcc 0 maxOcc unbounded
source	<pre> &lt;xs:element name="MessageParameters" type="ns:NotificationParameter" minOccurs="0" maxOccurs="unbounded"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;If the message used parameter replacement to be specific as to the meaning of the message, this is the list of parameters that were used.&lt;/xs:documentation&gt;   &lt;xs:appinfo&gt;     &lt;MaxLength&gt;TBD&lt;/MaxLength&gt;   &lt;/xs:appinfo&gt; &lt;/xs:annotation&gt; &lt;/xs:element&gt; </pre>

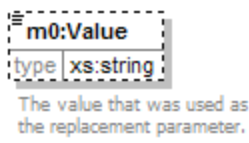
## complexType NotificationParameter

diagram	
children	<a href="#">m0:Id</a> <a href="#">m0:Value</a>
source	<pre> &lt;xs:complexType name="NotificationParameter"&gt;   &lt;xs:sequence&gt;     &lt;xs:element name="Id" type="xs:string" minOccurs="0"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;Name identifying the type of the data in the element 'Value'&lt;/xs:documentation&gt;       &lt;/xs:annotation&gt;     &lt;/xs:element&gt;     &lt;xs:element name="Value" type="xs:string" minOccurs="0"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;The value that was used as the replacement parameter.&lt;/xs:documentation&gt;       &lt;/xs:annotation&gt;     &lt;/xs:element&gt;   &lt;/xs:sequence&gt; &lt;/xs:complexType&gt; </pre>

### element NotificationParameter/Id

diagram	
type	<b>xs:string</b>
properties	isRef 0 minOcc 0 maxOcc 1 content simple
source	<pre>&lt;xs:element name="Id" type="xs:string" minOccurs="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Name identifying the type of the data in the element     'Value'&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>

### element NotificationParameter/Value

diagram	
type	<b>xs:string</b>
properties	isRef 0 minOcc 0 maxOcc 1 content simple
source	<pre>&lt;xs:element name="Value" type="xs:string" minOccurs="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;The value that was used as the replacement parameter.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>

### complexType **ParsedAddress**

diagram	
children	<a href="#">m0:ParsedUrbanizationCode</a> <a href="#">m0:ParsedStreetLine</a> <a href="#">m0:ParsedCity</a> <a href="#">m0:ParsedStateOrProvinceCode</a> <a href="#">m0:ParsedPostalCode</a> <a href="#">m0:ParsedCountryCode</a>
source	<pre>&lt;xs:complexType name="ParsedAddress"&gt;   &lt;xs:sequence&gt;     &lt;xs:element name="ParsedUrbanizationCode" type="ns:ParsedAddressPart" minOccurs="0"/&gt;     &lt;xs:element name="ParsedStreetLine" type="ns:ParsedAddressPart" minOccurs="0" maxOccurs="4"/&gt;     &lt;xs:element name="ParsedCity" type="ns:ParsedAddressPart" minOccurs="0"/&gt;     &lt;xs:element name="ParsedStateOrProvinceCode" type="ns:ParsedAddressPart" minOccurs="0"/&gt;     &lt;xs:element name="ParsedPostalCode" type="ns:ParsedAddressPart" minOccurs="0"/&gt;     &lt;xs:element name="ParsedCountryCode" type="ns:ParsedAddressPart" minOccurs="0"/&gt;   &lt;/xs:sequence&gt; &lt;/xs:complexType&gt;</pre>

### element **ParsedAddress/ParsedUrbanizationCode**

diagram	
type	<b>ns:ParsedAddressPart</b>
properties	isRef 0 minOcc 0 maxOcc 1
source	<pre>&lt;xs:element name="ParsedUrbanizationCode" type="ns:ParsedAddressPart" minOccurs="0"/&gt;</pre>

element **ParsedAddress/ParsedStreetLine**

diagram	<p>The diagram shows a dashed box containing the text 'm0:ParsedStreetLine' above 'type ns:ParsedAddressPart'. Below the box is the cardinality '0..4'.</p>
type	<b>ns:ParsedAddressPart</b>
properties	isRef 0 minOcc 0 maxOcc 4
source	<pre>&lt;xs:element name="ParsedStreetLine" type="ns:ParsedAddressPart" minOccurs="0" maxOccurs="4"/&gt;</pre>

element **ParsedAddress/ParsedCity**

diagram	<p>The diagram shows a dashed box containing the text 'm0:ParsedCity' above 'type ns:ParsedAddressPart'.</p>
type	<b>ns:ParsedAddressPart</b>
properties	isRef 0 minOcc 0 maxOcc 1
source	<pre>&lt;xs:element name="ParsedCity" type="ns:ParsedAddressPart" minOccurs="0"/&gt;</pre>

