FedEx Web Services

2013

Address Validation Service

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 guarantee. FedEx may suspend any applicable money-back guarantee in the event of equipment failure or if it becomes inoperative.

Confidential and Proprietary

The information contained in this guide is confidential and proprietary to FedEx Corporate Services, Inc. and its affiliates (collectively "FedEx"). No part of this guide may be distributed or disclosed in any form to any third party without written permission of FedEx. This guide is provided to you 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 guide, the FedEx Automation Agreement and the *FedEx Service Guide* shall be governed by the FedEx Automation Agreement and the *FedEx Service Guide*, in that order.

Disclaimer

All Improper Transaction scenarios are for example only. They do not reflect all error condition scenarios.

Contents

Tables

About This Guide

Document Organization

Resources

Support

This guide describes how to integrate with 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 each available service in addition to the business logic that drives each FedEx 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

- FedEx Services At-a-Glance: **fedex.com/us/services**
- *FedEx Service Guide* available at **fedex.com/us/service-guide**
- Microsoft Web Services: msdn.microsoft.com/en-us/library/ms950421.aspx
- O'Reilly XML.com: www.xml.com
- Secure Socket Layer Certificates: **fedex.com/us/developer/downloads/dev_cert.zip**
- Web Services organization home page: www.web-services.org

Support

- Contact FedEx Web Services technical support at **websupport@fedex.com**.
- For technical support, call 1.877.339.2774 and state "API" at the voice prompt.

  Support hours are Monday through Friday, 7:00 a.m. to 9:00 p.m. CST, and Saturday, 9:00 a.m. to 3:00 p.m. CST.
- For FedEx Customer Service, call **1.800.GoFedEx 1.800.463.3339**.

  Customers using a FedEx® Compatible Solutions Program automation solution should contact their software provider for support.

1 Introduction

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 the Address Validation WSDL. See Chapter 2: Address Validation Service for more information.
- **Give Customers More Options:** Help customers learn about all the available shipping options and rates with Ship Service WSDL, OpenShip WSDL, and Rate Services WSDL. You can also extend this service to your shopping cart and website, allowing customers to access money-saving information firsthand.
- **More Convenience:** Use the GlobalShipAddress Service WSDL to find the FedEx pickup location nearest your customer. Or, send an email 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 WSDL.
- **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:** In addition to ExpressTagAvailability, provide a simple way to allow customers to return an order with Email Labels. This service sends an email with the address (URL) of a website 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 Simple Object Access Protocol (SOAP) and Web Services Description Language (WSDL) standards for message transfer and service discovery. By following the standards, applications can seamlessly communicate with platform services.

Document Overview

This guide provides instructions for coding the functions you need to develop FedEx supported applications. The following chapters make up this guide:

- Introduction (this chapter):
  - Documentation overview and guidelines, including how to use the Help application and how to print this guide.
  - Overview information about web services, including a high-level description of FedEx Web Services methods.
  - Coding basics.

– Overview information about testing and certifying your application.

Each chapter covering FedEx Web Services coding includes:

- 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.

Printing All or Part of This Guide

You can print all or part of this guide from the PDF version.

Printing from the PDF Version

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

1. Open the PDF file and click the printer icon or click **File** > **Print**.
2. From the **Print** dialog box, print the complete document, specify a page range, or choose from any of the available print options.

Web Services, WSDL, and SOAP Overview

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

Web Services

Web services are 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.

Web Services are consumed by many different applications across many platforms. It is based on the basic principles that govern XML standards, one of which is how Namespaces can be declared and applied.

Namespaces are declared as an attribute of an element. It is not mandatory to declare namespaces only at the root element; rather it could be declared at any element in the XML document. The scope of a declared namespace begins at the element where it is declared and applies to the entire content of that element, unless overridden by another namespace declaration with the same prefix name, the content of an element is the content between the <opening-tag> and </closing-tag> of that element. So essentially, XML namespace declarations are scoped, meaning that the declared prefix (or default namespace) is in force for the element on which the declaration occurs (as well as its descendant elements). A namespace declared as follows:

<v12:RateReply xmlns:v12="http://

is semantically same as

<RateReply xmlns="http://fedex.com/ws/rate/v12">

or even (hypothetically) same as

<foo:RateReply xmlns:foo="http://fedex.com/ws/rate/v12">

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>.

WSDL Elements

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 only three messages: **Message - Definition** Input Message - Data web services receive Output Message - Data web services send Fault Message - Error messages from web services
<service>	Contains a <port> child element that describes the URL where the service is located. This is the location of the ultimate web service.
<binding>	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 World Wide Web Consortium (W3C) Website at w3.org/TR/wsdl.

SOAP

- Is a simple XML-based protocol that allows applications to exchange information over HTTP.
- Is built on open standards supported by numerous development tools on various platforms.
- Is a request interface object in your application programming language.
- Provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.
- Enables the data to pass through layers of intermediaries and arrive at the ultimate receiver the way it was intended.

Note: 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 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/v13">
<SOAP-ENV:Body>
<DeleteTagRequest>
<WebAuthenticationDetail>
    <UserCredential>
      <Key>
       User Key
      </Key>
    <Password>
     User Password
    </Password>
  </UserCredential>
</WebAuthenticationDetail>
<Client detail>
   <AccountNumber>xxxxxxxxx</Account number>
   <MeterNumber>xxxxxx</MeterNumber>
</ClientDetail>

<Version>
     <ServiceId>ship</ServiceId>
     <Major>12</Major>
     <Intermediate>0</Intermediate>
     <Minor>0</Minor>
</Version>
<DispatchLocationId>MQYA</DispatchLocationId>
<DispatchDate>2012-06-01</DispatchDate>
<Payment>
     <PaymentType>shipper</PaymentType>
     <Payor>
         <AccountNumber>xxxxxxxxx</AccountNumber>
         <CountryCode>US</CountryCode>
     </Payor>
</Payment>
<ConfirmationNumber>997037200019454</ConfirmationNumber>
</DeleteTagRequest>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Non-SOAP Web Services

FedEx offers a non-SOAP web services solution that you can use to send transactions 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 communications method.

To use the non-SOAP web service solution, you must have a working knowledge of HTTPS and Secure Socket Layering (SSL) encryption, the ability to provide a secure SSL connection to FedEx and the ability to code to an operation 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. The following is an example of a TrackRequest using the non-SOAP interface.

Example Track Request

```
 <q0:TrackRequest>
        <q0:WebAuthenticationDetail>
           <q0:UserCredential>
              <q0:Key>xxxxxxxxxxxxxxxx</q0:Key>
              <q0:Password/>
           </q0:UserCredential>
        </q0:WebAuthenticationDetail>
        <q0:ClientDetail>
           <q0:AccountNumber>xxxxxxxxx</q0:AccountNumber>
           <q0:MeterNumber>xxxxxxxx</q0:MeterNumber>
           <q0:IntegratorId/>
           <q0:Localization>
              <q0:LanguageCode>EN</q0:LanguageCode>
              <q0:LocaleCode>us</q0:LocaleCode>
           </q0:Localization>
        </q0:ClientDetail>
        <q0:TransactionDetail>

<q0:CustomerTransactionId>Basic_TrackRequest_q0_Internal</q0:Cus
tomerTransactionId>
           <q0:Localization>
              <q0:LanguageCode>EN</q0:LanguageCode>
              <q0:LocaleCode>us</q0:LocaleCode>
           </q0:Localization>
        </q0:TransactionDetail>
        <q0:Version>
           <q0:ServiceId>trck</q0:ServiceId>
           <q0:Major>7</q0:Major>
           <q0:Intermediate>0</q0:Intermediate>
           <q0:Minor>0</q0:Minor>
        </q0:Version>
        <q0:SelectionDetails>
           <q0:CarrierCode>FDXE</q0:CarrierCode>
           <q0:PackageIdentifier>
              <q0:Type>TRACKING_NUMBER_OR_DOORTAG</q0:Type>
              <q0:Value>797843158299</q0:Value>
           </q0:PackageIdentifier>
        </q0:SelectionDetails>

<q0:ProcessingOptions>INCLUDE_DETAILED_SCANS</q0:ProcessingOptio
ns>
      </q0:TrackRequest>
```