element **ParsedAddress/ParsedStateOrProvinceCode**

diagram	<p>The diagram shows a dashed box containing the text 'm0:ParsedStateOrProvinceCode' above 'type ns:ParsedAddressPart'.</p>
type	<b>ns:ParsedAddressPart</b>
properties	isRef 0 minOcc 0 maxOcc 1
source	<pre>&lt;xs:element name="ParsedStateOrProvinceCode" type="ns:ParsedAddressPart" minOccurs="0"/&gt;</pre>

element **ParsedAddress/ParsedPostalCode**

diagram	<p>The diagram shows a dashed box containing the text 'm0:ParsedPostalCode' above 'type ns:ParsedAddressPart'.</p>
type	<b>ns:ParsedAddressPart</b>
properties	isRef 0 minOcc 0 maxOcc 1
source	<pre>&lt;xs:element name="ParsedPostalCode" type="ns:ParsedAddressPart" minOccurs="0"/&gt;</pre>

### element `ParsedAddress/ParsedCountryCode`

diagram	
type	<code>ns:ParsedAddressPart</code>
properties	isRef 0 minOcc 0 maxOcc 1
source	<code>&lt;xs:element name="ParsedCountryCode" type="ns:ParsedAddressPart" minOccurs="0"/&gt;</code>

### complexType `ParsedAddressPart`

diagram	
children	<a href="#">m0:Elements</a>
source	<code>&lt;xs:complexType name="ParsedAddressPart"&gt;</code> <code>&lt;xs:sequence&gt;</code> <code>&lt;xs:element name="Elements" type="ns:ParsedElement" maxOccurs="unbounded"/&gt;</code> <code>&lt;/xs:sequence&gt;</code> <code>&lt;/xs:complexType&gt;</code>

### element `ParsedAddressPart/Elements`


diagram	
type	<code>ns:ParsedElement</code>
properties	isRef 0 minOcc 1 maxOcc unbounded
source	<code>&lt;xs:element name="Elements" type="ns:ParsedElement" maxOccurs="unbounded"/&gt;</code>

### complexType `ParsedElement`

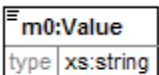
diagram	
children	<a href="#">m0:Name</a> <a href="#">m0:Value</a> <a href="#">m0:Changes</a>

source	<pre> &lt;xs:complexType name="ParsedElement"&gt;   &lt;xs:sequence&gt;     &lt;xs:element name="Name" type="xs:string"/&gt;     &lt;xs:element name="Value" type="xs:string"/&gt;     &lt;xs:element name="Changes" type="ns:AddressValidationChangeType" maxOccurs="unbounded"/&gt;   &lt;/xs:sequence&gt; &lt;/xs:complexType&gt; </pre>
--------	---

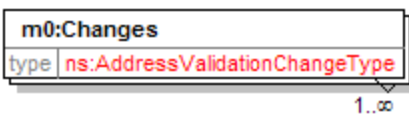
#### element ParsedElement/Name

diagram	
type	<b>xs:string</b>
properties	isRef 0 content simple
source	<pre>&lt;xs:element name="Name" type="xs:string"/&gt;</pre>

#### element ParsedElement/Value

diagram	
type	<b>xs:string</b>
properties	isRef 0 content simple
source	<pre>&lt;xs:element name="Value" type="xs:string"/&gt;</pre>

#### element ParsedElement/Changes

diagram	
type	<b>ns:AddressValidationChangeType</b>
properties	isRef 0 minOcc 1 maxOcc unbounded
source	<pre> &lt;xs:element name="Changes" type="ns:AddressValidationChangeType" maxOccurs="unbounded"/&gt; </pre>

### complexType ProposedAddressDetail

<p>diagram</p>	
<p>children</p>	<p><a href="#">m0:Score</a> <a href="#">m0:Changes</a> <a href="#">m0:ResidentialStatus</a> <a href="#">m0:DeliveryPointValidation</a> <a href="#">m0:CompanyName</a> <a href="#">m0:Address</a> <a href="#">m0:ParsedCompanyName</a> <a href="#">m0:ParsedAddress</a> <a href="#">m0:RemovedNonAddressData</a></p>
<p>source</p>	<pre>&lt;xs:complexType name="ProposedAddressDetail"&gt;   &lt;xs:sequence&gt;     &lt;xs:element name="Score" type="xs:integer" minOccurs="0"/&gt;     &lt;xs:element name="Changes" type="ns:AddressValidationChangeType" maxOccurs="unbounded"/&gt;     &lt;xs:element name="ResidentialStatus" type="ns:ResidentialStatusType" minOccurs="0"/&gt;     &lt;xs:element name="DeliveryPointValidation" type="ns:DeliveryPointValidationType" minOccurs="0"/&gt;     &lt;xs:element name="CompanyName" type="xs:string" minOccurs="0"/&gt;     &lt;xs:element name="Address" type="ns:Address"/&gt;     &lt;xs:element name="ParsedCompanyName" type="ns:ParsedAddressPart" minOccurs="0"/&gt;     &lt;xs:element name="ParsedAddress" type="ns:ParsedAddress" minOccurs="0"/&gt;     &lt;xs:element name="RemovedNonAddressData" type="xs:string" minOccurs="0"/&gt;   &lt;/xs:sequence&gt; &lt;/xs:complexType&gt;</pre>

### element ProposedAddressDetail/Score

<p>diagram</p>	
----------------	--

type	<b>xs:integer</b>
properties	isRef 0 minOcc 0 maxOcc 1 content simple
source	<code>&lt;xs:element name="Score" type="xs:integer" minOccurs="0"/&gt;</code>

#### element ProposedAddressDetail/Changes

diagram	<p>The diagram shows a box labeled 'm0:Changes' with a sub-label 'type   ns:AddressValidationChangeType' and a cardinality '1..∞' below it.</p>
type	<b>ns:AddressValidationChangeType</b>
properties	isRef 0 minOcc 1 maxOcc unbounded
source	<code>&lt;xs:element name="Changes" type="ns:AddressValidationChangeType" maxOccurs="unbounded"/&gt;</code>

#### element ProposedAddressDetail/ResidentialStatus

diagram	<p>The diagram shows a dashed box labeled 'm0:ResidentialStatus' with a sub-label 'type   ns:ResidentialStatusType' and a cardinality '0..1' below it.</p>
type	<b>ns:ResidentialStatusType</b>
properties	isRef 0 minOcc 0 maxOcc 1
source	<code>&lt;xs:element name="ResidentialStatus" type="ns:ResidentialStatusType" minOccurs="0"/&gt;</code>

#### element ProposedAddressDetail/DeliveryPointValidation

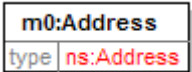
diagram	<p>The diagram shows a dashed box labeled 'm0:DeliveryPointValidation' with a sub-label 'type   ns:DeliveryPointValidationType' and a cardinality '0..1' below it.</p>
type	<b>ns:DeliveryPointValidationType</b>
properties	isRef 0 minOcc 0 maxOcc 1
source	<code>&lt;xs:element name="DeliveryPointValidation" type="ns:DeliveryPointValidationType" minOccurs="0"/&gt;</code>

#### element ProposedAddressDetail/CompanyName

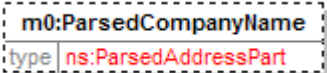
diagram	<p>The diagram shows a dashed box labeled 'm0:CompanyName' with a sub-label 'type   xs:string' and a cardinality '0..1' below it.</p>
---------	---

type	<b>xs:string</b>
properties	isRef 0 minOcc 0 maxOcc 1 content simple
source	<code>&lt;xs:element name="CompanyName" type="xs:string" minOccurs="0"/&gt;</code>

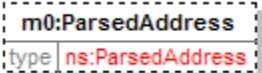
#### element ProposedAddressDetail/Address

diagram	
type	<b>ns:Address</b>
properties	isRef 0
source	<code>&lt;xs:element name="Address" type="ns:Address"/&gt;</code>

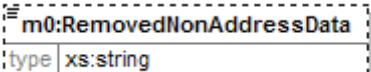
#### element ProposedAddressDetail/ParsedCompanyName

diagram	
type	<b>ns:ParsedAddressPart</b>
properties	isRef 0 minOcc 0 maxOcc 1
source	<code>&lt;xs:element name="ParsedCompanyName" type="ns:ParsedAddressPart" minOccurs="0"/&gt;</code>

#### element ProposedAddressDetail/ParsedAddress

diagram	
type	<b>ns:ParsedAddress</b>
properties	isRef 0 minOcc 0 maxOcc 1
source	<code>&lt;xs:element name="ParsedAddress" type="ns:ParsedAddress" minOccurs="0"/&gt;</code>

#### element ProposedAddressDetail/RemovedNonAddressData

diagram	
type	<b>xs:string</b>
properties	isRef 0 minOcc 0 maxOcc 1 content simple
source	<code>&lt;xs:element name="RemovedNonAddressData" type="xs:string" minOccurs="0"/&gt;</code>