Error Handling

Error handling for non-SOAP operations 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/v8' 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/8">
<ship:AccountNumberx>000000000000000000</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: 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, and host systems:

```
POST /xml HTTP/1.0
Referrer: YourCompanyNameGoesHere
Host: wsbeta.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, you will need to open an SSL connection to the FedEx test server and send the request through FedEx by using 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 an 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 implementing FedEx Web Services, note the following guidelines:

- FedEx Web Services are designed for use by skilled developers who are familiar with the communication standards SOAP and 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 graphical user interface (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.
- FedEx Web Services are designed to support any operating system and coding language. Downloadable sample code is available in Java, C#, VB, .Net and PHP languages from the FedEx Developer Resource Center Technical Resources.

- Transactions submitted to FedEx using FedEx Web Services are required to have a minimum of 128-bit encryption to complete the request.

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 you send it to FedEx. FedEx replies with a response that contains the data resulting from the instructions you sent in.

Note: The 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, such as 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.

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 conform to industry standards and are compatible with a comprehensive array of developers' 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.

To obtain FedEx Web Services and begin integrating with an application, you need to access documentation, sample code, and sample service requests and replies with the WSDLs from the FedEx Developer Resource Center Technical Resources. Also, obtain a test meter number to engage in real-time online testing in the FedEx hosted test environment.

Note: Not all services are available outside the U.S.

Testing

FedEx supplies a complete online operating environment with which to test your applications against live FedEx servers. 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.

Production credentials can be obtained prior to the certification process. Advanced services are not enabled, but standard services are enabled. Refer to Preproduction Assistance for more information on support from FedEx.

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 thru Friday, 7 a.m. to 9 p.m. and Saturday 9 a.m. to 3 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 Solutions 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 CSP providers, go to the Available CSP Solutions page at http://www.fedex.com/us/compatible-solutions/customer/.

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.

Once an application has completed the above mentioned process and requirements, FedEx will enable the provider's CSP credentials for processing all applicable services in the production environment. The URL needed to direct the CSP application to the FedEx production servers will also be provided. The provider would then need to obtain production User Credentials (Register CSP User Service) and a production meter number (Subscribe Service). Once this information has been obtained with the connection directed to the production servers, the provider's application is considered 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 if 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.

Requirements and Resources for Corporate Developers	
Must be accepted into the FedEx® Compatible Solutions Program (CSP)	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

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 (EULA) to FedEx to obtain a production test meter.

Requirements and Resources for Commercial Developers

Commercial developers create solutions with the intent of distributing and/or 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.

If you are a commercial developer interested in becoming a FedEx Compatible Solutions Program provider, go to **http://www.fedex.com/us/compatible-solutions/customer/** for more information about the FedEx Compatible Solutions Program (CSP).

URL Errors

If a VB.NET or C# project still sends transactions to the test server after changing the URL in the WSDLs to print 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 FedEx 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.

2 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 Request

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.

Note: The Address Validation Service is an advanced service and must be enabled by FedEx Customer Support for production use. Contact your FedEx account executive for more information.

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.

Note: Use the information returned from an AddressValidationRequest as for suggested use only, rather than a reference.

Address Validation Service Details

The followings 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.

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.
- 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.

Address Results

You should confirm an address for accuracy before using it to ship a package. Occasionally EAS will return multiple addresses. A case in point is when the directional indicator is missing (e.g. North or South). To narrow your results, you can provide more specific address information and check the address again. To confirm the address accuracy, you can provide more specific address data 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

Table 1. Address Validation Request Elements

Element	Required	Description
AddressValidationRequest/ RequestTimestamp	Required	Time of request based on shipper’s time zone. Defaults to CDT.

Table 1. Address Validation Request Elements, continued

Element	Required	Description
		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: • Company • City • StateorProvinceCode • PostalCode • UrbanizationCode • CountryCode • CountryName • Residential *Note: Up to 100 addresses can be validated in one request.*
AddressValidationRequest/ AddressValidationOptions	Optional	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: VerifyAddresses to validate all address elements and return in the reply. CheckResidentialStatus check addresses for residential status only. MaximumNumberOfMatches allows you to configure the number of possible matches returned. Maximum is 10. StreetAccuracy: Values are: EXACT, TIGHT, MEDIUM, and LOOSE. DirectionalAccuracy: Values are EXACT, TIGHT, MEDIUM and LOOSE. CompanyNameAccuracy: Values are EXACT, TIGHT, MEDIUM and LOOSE. 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: • EXACT: input must match the database exactly. • TIGHT: matching of address is allowed for slight variance • MEDIUM: matching of address allows for more variance of address and provides corrections [default] • LOOSE: matching of address is minimal **Warning:** 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. ConvertToUpperCase element controls whether addresses are returned in upper case text. 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. ReturnParsedElements returns the address validation elements in the reply, as verified by the system before validation.

AddressValidationReply Elements

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

Table 2. 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	Returned values are: • APARTMENT_NUMBER_NOT_FOUND • APARTMENT_NUMBER_REQUIRED • NORMALIZED • REMOVED_DATA • BOX_NUMBER_REQUIRED • NO_CHANGES • MODIFIED_TO_ACHIEVE_MATCH • STREET_RANGE_MATCH • BOX_NUMBER_MATCH** • RR_OR_HC_MATCH • CITY_MATCH • POSTAL_CODE_MATCH • RR_OR_HC_BOX_NUMBER_NEEDED • FORMATTED_FOR_COUNTRY • APO_OR_FPO_MATCH • GENERAL_DELIVERY_MATCH • FIELD_TRUNCATED • UNABLE_TO_APPEND_NON_ADDRESS_DATA • INSUFFICIENT_DATA • HOUSE_OR_BOX_NUMBER_NOT_FOUND • POSTAL_CODE_NOT_FOUND • INVALID_COUNTRY • SERVICE_UNAVAILABLE_FOR_ADDRESS **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. *See* Address Validation Coding Details for more information.
ProposedAddressDetails/ResidentialStatus	Returned values are: • UNDETERMINED • BUSINESS • RESIDENTIAL • INSUFFICIENT_DATA • UNAVAILABLE • NOT_APPLICABLE_TO_COUNTRY
ProposedAddressDetails/ DeliveryPointValidation	Returned values are: • CONFIRMED • UNCONFIRMED • UNAVAILABLE
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.

Table 2. Address Validation Reply Elements, continued

Element	Description
ProposedAddressDetails/RemovedNon-AddressData	Any information removed from the submitted address before validation.

Notification

If the Enterprise Address Service (EAS) returns a nonzero systemStatus, then the AddressValidationReply will contain a severity notification of FAILURE and a code equal to that systemStatus.

Mapping Enterprise Address Service Changes

Table 3. Enterprise Address Service Indicator Changes Elements

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

Table 3. Enterprise Address Service Indicator Changes Elements, continued

EAS addressIndicator (indicator attribute)	Changes Element
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

Mapping Enterprise Address Service Residential Status

Table 4. Enterprise Address Service Residential Status Elements

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

Table 4. Enterprise Address Service Residential Status Elements, continued

EAS businessResidentialIndicator	Residential Status Element
6	NOT_APPLICABLE_TO_COUNTRY

Mapping Enterprise Address Service DeliveryPointValidation

Table 5. Enterprise Address Service DeliveryPointValidation Elements

EAS dpvIndicator	DeliveryPointValidation Element
1	CONFIRMED
2	UNCONFIRMED
3	UNAVAILABLE

Mapping Enterprise Address Service Changes

Table 6. Enterprise Address Service Indicator Changes Elements

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

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.

To solve this issue:

- 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 ParsedAddres
        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:

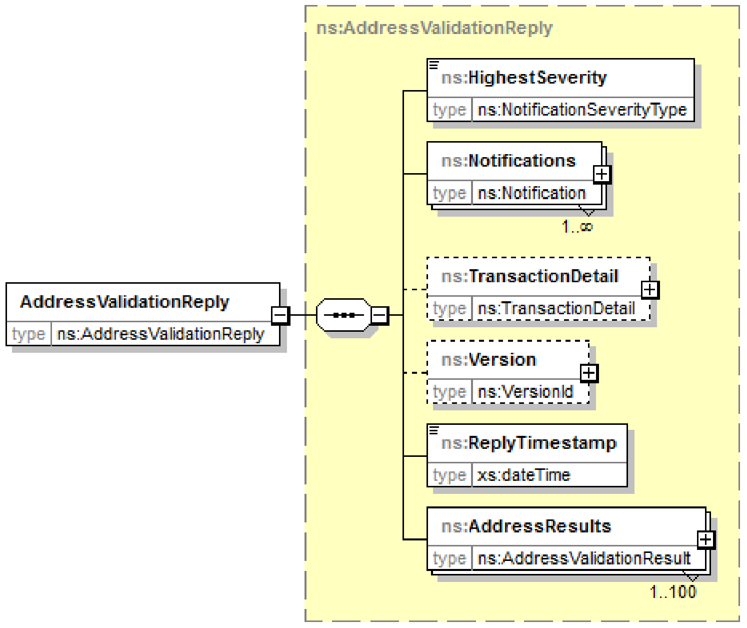
```
/*
     public com.fedex.addressvalidation.stub.ParsedElement[] getParsedStreetLine(int i) {
         return this.parsedStreetLine[i];
     }
     public void setParsedStreetLine(
         int i, com.fedex.addressvalidation.stub.ParsedElement[] _value) {
             this.parsedStreetLine[i] = _value;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

Schema AddressValidationService_v2.xsd

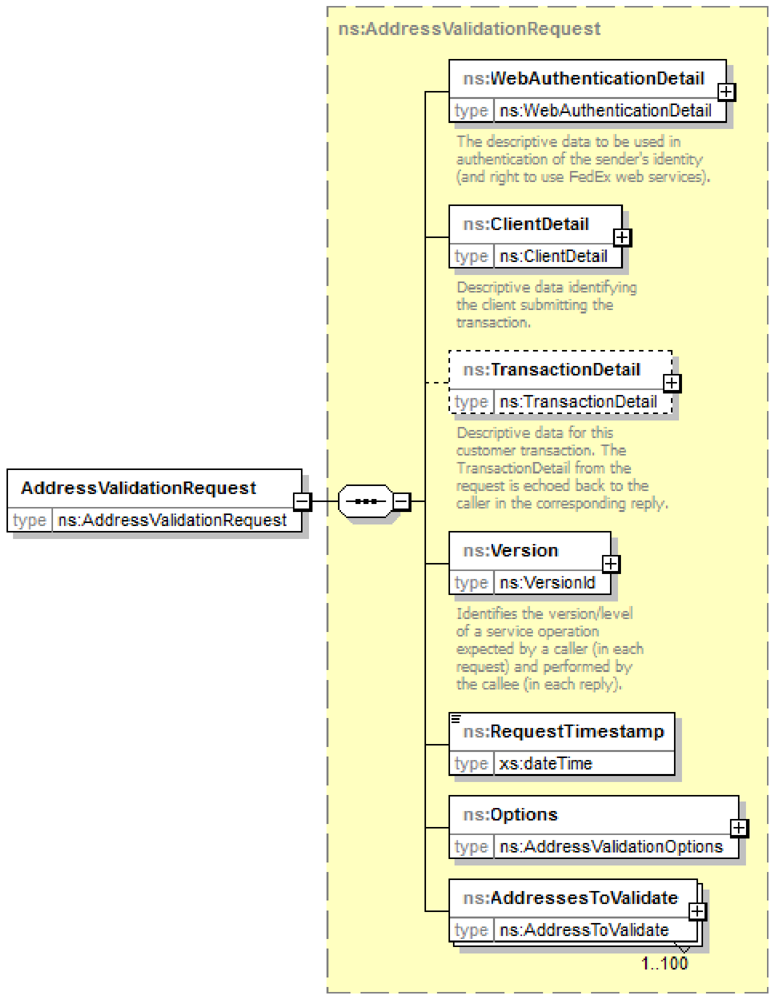
targetNamespace: **http://fedex.com/ws/addressvalidation/v2**

Elements	Complex types	Simple types
AddressValidationReply	Address	AddressValidationAccuracyType
AddressValidationRequest	AddressToValidate	AddressValidationChangeType
	AddressValidationOptions	DeliveryPointValidationType
	AddressValidationReply	NotificationSeverityType
	AddressValidationRequest	ResidentialStatusType
	AddressValidationResult	
	ClientDetail	
	Localization	
	Notification	
	NotificationParameter	
	ParsedAddress	
	ParsedAddressPart	
	ParsedElement	
	ProposedAddressDetail	
	TransactionDetail	
	VersionId	
	WebAuthenticationCredential	
	WebAuthenticationDetail	

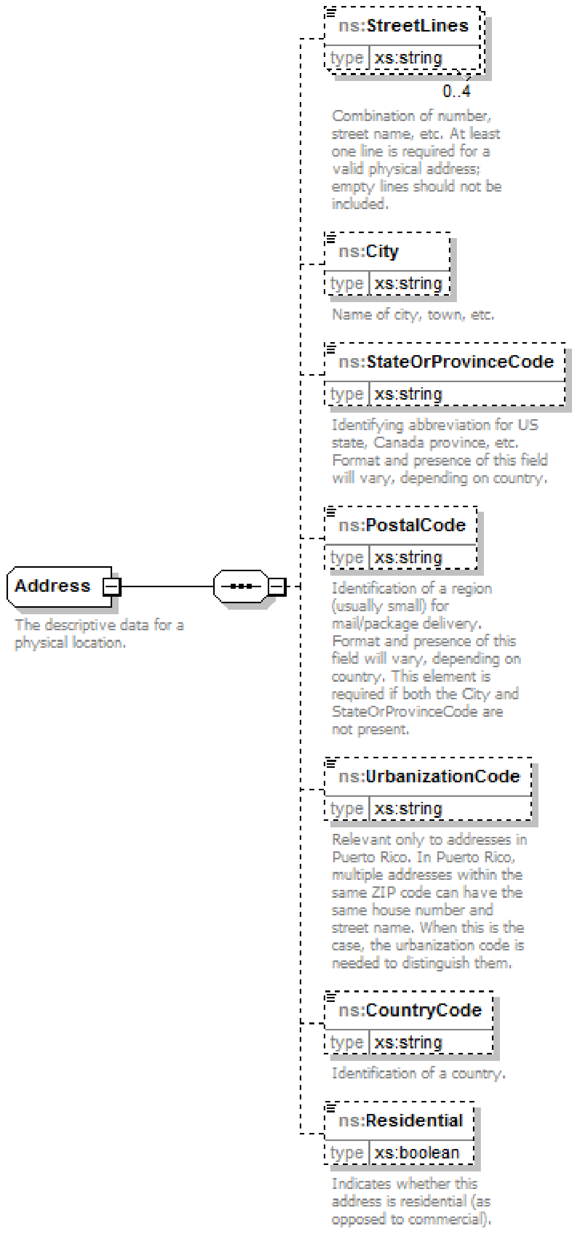
element **AddressValidationReply**

diagram	 ns:AddressValidationReply AddressValidationReply type ns:AddressValidationReply ns:HighestSeverity type ns:NotificationSeverityType ns:Notifications type ns:Notification 1..∞ ns:TransactionDetail type ns:TransactionDetail ns:Version type ns:VersionId ns:ReplyTimestamp type xs:dateTime ns:AddressResults type ns:AddressValidationResult 1..100
namespace	http://fedex.com/ws/addressvalidation/v2
type	**ns:AddressValidationReply**
properties	content complex
children	**ns:HighestSeverity ns:Notifications ns:TransactionDetail ns:Version ns:ReplyTimestamp ns:AddressResults**
source	`<xs:element name="AddressValidationReply" type="ns:AddressValidationReply"/>`

element **AddressValidationRequest**

diagram	 ns:AddressValidationRequest AddressValidationRequest type ns:AddressValidationRequest ns:WebAuthenticationDetail type ns:WebAuthenticationDetail The descriptive data to be used in authentication of the sender's identity (and right to use FedEx web services). ns:ClientDetail type ns:ClientDetail Descriptive data identifying the client submitting the transaction. ns:TransactionDetail type ns:TransactionDetail Descriptive data for this customer transaction. The TransactionDetail from the request is echoed back to the caller in the corresponding reply. ns:Version 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). ns:RequestTimestamp type xs:dateTime ns:Options type ns:AddressValidationOptions ns:AddressesToValidate type ns:AddressToValidate 1..100
namespace	http://fedex.com/ws/addressvalidation/v2
type	**ns:AddressValidationRequest**
properties	content complex
children	**ns:WebAuthenticationDetail ns:ClientDetail ns:TransactionDetail ns:Version ns:RequestTimestamp ns:Options ns:AddressesToValidate**
source	<xs:element name="AddressValidationRequest" type="ns:AddressValidationRequest"/>

complexType **Address**

diagram	

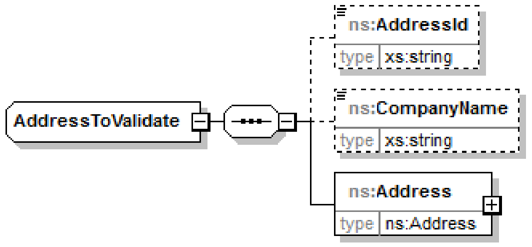
Address
The descriptive data for a physical location.
ns:StreetLines
type xs:string
0..4
Combination of number, street name, etc. At least one line is required for a valid physical address; empty lines should not be included.
ns:City
type xs:string
Name of city, town, etc.
ns:StateOrProvinceCode
type xs:string
Identifying abbreviation for US state, Canada province, etc. Format and presence of this field will vary, depending on country.
ns:PostalCode
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.
ns:UrbanizationCode
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.
ns:CountryCode
type xs:string
Identification of a country.
ns:Residential
type xs:boolean
Indicates whether this address is residential (as opposed to commercial).

namespace	http://fedex.com/ws/addressvalidation/v2
children	**ns:StreetLines ns:City ns:StateOrProvinceCode ns:PostalCode ns:UrbanizationCode ns:CountryCode ns:Residential**
used by	elements **AddressToValidate/Address ProposedAddressDetail/Address**
annotation	documentation The descriptive data for a physical location.
source	see below

```
<xs:complexType name="Address">
  <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>
          <xs:MaxLength>35</xs: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>
          <xs:MaxLength>
            <ns:Express>35</ns:Express>
            <ns:Ground>20</ns:Ground>
          </xs: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>
          <xs:MaxLength>14</xs:MaxLength>
        </xs:appinfo>
      </xs:annotation>
    </xs:element>
    <xs:element name="PostalCode" type="xs:string" minOccurs="0">
      <xs:annotation>
```