## complexType TransactionDetail

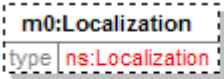
<p>diagram</p>	<p>Descriptive data for this customer transaction. The TransactionDetail from the request is echoed back to the caller in the corresponding reply.</p> <p><b>m0:CustomerTransactionId</b> type   xs:string Identifies a customer-supplied unique identifier for this transaction. It is returned in the reply message to aid in matching requests to replies.</p> <p><b>m0:Localization</b> type   ns:Localization Governs any future language/translations applied to the data payload(contrasted with ClientDetail.localization, which governs Notification.localizedMessage language selection).</p>
<p>children</p>	<p><a href="#">m0:CustomerTransactionId</a> <a href="#">m0:Localization</a></p>
<p>source</p>	<pre>&lt;xs:complexType name="TransactionDetail"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Descriptive data for this customer transaction. The TransactionDetail from the request is echoed back to the caller in the corresponding reply.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt;   &lt;xs:sequence&gt;     &lt;xs:element name="CustomerTransactionId" type="xs:string" minOccurs="0"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;Identifies a customer-supplied unique identifier for this transaction. It is returned in the reply message to aid in matching requests to replies.&lt;/xs:documentation&gt;         &lt;xs:appinfo&gt;           &lt;MaxLength&gt;40&lt;/MaxLength&gt;         &lt;/xs:appinfo&gt;       &lt;/xs:annotation&gt;     &lt;/xs:element&gt;     &lt;xs:element name="Localization" type="ns:Localization" minOccurs="0"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;Governs any future language/translations applied to the data payload(contrasted with ClientDetail.localization, which governs Notification.localizedMessage language selection).&lt;/xs:documentation&gt;       &lt;/xs:annotation&gt;     &lt;/xs:element&gt;   &lt;/xs:sequence&gt; &lt;/xs:complexType&gt;</pre>

## element TransactionDetail/CustomerTransactionId

<p>diagram</p>	<p><b>m0:CustomerTransactionId</b> type   xs:string Identifies a customer-supplied unique identifier for this transaction. It is returned in the reply message to aid in matching requests to replies.</p>
<p>type</p>	<p><b>xs:string</b></p>

properties	isRef 0 minOcc 0 maxOcc 1 content simple
source	<pre>&lt;xs:element name="CustomerTransactionId" type="xs:string" minOccurs="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Identifies a customer-supplied unique identifier for this transaction. It is returned in the reply message to aid in matching requests to replies.&lt;/xs:documentation&gt;   &lt;xs:appinfo&gt;     &lt;MaxLength&gt;40&lt;/MaxLength&gt;   &lt;/xs:appinfo&gt; &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>

### element TransactionDetail/Localization

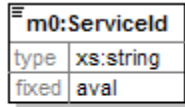
diagram	 <p>Governs any future language/translations applied to the data payload(contrasted with ClientDetail.localization, which governs Notification.localizedMessage language selection).</p>
type	<b>ns:Localization</b>
properties	isRef 0 minOcc 0 maxOcc 1
source	<pre>&lt;xs:element name="Localization" type="ns:Localization" minOccurs="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Governs any future language/translations applied to the data payload(contrasted with ClientDetail.localization, which governs Notification.localizedMessage language selection).&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>

## complexType VersionId

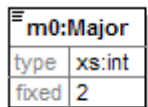
<p>diagram</p>	<pre> classDiagram     class VersionId {         +ServiceId m0:ServiceId         +Major m0:Major         +Intermediate m0:Intermediate         +Minor m0:Minor     }     class m0ServiceId {         type xs:string         fixed aval     }     class m0Major {         type xs:int         fixed 2     }     class m0Intermediate {         type xs:int         fixed 0     }     class m0Minor {         type xs:int         fixed 0     }     VersionId -- m0ServiceId     VersionId -- m0Major     VersionId -- m0Intermediate     VersionId -- m0Minor     </pre> <p><b>VersionId</b> Identifies the version/level of a service operation expected by a caller (in each request) and performed by the callee (in each reply).</p> <p><b>m0:ServiceId</b> type xs:string fixed aval Identifies a system or sub-system which performs an operation.</p> <p><b>m0:Major</b> type xs:int fixed 2 Identifies the service business level. For the initial FedEx Web Service release this value should be set to 1.</p> <p><b>m0:Intermediate</b> type xs:int fixed 0 Identifies the service interface level. For the initial FedEx Web Service release this value should be set to 0.</p> <p><b>m0:Minor</b> type xs:int fixed 0 Identifies the service code level. For the initial FedEx Web Service release this value should be set to 0.</p>
<p>children</p>	<p><a href="#">m0:ServiceId</a> <a href="#">m0:Major</a> <a href="#">m0:Intermediate</a> <a href="#">m0:Minor</a></p>
<p>source</p>	<pre> &lt;xs:complexType name="VersionId"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Identifies the version/level of a service operation expected by a caller (in each request) and performed by the callee (in each reply).&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt;   &lt;xs:sequence&gt;     &lt;xs:element name="ServiceId" type="xs:string" fixed="aval"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;Identifies a system or sub-system which performs an operation.&lt;/xs:documentation&gt;       &lt;/xs:annotation&gt;     &lt;/xs:element&gt;     &lt;xs:element name="Major" type="xs:int" fixed="2"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;Identifies the service business level. For the initial FedEx Web Service release this value should be set to 1.&lt;/xs:documentation&gt;       &lt;/xs:annotation&gt;     &lt;/xs:element&gt;     &lt;xs:element name="Intermediate" type="xs:int" fixed="0"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;Identifies the service interface level. For the initial FedEx Web Service release this value should be set to 0.&lt;/xs:documentation&gt;       &lt;/xs:annotation&gt;     &lt;/xs:element&gt;   &lt;/xs:sequence&gt; &lt;/xs:complexType&gt; </pre>