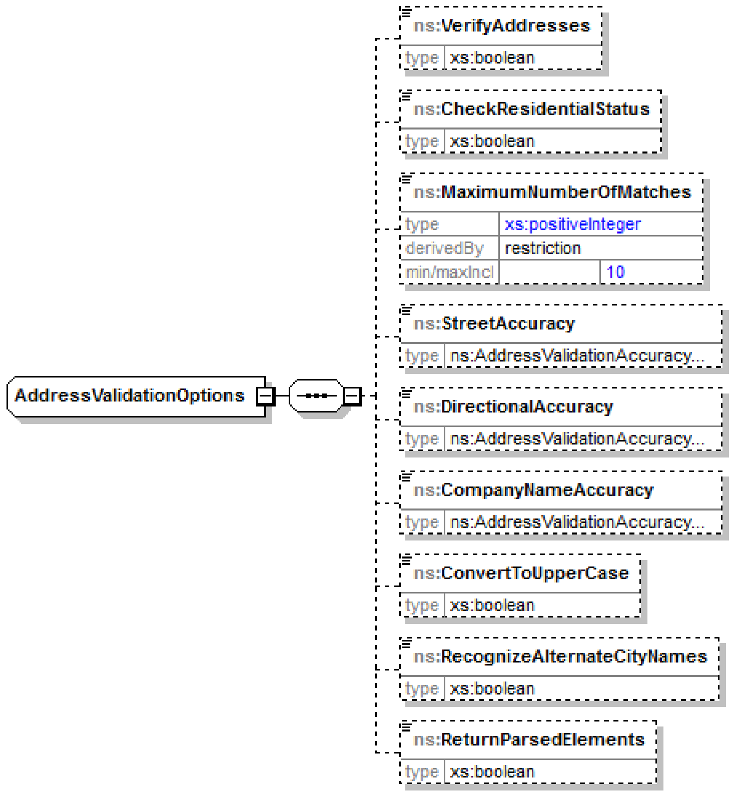
```
        <xs:documentation>Identification of a region (usually small) for mail/package delivery. Format and presence of this field will
vary, depending on country. This element is required if both the City and StateOrProvinceCode are not
present.</xs:documentation>
        <xs:appinfo>
          <xs:MaxLength>16</xs: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>
          <xs:MaxLength>100</xs: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>
        <xs:appinfo>
          <xs:MaxLength>2</xs:MaxLength>
        </xs:appinfo>
      </xs:annotation>
    </xs:element>
    <xs:element name="Residential" type="xs:boolean" minOccurs="0">
      <xs:annotation>
        <xs:documentation>Indicates whether this address is residential (as opposed to commercial).</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

complexType **AddressToValidate**

diagram	 AddressToValidate ns:AddressId type xs:string ns:CompanyName type xs:string ns:Address type ns:Address
namespace	http://fedex.com/ws/addressvalidation/v2
children	**ns:AddressId ns:CompanyName ns:Address**
used by	element **AddressValidationRequest/AddressesToValidate**

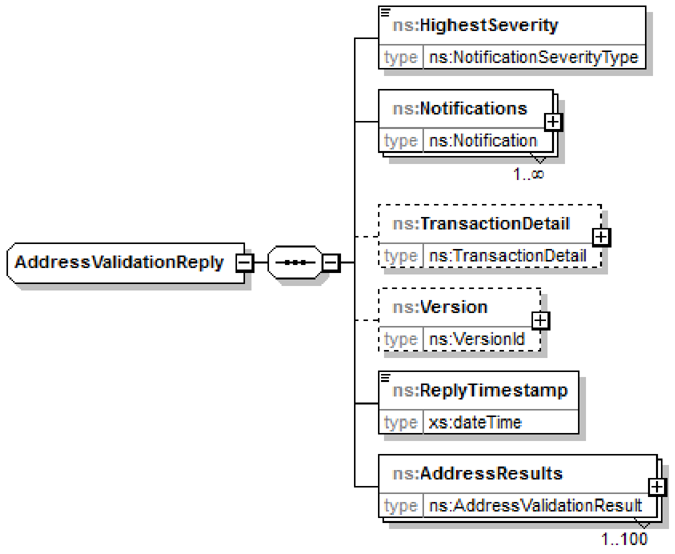
source	`<xs:complexType name="AddressToValidate">`
	`  <xs:sequence>`
	`    <xs:element name="AddressId" type="xs:string" minOccurs="0"/>`
	`    <xs:element name="CompanyName" type="xs:string" minOccurs="0"/>`
	`    <xs:element name="Address" type="ns:Address"/>`
	`  </xs:sequence>`
	`</xs:complexType>`

complexType **AddressValidationOptions**

diagram	 AddressValidationOptions ns:VerifyAddresses type xs:boolean ns:CheckResidentialStatus type xs:boolean ns:MaximumNumberOfMatches type xs:positiveInteger derivedBy restriction min/maxIncl 10 ns:StreetAccuracy type ns:AddressValidationAccuracy... ns:DirectionalAccuracy type ns:AddressValidationAccuracy... ns:CompanyNameAccuracy type ns:AddressValidationAccuracy... ns:ConvertToUpperCase type xs:boolean ns:RecognizeAlternateCityNames type xs:boolean ns:ReturnParsedElements type xs:boolean
namespace	http://fedex.com/ws/addressvalidation/v2
children	**ns:VerifyAddresses ns:CheckResidentialStatus ns:MaximumNumberOfMatches ns:StreetAccuracy ns:DirectionalAccuracy ns:CompanyNameAccuracy ns:ConvertToUpperCase ns:RecognizeAlternateCityNames ns:ReturnParsedElements**
used by	element **AddressValidationRequest/Options**
source	see below

```
<xs:complexType name="AddressValidationOptions">
  <xs:sequence>
    <xs:element name="VerifyAddresses" type="xs:boolean" minOccurs="0"/>
    <xs:element name="CheckResidentialStatus" type="xs:boolean" minOccurs="0"/>
    <xs:element name="MaximumNumberOfMatches" minOccurs="0">
      <xs:simpleType>
        <xs:restriction base="xs:positiveInteger">
          <xs:maxInclusive value="10"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="StreetAccuracy" type="ns:AddressValidationAccuracyType" minOccurs="0"/>
    <xs:element name="DirectionalAccuracy" type="ns:AddressValidationAccuracyType" minOccurs="0"/>
    <xs:element name="CompanyNameAccuracy" type="ns:AddressValidationAccuracyType" minOccurs="0"/>
    <xs:element name="ConvertToUpperCase" type="xs:boolean" minOccurs="0"/>
    <xs:element name="RecognizeAlternateCityNames" type="xs:boolean" minOccurs="0"/>
    <xs:element name="ReturnParsedElements" type="xs:boolean" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

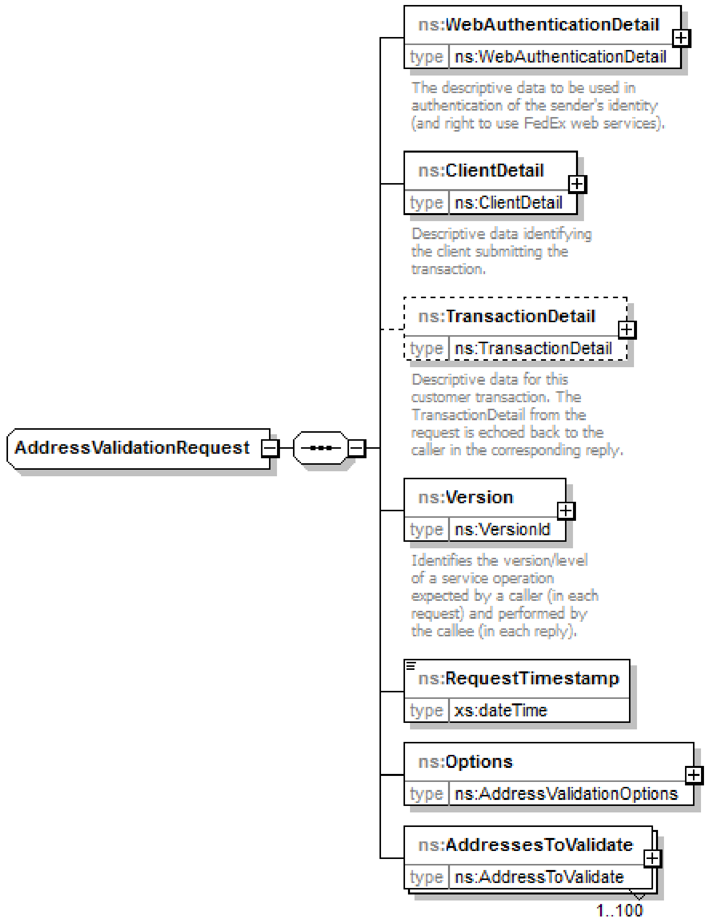
complexType **AddressValidationReply**

diagram	
namespace	http://fedex.com/ws/addressvalidation/v2
children	**ns:HighestSeverity ns:Notifications ns:TransactionDetail ns:Version ns:ReplyTimestamp ns:AddressResults**
used by	element **AddressValidationReply**
source	see below

```
<xs:complexType name="AddressValidationReply">
  <xs:sequence>
    <xs:element name="HighestSeverity" type="ns:NotificationSeverityType"/>
    <xs:element name="Notifications" type="ns:Notification" maxOccurs="unbounded"/>
    <xs:element name="TransactionDetail" type="ns:TransactionDetail" minOccurs="0"/>
```

	`<xs:element name="Version" type="ns:VersionId" minOccurs="0"/>`
	`<xs:element name="ReplyTimestamp" type="xs:dateTime"/>`
	`<xs:element name="AddressResults" type="ns:AddressValidationResult" maxOccurs="100"/>`
	`</xs:sequence>`
	`</xs:complexType>`