	<pre> &lt;xs:element name="Minor" type="xs:int" fixed="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Identifies the service code level. For the initial FedEx Web Service release this value should be set to 0.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt; &lt;/xs:sequence&gt; &lt;/xs:complexType&gt; </pre>
--	---

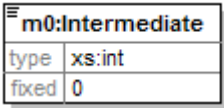
### element VersionId/ServiceId

diagram	 <p>Identifies a system or sub-system which performs an operation.</p>						
type	<b>xs:string</b>						
properties	<table border="0"> <tr><td>isRef</td><td>0</td></tr> <tr><td>content</td><td>simple</td></tr> <tr><td>fixed</td><td>aval</td></tr> </table>	isRef	0	content	simple	fixed	aval
isRef	0						
content	simple						
fixed	aval						
source	<pre> &lt;xs:element name="ServiceId" type="xs:string" fixed="aval"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Identifies a system or sub-system which performs an operation.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt; </pre>						

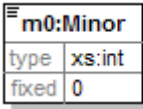
### element VersionId/Major

diagram	 <p>Identifies the service business level. For the initial FedEx Web Service release this value should be set to 1.</p>						
type	<b>xs:int</b>						
properties	<table border="0"> <tr><td>isRef</td><td>0</td></tr> <tr><td>content</td><td>simple</td></tr> <tr><td>fixed</td><td>2</td></tr> </table>	isRef	0	content	simple	fixed	2
isRef	0						
content	simple						
fixed	2						
source	<pre> &lt;xs:element name="Major" type="xs:int" fixed="2"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Identifies the service business level. For the initial FedEx Web Service release this value should be set to 1.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt; </pre>						

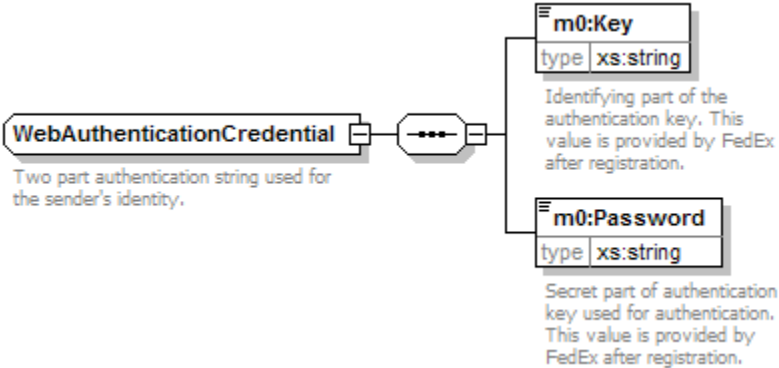
### element VersionId/Intermediate

diagram	 <p>Identifies the service interface level. For the initial FedEx Web Service release this value should be set to 0.</p>						
type	<b>xs:int</b>						
properties	<table border="0"> <tr> <td>isRef</td> <td>0</td> </tr> <tr> <td>content</td> <td>simple</td> </tr> <tr> <td>fixed</td> <td>0</td> </tr> </table>	isRef	0	content	simple	fixed	0
isRef	0						
content	simple						
fixed	0						
source	<pre>&lt;xs:element name="Intermediate" type="xs:int" fixed="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Identifies the service interface level. For the initial FedEx Web Service     release this value should be set to 0.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>						

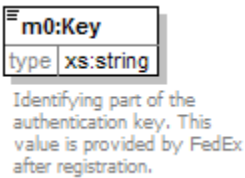
### element VersionId/Minor

diagram	 <p>Identifies the service code level. For the initial FedEx Web Service release this value should be set to 0.</p>						
type	<b>xs:int</b>						
properties	<table border="0"> <tr> <td>isRef</td> <td>0</td> </tr> <tr> <td>content</td> <td>simple</td> </tr> <tr> <td>fixed</td> <td>0</td> </tr> </table>	isRef	0	content	simple	fixed	0
isRef	0						
content	simple						
fixed	0						
source	<pre>&lt;xs:element name="Minor" type="xs:int" fixed="0"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Identifies the service code level. For the initial FedEx Web Service release     this value should be set to 0.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>						

## complexType WebAuthenticationCredential

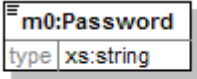
diagram	
children	<a href="#">m0:Key</a> <a href="#">m0:Password</a>
source	<pre> &lt;xs:complexType name="WebAuthenticationCredential"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Two part authentication string used for the sender's identity.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt;   &lt;xs:sequence&gt;     &lt;xs:element name="Key" type="xs:string"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;Identifying part of the authentication key. This value is provided by FedEx after registration.&lt;/xs:documentation&gt;         &lt;xs:appinfo&gt;           &lt;MaxLength&gt;16&lt;/MaxLength&gt;         &lt;/xs:appinfo&gt;       &lt;/xs:annotation&gt;     &lt;/xs:element&gt;     &lt;xs:element name="Password" type="xs:string"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;Secret part of authentication key used for authentication. This value is provided by FedEx after registration.&lt;/xs:documentation&gt;         &lt;xs:appinfo&gt;           &lt;MaxLength&gt;25&lt;/MaxLength&gt;         &lt;/xs:appinfo&gt;       &lt;/xs:annotation&gt;     &lt;/xs:element&gt;   &lt;/xs:sequence&gt; &lt;/xs:complexType&gt; </pre>