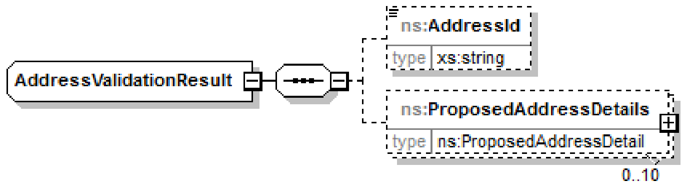
complexType **AddressValidationRequest**

diagram	 AddressValidationRequest ns:WebAuthenticationDetail type ns:WebAuthenticationDetail The descriptive data to be used in authentication of the sender's identity (and right to use FedEx web services). ns:ClientDetail type ns:ClientDetail Descriptive data identifying the client submitting the transaction. ns:TransactionDetail type ns:TransactionDetail Descriptive data for this customer transaction. The TransactionDetail from the request is echoed back to the caller in the corresponding reply. ns:Version 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). ns:RequestTimestamp type xs:dateTime ns:Options type ns:AddressValidationOptions ns:AddressesToValidate type ns:AddressToValidate 1..100
namespace	http://fedex.com/ws/addressvalidation/v2

children	**ns:WebAuthenticationDetail ns:ClientDetail ns:TransactionDetail ns:Version ns:RequestTimestamp ns:Options ns:AddressesToValidate**
used by	element **AddressValidationRequest**
source	see below

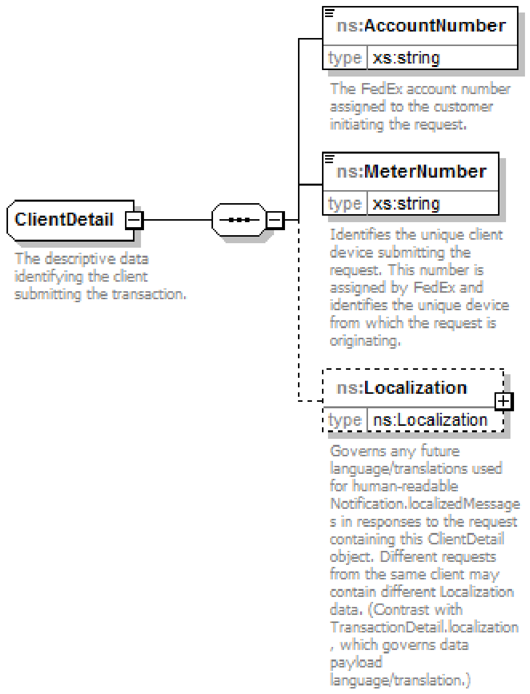
```
<xs:complexType name="AddressValidationRequest">
  <xs:sequence>
    <xs:element name="WebAuthenticationDetail" type="ns:WebAuthenticationDetail">
      <xs:annotation>
        <xs:documentation>The descriptive data to be used in authentication of the sender's identity (and right to use FedEx web services).</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="ClientDetail" type="ns:ClientDetail">
      <xs:annotation>
        <xs:documentation>Descriptive data identifying the client submitting the transaction.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="TransactionDetail" type="ns:TransactionDetail" minOccurs="0">
      <xs:annotation>
        <xs:documentation>Descriptive data for this customer transaction. The TransactionDetail from the request is echoed back to the caller in the corresponding reply.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="Version" type="ns:VersionId">
      <xs:annotation>
        <xs:documentation>Identifies the version/level of a service operation expected by a caller (in each request) and performed by the callee (in each reply).</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="RequestTimestamp" type="xs:dateTime"/>
    <xs:element name="Options" type="ns:AddressValidationOptions"/>
    <xs:element name="AddressesToValidate" type="ns:AddressToValidate" maxOccurs="100"/>
  </xs:sequence>
</xs:complexType>
```

complexType **AddressValidationResult**

diagram	 AddressValidationResult ns:AddressId type xs:string ns:ProposedAddressDetails type ns:ProposedAddressDetail 0..10
namespace	http://fedex.com/ws/addressvalidation/v2
children	**ns:AddressId ns:ProposedAddressDetails**
used by	element **AddressValidationReply/AddressResults**

source	`<xs:complexType name="AddressValidationResult">`
	  `<xs:sequence>`
	    `<xs:element name="AddressId" type="xs:string" minOccurs="0"/>`
	    `<xs:element name="ProposedAddressDetails" type="ns:ProposedAddressDetail" minOccurs="0" maxOccurs="10"/>`
	  `</xs:sequence>`
	`</xs:complexType>`

complexType **ClientDetail**

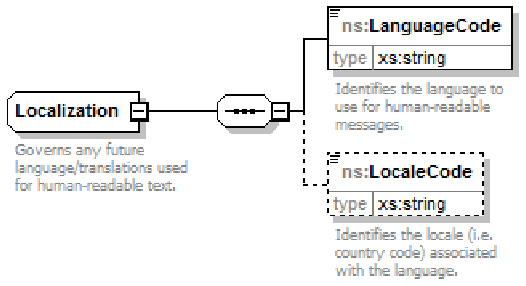
diagram	 ns:AccountNumber type xs:string The FedEx account number assigned to the customer initiating the request. ns:MeterNumber type xs:string Identifies the unique client device submitting the request. This number is assigned by FedEx and identifies the unique device from which the request is originating. ClientDetail The descriptive data identifying the client submitting the transaction. ns:Localization type ns:Localization 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.)
namespace	http://fedex.com/ws/addressvalidation/v2
children	**ns:AccountNumber ns:MeterNumber ns:Localization**
used by	element **AddressValidationRequest/ClientDetail**
annotation	documentation The descriptive data identifying the client submitting the transaction.

source

```
<xs:complexType name="ClientDetail">
  <xs:annotation>
    <xs:documentation>The descriptive data identifying the client submitting the transaction.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="AccountNumber" type="xs:string">
      <xs:annotation>
        <xs:documentation>The FedEx account number assigned to the customer initiating the request.</xs:documentation>
        <xs:appinfo>
          <xs:MaxLength>12</xs:MaxLength>
        </xs:appinfo>
```

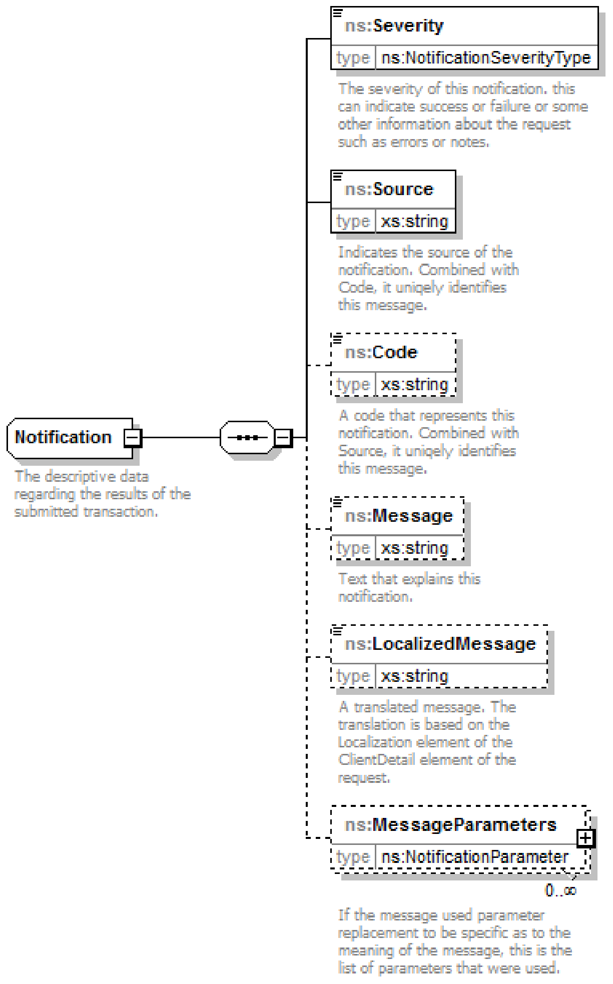
	```
	      </xs:annotation>
	    </xs:element>
	    <xs:element name="MeterNumber" type="xs:string">
	      <xs:annotation>
	        <xs:documentation>Identifies the unique client device submitting the request. This number is assigned by FedEx and identifies the unique device from which the request is originating.</xs:documentation>
	        <xs:appinfo>
	          <xs:MaxLength>10</xs:MaxLength>
	        </xs:appinfo>
	      </xs:annotation>
	    </xs:element>
	    <xs:element name="Localization" type="ns:Localization" minOccurs="0">
	      <xs:annotation>
	        <xs:documentation>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.)</xs:documentation>
	      </xs:annotation>
	    </xs:element>
	  </xs:sequence>
	</xs:complexType>
	```