## element WebAuthenticationCredential/Key

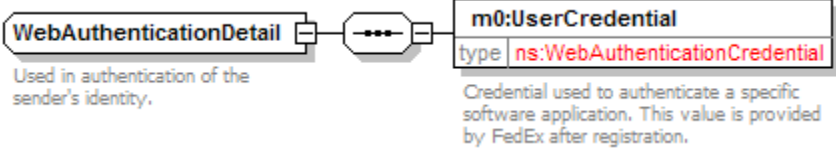
diagram	
type	<b>xs:string</b>
properties	isRef 0 content simple

source	<pre> &lt;xs:element name="Key" type="xs:string"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Identifying part of the authentication key. This value is provided by FedEx after registration.&lt;/xs:documentation&gt;     &lt;xs:appinfo&gt;       &lt;MaxLength&gt;16&lt;/MaxLength&gt;     &lt;/xs:appinfo&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt; </pre>
--------	--

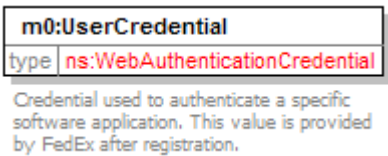
### element **WebAuthenticationCredential/Password**

diagram	 <p>Secret part of authentication key used for authentication. This value is provided by FedEx after registration.</p>
type	<b>xs:string</b>
properties	isRef 0 content simple
source	<pre> &lt;xs:element name="Password" type="xs:string"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Secret part of authentication key used for authentication. This value is provided by FedEx after registration.&lt;/xs:documentation&gt;     &lt;xs:appinfo&gt;       &lt;MaxLength&gt;25&lt;/MaxLength&gt;     &lt;/xs:appinfo&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt; </pre>

### complexType **WebAuthenticationDetail**

diagram	 <p>Used in authentication of the sender's identity.</p> <p>Credential used to authenticate a specific software application. This value is provided by FedEx after registration.</p>
children	<a href="#">m0:UserCredential</a>
source	<pre> &lt;xs:complexType name="WebAuthenticationDetail"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Used in authentication of the sender's identity.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt;   &lt;xs:sequence&gt;     &lt;xs:element name="UserCredential" type="ns:WebAuthenticationCredential"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;Credential used to authenticate a specific software application. This value is provided by FedEx after registration.&lt;/xs:documentation&gt;       &lt;/xs:annotation&gt;     &lt;/xs:element&gt;   &lt;/xs:sequence&gt; &lt;/xs:complexType&gt; </pre>

### element WebAuthenticationDetail/UserCredential

diagram	
type	ns:WebAuthenticationCredential
properties	isRef 0
source	<pre>&lt;xs:element name="UserCredential" type="ns:WebAuthenticationCredential"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Credential used to authenticate a specific software application. This value is provided by FedEx after registration.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:element&gt;</pre>

### simpleType AddressValidationAccuracyType

type	restriction of xs:string
facets	<ul style="list-style-type: none"> <li>enumeration EXACT</li> <li>enumeration TIGHT</li> <li>enumeration MEDIUM</li> <li>enumeration LOOSE</li> </ul>
source	<pre>&lt;xs:simpleType name="AddressValidationAccuracyType"&gt;   &lt;xs:restriction base="xs:string"&gt;     &lt;xs:enumeration value="EXACT"/&gt;     &lt;xs:enumeration value="TIGHT"/&gt;     &lt;xs:enumeration value="MEDIUM"/&gt;     &lt;xs:enumeration value="LOOSE"/&gt;   &lt;/xs:restriction&gt; &lt;/xs:simpleType&gt;</pre>

### simpleType AddressValidationChangeType

type	restriction of xs:string
facets	<ul style="list-style-type: none"> <li>enumeration APARTMENT_NUMBER_NOT_FOUND</li> <li>enumeration APARTMENT_NUMBER_REQUIRED</li> <li>enumeration NORMALIZED</li> <li>enumeration REMOVED_DATA</li> <li>enumeration BOX_NUMBER_REQUIRED</li> <li>enumeration NO_CHANGES</li> <li>enumeration MODIFIED_TO_ACHIEVE_MATCH</li> <li>enumeration STREET_RANGE_MATCH</li> <li>enumeration BOX_NUMBER_MATCH</li> <li>enumeration RR_OR_HC_MATCH</li> <li>enumeration CITY_MATCH</li> <li>enumeration POSTAL_CODE_MATCH</li> <li>enumeration RR_OR_HC_BOX_NUMBER_NEEDED</li> <li>enumeration FORMATTED_FOR_COUNTRY</li> <li>enumeration APO_OR_FPO_MATCH</li> <li>enumeration GENERAL_DELIVERY_MATCH</li> <li>enumeration FIELD_TRUNCATED</li> <li>enumeration UNABLE_TO_APPEND_NON_ADDRESS_DATA</li> <li>enumeration INSUFFICIENT_DATA</li> </ul>

	<pre> enumeration HOUSE_OR_BOX_NUMBER_NOT_FOUND enumeration POSTAL_CODE_NOT_FOUND enumeration INVALID_COUNTRY enumeration SERVICE_UNAVAILABLE_FOR_ADDRESS </pre>
source	<pre> &lt;xs:simpleType name="AddressValidationChangeType"&gt;   &lt;xs:restriction base="xs:string"&gt;     &lt;xs:enumeration value="APARTMENT_NUMBER_NOT_FOUND"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;           (EAS 100) Apartment number not           found for this address.         &lt;/xs:documentation&gt;       &lt;/xs:annotation&gt;     &lt;/xs:enumeration&gt;     &lt;xs:enumeration value="APARTMENT_NUMBER_REQUIRED"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;           (EAS 101) Address requires           apartment number.         &lt;/xs:documentation&gt;       &lt;/xs:annotation&gt;     &lt;/xs:enumeration&gt;     &lt;xs:enumeration value="NORMALIZED"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;           (EAS 102) Address normalized -           abbreviations applied.         &lt;/xs:documentation&gt;       &lt;/xs:annotation&gt;     &lt;/xs:enumeration&gt;     &lt;xs:enumeration value="REMOVED_DATA"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;           (EAS 103) Dropped data.         &lt;/xs:documentation&gt;       &lt;/xs:annotation&gt;     &lt;/xs:enumeration&gt;     &lt;xs:enumeration value="BOX_NUMBER_REQUIRED"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;           (EAS 104) Address requires box           number.         &lt;/xs:documentation&gt;       &lt;/xs:annotation&gt;     &lt;/xs:enumeration&gt;     &lt;xs:enumeration value="NO_CHANGES"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt;           (EAS 200) Match - no changes           applied to input address.         &lt;/xs:documentation&gt;       &lt;/xs:annotation&gt;     &lt;/xs:enumeration&gt;     &lt;xs:enumeration value="MODIFIED_TO_ACHIEVE_MATCH"&gt;       &lt;xs:annotation&gt;         &lt;xs:documentation&gt; </pre>

	<p>achieve match. (EAS 201) Address modified to</p> <pre> &lt;/xs:documentation&gt; &lt;/xs:annotation&gt; &lt;/xs:enumeration&gt; &lt;xs:enumeration value="STREET_RANGE_MATCH"&gt; &lt;xs:annotation&gt; &lt;xs:documentation&gt; </pre> <p>(EAS 202) Match to street range.</p> <pre> &lt;/xs:documentation&gt; &lt;/xs:annotation&gt; &lt;/xs:enumeration&gt; &lt;xs:enumeration value="BOX_NUMBER_MATCH"&gt; &lt;xs:annotation&gt; &lt;xs:documentation&gt; </pre> <p>(EAS 203) Match to box number.</p> <pre> &lt;/xs:documentation&gt; &lt;/xs:annotation&gt; &lt;/xs:enumeration&gt; &lt;xs:enumeration value="RR_OR_HC_MATCH"&gt; &lt;xs:annotation&gt; &lt;xs:documentation&gt; </pre> <p>(EAS 204) Match to Rural Route (RR) / Highway Contract (HC) address.</p> <pre> &lt;/xs:documentation&gt; &lt;/xs:annotation&gt; &lt;/xs:enumeration&gt; &lt;xs:enumeration value="CITY_MATCH"&gt; &lt;xs:annotation&gt; &lt;xs:documentation&gt; </pre> <p>(EAS 205) Match to city (non-US, non-Canada).</p> <pre> &lt;/xs:documentation&gt; &lt;/xs:annotation&gt; &lt;/xs:enumeration&gt; &lt;xs:enumeration value="POSTAL_CODE_MATCH"&gt; &lt;xs:annotation&gt; &lt;xs:documentation&gt; </pre> <p>(EAS 206) Match to postal code only (non-street)</p> <pre> &lt;/xs:documentation&gt; &lt;/xs:annotation&gt; &lt;/xs:enumeration&gt; &lt;xs:enumeration value="RR_OR_HC_BOX_NUMBER_NEEDED"&gt; &lt;xs:annotation&gt; &lt;xs:documentation&gt; </pre> <p>(EAS 207) Need box number for Rural Route / Highway Contract (HC) match.</p> <pre> &lt;/xs:documentation&gt; &lt;/xs:annotation&gt; &lt;/xs:enumeration&gt; &lt;xs:enumeration value="FORMATTED_FOR_COUNTRY"&gt; &lt;xs:annotation&gt; &lt;xs:documentation&gt; </pre> <p>(EAS 208) Formatting performed for country (non-US, non-Canada).</p>
--	---