complexType **Localization**

diagram	 ns:LanguageCode type xs:string Identifies the language to use for human-readable messages. Localization Governs any future language/translations used for human-readable text. ns:LocaleCode type xs:string Identifies the locale (i.e. country code) associated with the language.
namespace	http://fedex.com/ws/addressvalidation/v2
children	**ns:LanguageCode ns:LocaleCode**
used by	elements **ClientDetail/Localization TransactionDetail/Localization**
annotation	documentation Governs any future language/translations used for human-readable text.
source	`<xs:complexType name="Localization">`

```xml
    <xs:annotation>
      <xs:documentation>Governs any future language/translations used for human-readable text.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="LanguageCode" type="xs:string">
        <xs:annotation>
          <xs:documentation>Identifies the language to use for human-readable messages.</xs:documentation>
          <xs:appinfo>
            <xs:MaxLength>2</xs:MaxLength>
          </xs:appinfo>
        </xs:annotation>
      </xs:element>
      <xs:element name="LocaleCode" type="xs:string" minOccurs="0">
        <xs:annotation>
          <xs:documentation>Identifies the locale (i.e.   country code) associated with the language.</xs:documentation>
          <xs:appinfo>
            <xs:MaxLength>2</xs:MaxLength>
          </xs:appinfo>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
```

complexType **Notification**

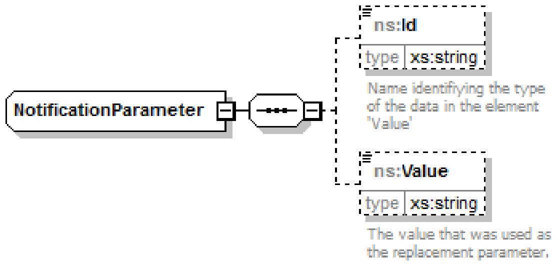
diagram	 ns:Severity type ns:NotificationSeverityType The severity of this notification. this can indicate success or failure or some other information about the request such as errors or notes. ns:Source type xs:string Indicates the source of the notification. Combined with Code, it uniqely identifies this message. ns:Code type xs:string A code that represents this notification. Combined with Source, it uniqely identifies this message. ns:Message type xs:string Text that explains this notification. ns:LocalizedMessage type xs:string A translated message. The translation is based on the Localization element of the ClientDetail element of the request. ns:MessageParameters type ns:NotificationParameter 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. Notification The descriptive data regarding the results of the submitted transaction.
namespace	http://fedex.com/ws/addressvalidation/v2
children	**ns:Severity ns:Source ns:Code ns:Message ns:LocalizedMessage ns:MessageParameters**
used by	element **AddressValidationReply/Notifications**
annotation	documentation The descriptive data regarding the results of the submitted transaction.

source	`<xs:complexType name="Notification">`
	`  <xs:annotation>`

```xml
    <xs:documentation>The descriptive data regarding the results of the submitted transaction.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="Severity" type="ns:NotificationSeverityType">
      <xs:annotation>
        <xs:documentation>The severity of this notification. this can indicate success or failure or some other information about the
request such as errors or notes.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="Source" type="xs:string">
      <xs:annotation>
        <xs:documentation>Indicates the source of the notification. Combined with Code, it uniqely 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 uniqely identifies this
message.</xs:documentation>
        <xs:appinfo>
          <xs:MaxLength>8</xs: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>
          <xs:MaxLength>255</xs: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>
          <xs:MaxLength>TBD</xs:MaxLength>
        </xs:appinfo>
      </xs:annotation>
    </xs:element>
    <xs:element name="MessageParameters" type="ns:NotificationParameter" minOccurs="0" maxOccurs="unbounded">
```

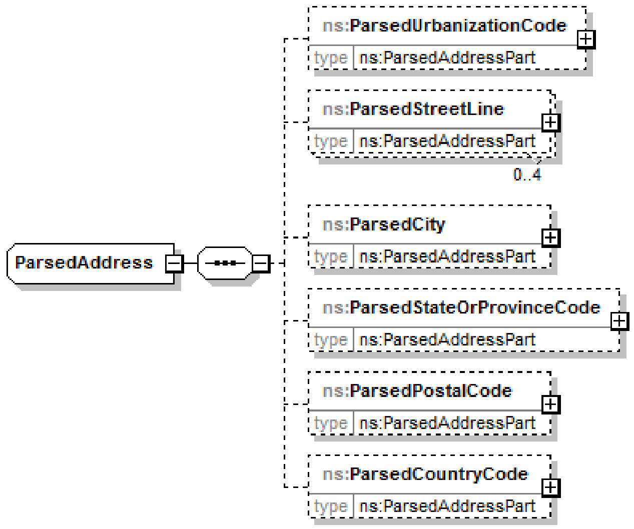
	`<xs:annotation>`
	`<xs:documentation>If the message used parameter replacement to be specific as to the meaning of the message, this is the list of parameters that were used.</xs:documentation>`
	`<xs:appinfo>`
	`<xs:MaxLength>TBD</xs:MaxLength>`
	`</xs:appinfo>`
	`</xs:annotation>`
	`</xs:element>`
	`</xs:sequence>`
	`</xs:complexType>`

complexType **NotificationParameter**

diagram	 ns:Id type xs:string Name identifiying the type of the data in the element 'Value' NotificationParameter ns:Value type xs:string The value that was used as the replacement parameter.
namespace	http://fedex.com/ws/addressvalidation/v2
children	**ns:Id ns:Value**
used by	element **Notification/MessageParameters**

source	`<xs:complexType name="NotificationParameter">`
	`<xs:sequence>`
	`<xs:element name="Id" type="xs:string" minOccurs="0">`
	`<xs:annotation>`
	`<xs:documentation>Name identifiying the type of the data in the element 'Value'</xs:documentation>`
	`</xs:annotation>`
	`</xs:element>`
	`<xs:element name="Value" type="xs:string" minOccurs="0">`
	`<xs:annotation>`
	`<xs:documentation>The value that was used as the replacement parameter.</xs:documentation>`
	`</xs:annotation>`
	`</xs:element>`
	`</xs:sequence>`
	`</xs:complexType>`

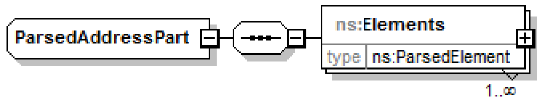
complexType **ParsedAddress**

diagram	 ns:ParsedUrbanizationCode type ns:ParsedAddressPart ns:ParsedStreetLine type ns:ParsedAddressPart 0..4 ParsedAddress ns:ParsedCity type ns:ParsedAddressPart ns:ParsedStateOrProvinceCode type ns:ParsedAddressPart ns:ParsedPostalCode type ns:ParsedAddressPart ns:ParsedCountryCode type ns:ParsedAddressPart
namespace	http://fedex.com/ws/addressvalidation/v2
children	**ns:ParsedUrbanizationCode ns:ParsedStreetLine ns:ParsedCity ns:ParsedStateOrProvinceCode ns:ParsedPostalCode ns:ParsedCountryCode**
used by	element **ProposedAddressDetail/ParsedAddress**

source

```
<xs:complexType name="ParsedAddress">
  <xs:sequence>
    <xs:element name="ParsedUrbanizationCode" type="ns:ParsedAddressPart" minOccurs="0"/>
    <xs:element name="ParsedStreetLine" type="ns:ParsedAddressPart" minOccurs="0" maxOccurs="4"/>
    <xs:element name="ParsedCity" type="ns:ParsedAddressPart" minOccurs="0"/>
    <xs:element name="ParsedStateOrProvinceCode" type="ns:ParsedAddressPart" minOccurs="0"/>
    <xs:element name="ParsedPostalCode" type="ns:ParsedAddressPart" minOccurs="0"/>
    <xs:element name="ParsedCountryCode" type="ns:ParsedAddressPart" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

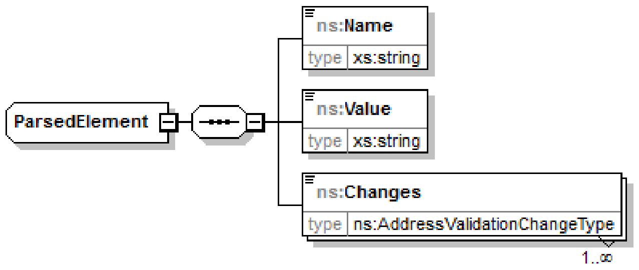
complexType **ParsedAddressPart**

diagram	 ParsedAddressPart ns:Elements type ns:ParsedElement 1..∞

namespace	http://fedex.com/ws/addressvalidation/v2
children	**ns:Elements**
used by	elements **ParsedAddress/ParsedCity ProposedAddressDetail/ParsedCompanyName ParsedAddress/ParsedCountryCode ParsedAddress/ParsedPostalCode ParsedAddress/ParsedStateOrProvinceCode ParsedAddress/ParsedStreetLine ParsedAddress/ParsedUrbanizationCode**

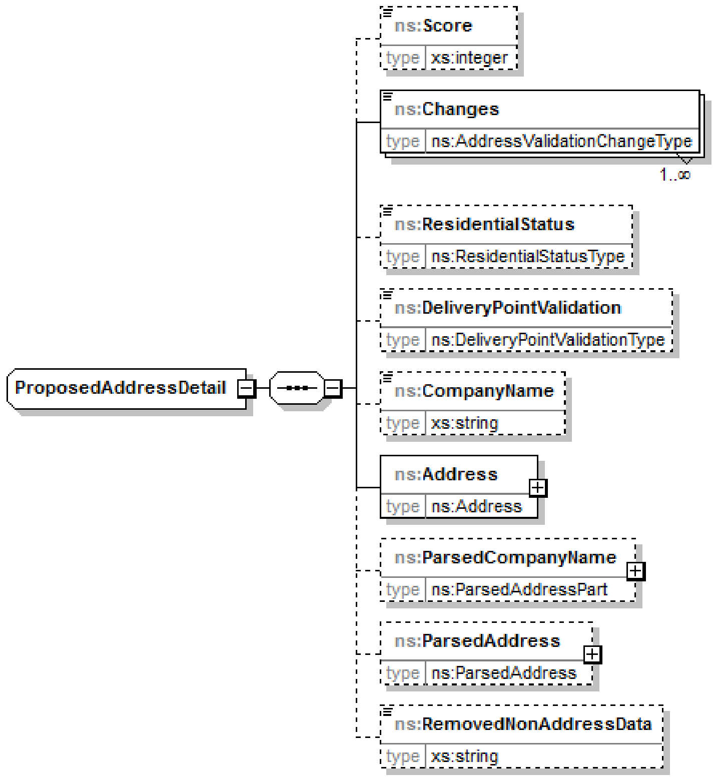
source	`<xs:complexType name="ParsedAddressPart">`
	`  <xs:sequence>`
	`    <xs:element name="Elements" type="ns:ParsedElement" maxOccurs="unbounded"/>`
	`  </xs:sequence>`
	`</xs:complexType>`

complexType **ParsedElement**

diagram	 ParsedElement ns:Name type xs:string ns:Value type xs:string ns:Changes type ns:AddressValidationChangeType 1..∞
namespace	http://fedex.com/ws/addressvalidation/v2
children	**ns:Name ns:Value ns:Changes**
used by	element **ParsedAddressPart/Elements**

source	`<xs:complexType name="ParsedElement">`
	`  <xs:sequence>`
	`    <xs:element name="Name" type="xs:string"/>`
	`    <xs:element name="Value" type="xs:string"/>`
	`    <xs:element name="Changes" type="ns:AddressValidationChangeType" maxOccurs="unbounded"/>`
	`  </xs:sequence>`
	`</xs:complexType>`

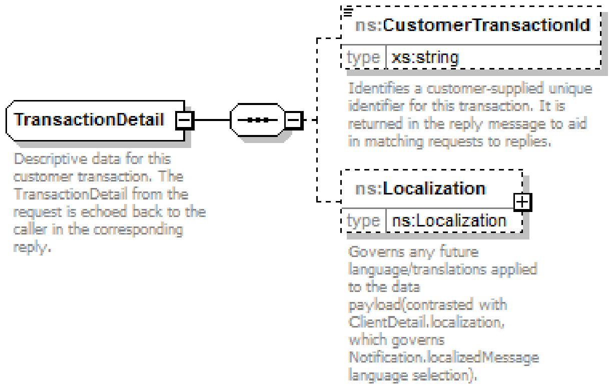
complexType **ProposedAddressDetail**

diagram	 ProposedAddressDetail ns:Score type xs:integer ns:Changes type ns:AddressValidationChangeType 1..∞ ns:ResidentialStatus type ns:ResidentialStatusType ns:DeliveryPointValidation type ns:DeliveryPointValidationType ns:CompanyName type xs:string ns:Address type ns:Address ns:ParsedCompanyName type ns:ParsedAddressPart ns:ParsedAddress type ns:ParsedAddress ns:RemovedNonAddressData type xs:string
namespace	http://fedex.com/ws/addressvalidation/v2
children	**ns:Score ns:Changes ns:ResidentialStatus ns:DeliveryPointValidation ns:CompanyName ns:Address ns:ParsedCompanyName ns:ParsedAddress ns:RemovedNonAddressData**
used by	element **AddressValidationResult/ProposedAddressDetails**

source	`<xs:complexType name="ProposedAddressDetail">`
	  `<xs:sequence>`
	    `<xs:element name="Score" type="xs:integer" minOccurs="0"/>`
	    `<xs:element name="Changes" type="ns:AddressValidationChangeType" maxOccurs="unbounded"/>`
	    `<xs:element name="ResidentialStatus" type="ns:ResidentialStatusType" minOccurs="0"/>`
	    `<xs:element name="DeliveryPointValidation" type="ns:DeliveryPointValidationType" minOccurs="0"/>`
	    `<xs:element name="CompanyName" type="xs:string" minOccurs="0"/>`
	    `<xs:element name="Address" type="ns:Address"/>`
	    `<xs:element name="ParsedCompanyName" type="ns:ParsedAddressPart" minOccurs="0"/>`

	`<xs:element name="ParsedAddress" type="ns:ParsedAddress" minOccurs="0"/>`
	`<xs:element name="RemovedNonAddressData" type="xs:string" minOccurs="0"/>`
	`</xs:sequence>`
	`</xs:complexType>`

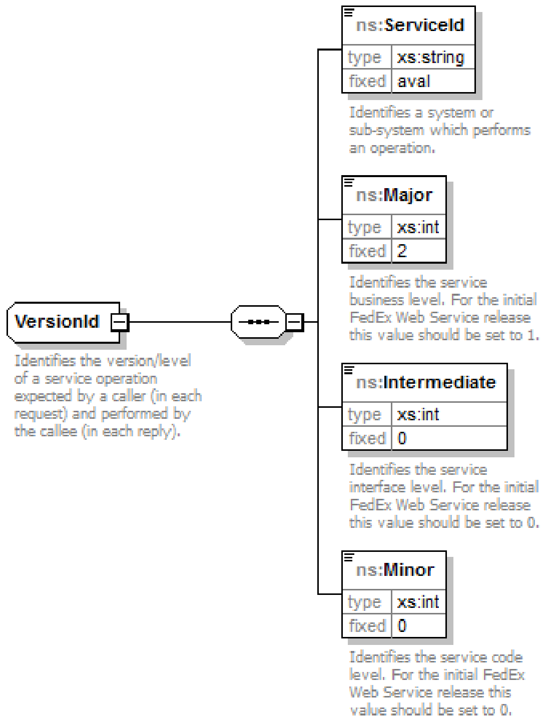
complexType **TransactionDetail**

diagram	 TransactionDetail Descriptive data for this customer transaction. The TransactionDetail from the request is echoed back to the caller in the corresponding reply. ns:CustomerTransactionId 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. ns:Localization type ns:Localization Governs any future language/translations applied to the data payload(contrasted with ClientDetail.localization, which governs Notification.localizedMessage language selection).
namespace	http://fedex.com/ws/addressvalidation/v2
children	**ns:CustomerTransactionId ns:Localization**
used by	elements **AddressValidationRequest/TransactionDetail AddressValidationReply/TransactionDetail**
annotation	documentation Descriptive data for this customer transaction. The TransactionDetail from the request is echoed back to the caller in the corresponding reply.

source	`<xs:complexType name="TransactionDetail">`
	`<xs:annotation>`
	`<xs:documentation>Descriptive data for this customer transaction. The TransactionDetail from the request is echoed back to the caller in the corresponding reply.</xs:documentation>`
	`</xs:annotation>`
	`<xs:sequence>`
	`<xs:element name="CustomerTransactionId" type="xs:string" minOccurs="0">`
	`<xs:annotation>`
	`<xs:documentation>Identifies a customer-supplied unique identifier for this transaction. It is returned in the reply message to aid in matching requests to replies.</xs:documentation>`
	`<xs:appinfo>`
	`<xs:MaxLength>40</xs:MaxLength>`
	`</xs:appinfo>`
	`</xs:annotation>`

	`</xs:element>`
	`<xs:element name="Localization" type="ns:Localization" minOccurs="0">`
	`<xs:annotation>`
	`<xs:documentation>Governs any future language/translations applied to the data payload(contrasted with ClientDetail.localization, which governs Notification.localizedMessage language selection).</xs:documentation>`
	`</xs:annotation>`
	`</xs:element>`
	`</xs:sequence>`
	`</xs:complexType>`

complexType **VersionId**

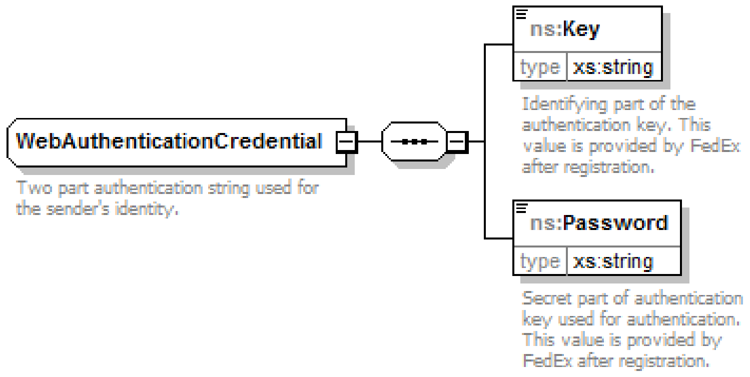
diagram	 ns:ServiceId type xs:string fixed aval Identifies a system or sub-system which performs an operation. ns:Major type xs:int fixed 2 Identifies the service business level. For the initial FedEx Web Service release this value should be set to 1. VersionId Identifies the version/level of a service operation expected by a caller (in each request) and performed by the callee (in each reply). ns:Intermediate type xs:int fixed 0 Identifies the service interface level. For the initial FedEx Web Service release this value should be set to 0. ns:Minor type xs:int fixed 0 Identifies the service code level. For the initial FedEx Web Service release this value should be set to 0.
namespace	http://fedex.com/ws/addressvalidation/v2
children	**ns:ServiceId ns:Major ns:Intermediate ns:Minor**
used by	elements **AddressValidationRequest/Version AddressValidationReply/Version**

annotation	documentation Identifies the version/level of a service operation expected by a caller (in each request) and performed by the callee (in each reply).

source