	<pre> &lt;/xs:documentation&gt; &lt;/xs:annotation&gt; &lt;/xs:enumeration&gt; &lt;xs:enumeration value="APO_OR_FPO_MATCH"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;       (EAS 209) Match to military address (e.g. APO/FPO).     &lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:enumeration&gt; &lt;xs:enumeration value="GENERAL_DELIVERY_MATCH"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;       (EAS 210) Match to general delivery.     &lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:enumeration&gt; &lt;xs:enumeration value="FIELD_TRUNCATED"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;       (EAS 211) Address exceeded 35 character plug-in limit.     &lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:enumeration&gt; &lt;xs:enumeration value="UNABLE_TO_APPEND_NON_ADDRESS_DATA"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;       (EAS 212) Unable to append non- address; data 35 character limit imposed.     &lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:enumeration&gt; &lt;xs:enumeration value="INSUFFICIENT_DATA"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;       (EAS 300) Insufficient data for address verification.     &lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:enumeration&gt; &lt;xs:enumeration value="HOUSE_OR_BOX_NUMBER_NOT_FOUND"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;       (EAS 301) Address (house or box number) not found.     &lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:enumeration&gt; &lt;xs:enumeration value="POSTAL_CODE_NOT_FOUND"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;       (EAS 303) Postal code not found.     &lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:enumeration&gt; &lt;/xs:documentation&gt; </pre>
--	--

	<pre> &lt;/xs:annotation&gt; &lt;/xs:enumeration&gt; &lt;xs:enumeration value="INVALID_COUNTRY"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;       (EAS 305) Invalid country.     &lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:enumeration&gt; &lt;xs:enumeration value="SERVICE_UNAVAILABLE_FOR_ADDRESS"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;       (EAS 400) Service unavailable for       request address.     &lt;/xs:documentation&gt;   &lt;/xs:annotation&gt; &lt;/xs:enumeration&gt; &lt;/xs:restriction&gt; &lt;/xs:simpleType&gt; </pre>
--	--

#### simpleType **DeliveryPointValidationType**

type	restriction of <b>xs:string</b>
facets	<pre> enumeration CONFIRMED enumeration UNCONFIRMED enumeration UNAVAILABLE </pre>
source	<pre> &lt;xs:simpleType name="DeliveryPointValidationType"&gt;   &lt;xs:restriction base="xs:string"&gt;     &lt;xs:enumeration value="CONFIRMED"/&gt;     &lt;xs:enumeration value="UNCONFIRMED"/&gt;     &lt;xs:enumeration value="UNAVAILABLE"/&gt;   &lt;/xs:restriction&gt; &lt;/xs:simpleType&gt; </pre>

#### simpleType **NotificationSeverityType**

type	restriction of <b>xs:string</b>
facets	<pre> enumeration ERROR enumeration FAILURE enumeration NOTE enumeration SUCCESS enumeration WARNING </pre>
source	<pre> &lt;xs:simpleType name="NotificationSeverityType"&gt;   &lt;xs:annotation&gt;     &lt;xs:documentation&gt;Identifies the set of severity values for a Notification.&lt;/xs:documentation&gt;   &lt;/xs:annotation&gt;   &lt;xs:restriction base="xs:string"&gt;     &lt;xs:enumeration value="ERROR"/&gt;     &lt;xs:enumeration value="FAILURE"/&gt;     &lt;xs:enumeration value="NOTE"/&gt;     &lt;xs:enumeration value="SUCCESS"/&gt;     &lt;xs:enumeration value="WARNING"/&gt;   &lt;/xs:restriction&gt; &lt;/xs:simpleType&gt; </pre>

### simpleType ResidentialStatusType

type	restriction of <b>xs:string</b>
facets	enumeration UNDETERMINED enumeration BUSINESS enumeration RESIDENTIAL enumeration INSUFFICIENT_DATA enumeration UNAVAILABLE enumeration NOT_APPLICABLE_TO_COUNTRY
source	<pre>&lt;xs:simpleType name="ResidentialStatusType"&gt; &lt;xs:restriction base="xs:string"&gt;   &lt;xs:enumeration value="UNDETERMINED"/&gt;   &lt;xs:enumeration value="BUSINESS"/&gt;   &lt;xs:enumeration value="RESIDENTIAL"/&gt;   &lt;xs:enumeration value="INSUFFICIENT_DATA"/&gt;   &lt;xs:enumeration value="UNAVAILABLE"/&gt;   &lt;xs:enumeration value="NOT_APPLICABLE_TO_COUNTRY"/&gt; &lt;/xs:restriction&gt; &lt;/xs:simpleType&gt;</pre>