```
<xs:complexType name="VersionId">
  <xs:annotation>
    <xs:documentation>Identifies the version/level of a service operation expected by a caller (in each request) and performed by
the callee (in each reply).</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="ServiceId" type="xs:string" fixed="aval">
      <xs:annotation>
        <xs:documentation>Identifies a system or sub-system which performs an operation.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="Major" type="xs:int" fixed="2">
      <xs:annotation>
        <xs:documentation>Identifies the service business level. For the initial FedEx Web Service release this value should be set
to 1.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="Intermediate" type="xs:int" fixed="0">
      <xs:annotation>
        <xs:documentation>Identifies the service interface level. For the initial FedEx Web Service release this value should be set
to 0.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="Minor" type="xs:int" fixed="0">
      <xs:annotation>
        <xs:documentation>Identifies the service code level. For the initial FedEx Web Service release this value should be set to
0.</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

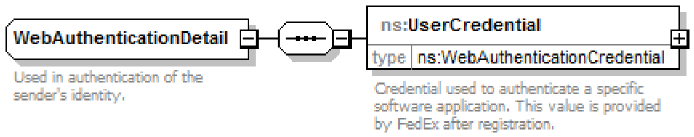
complexType **WebAuthenticationCredential**

diagram	 WebAuthenticationCredential Two part authentication string used for the sender's identity. ns:Key type xs:string Identifying part of the authentication key. This value is provided by FedEx after registration. ns:Password type xs:string Secret part of authentication key used for authentication. This value is provided by FedEx after registration.
namespace	http://fedex.com/ws/addressvalidation/v2
children	**ns:Key ns:Password**
used by	element **WebAuthenticationDetail/UserCredential**
annotation	documentation Two part authentication string used for the sender's identity.
source	see below

```
<xs:complexType name="WebAuthenticationCredential">
  <xs:annotation>
    <xs:documentation>Two part authentication string used for the sender's identity.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="Key" type="xs:string">
      <xs:annotation>
        <xs:documentation>Identifying part of the authentication key. This value is provided by FedEx after
registration.</xs:documentation>
        <xs:appinfo>
          <xs:MaxLength>16</xs:MaxLength>
        </xs:appinfo>
      </xs:annotation>
    </xs:element>
    <xs:element name="Password" type="xs:string">
      <xs:annotation>
        <xs:documentation>Secret part of authentication key used for authentication. This value is provided by FedEx after
registration.</xs:documentation>
        <xs:appinfo>
          <xs:MaxLength>25</xs:MaxLength>
        </xs:appinfo>
      </xs:annotation>
```

	```
	    </xs:element>
	  </xs:sequence>
	</xs:complexType>
	```

complexType **WebAuthenticationDetail**

diagram	 WebAuthenticationDetail Used in authentication of the sender's identity. ns:UserCredential type ns:WebAuthenticationCredential Credential used to authenticate a specific software application. This value is provided by FedEx after registration.
namespace	http://fedex.com/ws/addressvalidation/v2
children	**ns:UserCredential**
used by	element **AddressValidationRequest/WebAuthenticationDetail**
annotation	documentation Used in authentication of the sender's identity.

source	
	```
	<xs:complexType name="WebAuthenticationDetail">
	  <xs:annotation>
	    <xs:documentation>Used in authentication of the sender's identity.</xs:documentation>
	  </xs:annotation>
	  <xs:sequence>
	    <xs:element name="UserCredential" type="ns:WebAuthenticationCredential">
	      <xs:annotation>
	        <xs:documentation>Credential used to authenticate a specific software application. This value is provided by FedEx after registration.</xs:documentation>
	      </xs:annotation>
	    </xs:element>
	  </xs:sequence>
	</xs:complexType>
	```

simpleType **AddressValidationAccuracyType**

namespace	http://fedex.com/ws/addressvalidation/v2
type	restriction of **xs:string**
used by	elements **AddressValidationOptions/CompanyNameAccuracy AddressValidationOptions/DirectionalAccuracy AddressValidationOptions/StreetAccuracy**
facets	Kind / Value / annotation enumeration / EXACT

	enumeration TIGHT enumeration MEDIUM enumeration LOOSE

source	`<xs:simpleType name="AddressValidationAccuracyType">`
	`  <xs:restriction base="xs:string">`
	`    <xs:enumeration value="EXACT"/>`
	`    <xs:enumeration value="TIGHT"/>`
	`    <xs:enumeration value="MEDIUM"/>`
	`    <xs:enumeration value="LOOSE"/>`
	`  </xs:restriction>`
	`</xs:simpleType>`

simpleType **AddressValidationChangeType**

namespace	http://fedex.com/ws/addressvalidation/v2
type	restriction of **xs:string**
used by	elements **ProposedAddressDetail/Changes ParsedElement/Changes**
facets	see below

Kind	Value	annotation
enumeration	APARTMENT_NUMBER_NOT_FOUND	documentation (EAS 100) Apartment number not found for this address.
enumeration	APARTMENT_NUMBER_REQUIRED	documentation (EAS 101) Address requires apartment number.
enumeration	NORMALIZED	documentation (EAS 102) Address normalized - abbreviations applied.
enumeration	REMOVED_DATA	documentation (EAS 103) Dropped data.
enumeration	BOX_NUMBER_REQUIRED	documentation (EAS 104) Address requires box number.

	enumeration	NO_CHANGES	documentation (EAS 200) Match - no changes applied to input address.
	enumeration	MODIFIED_TO_ACHIEVE_MATCH	documentation (EAS 201) Address modified to achieve match.
	enumeration	STREET_RANGE_MATCH	documentation (EAS 202) Match to street range.
	enumeration	BOX_NUMBER_MATCH	documentation (EAS 203) Match to box number.
	enumeration	RR_OR_HC_MATCH	documentation (EAS 204) Match to Rural Route (RR) / Highway Contract (HC) address.
	enumeration	CITY_MATCH	documentation (EAS 205) Match to city (non-US, non-Canada).
	enumeration	POSTAL_CODE_MATCH	documentation (EAS 206) Match to postal code only (non-street)
	enumeration	RR_OR_HC_BOX_NUMBER_NEEDED	documentation (EAS 207) Need box number for Rural Route / Highway Contract (HC) match.
	enumeration	FORMATTED_FOR_COUNTRY	documentation (EAS 208) Formatting performed for country (non-US, non-Canada).
	enumeration	APO_OR_FPO_MATCH	documentation

			(EAS 209) Match to military address (e.g. APO/FPO).
	enumeration	GENERAL_DELIVERY_MATCH	documentation (EAS 210) Match to general delivery.
	enumeration	FIELD_TRUNCATED	documentation (EAS 211) Address exceeded 35 character plug-in limit.
	enumeration	UNABLE_TO_APPEND_NON_ADDRESS_DATA	documentation (EAS 212) Unable to append non-address; data 35 character limit imposed.
	enumeration	INSUFFICIENT_DATA	documentation (EAS 300) Insufficient data for address verification.
	enumeration	HOUSE_OR_BOX_NUMBER_NOT_FOUND	documentation (EAS 301) Address (house or box number) not found.
	enumeration	POSTAL_CODE_NOT_FOUND	documentation (EAS 303) Postal code not found.
	enumeration	INVALID_COUNTRY	documentation (EAS 305) Invalid country.
	enumeration	SERVICE_UNAVAILABLE_FOR_ADDRESS	documentation (EAS 400) Service unavailable for request address.

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

```
            <xs:documentation>
                                                    (EAS 201) Address modified to achieve match.
                                                </xs:documentation>
          </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="STREET_RANGE_MATCH">
          <xs:annotation>
            <xs:documentation>
                                                    (EAS 202) Match to street range.
                                                </xs:documentation>
          </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="BOX_NUMBER_MATCH">
          <xs:annotation>
            <xs:documentation>
                                                    (EAS 203) Match to box number.
                                                </xs:documentation>
          </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="RR_OR_HC_MATCH">
          <xs:annotation>
            <xs:documentation>
                                                    (EAS 204) Match to Rural Route (RR) / Highway Contract (HC)
address.
                                                </xs:documentation>
          </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="CITY_MATCH">
          <xs:annotation>
            <xs:documentation>
                                                    (EAS 205) Match to city (non-US, non-Canada).
                                                </xs:documentation>
          </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="POSTAL_CODE_MATCH">
          <xs:annotation>
            <xs:documentation>
                                                    (EAS 206) Match to postal code only (non-street)
                                                </xs:documentation>
          </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="RR_OR_HC_BOX_NUMBER_NEEDED">
```

	```
	        <xs:annotation>
	          <xs:documentation>
	                                        (EAS 207) Need box number for Rural Route / Highway Contract
	(HC) match.
	                                    </xs:documentation>
	        </xs:annotation>
	      </xs:enumeration>
	      <xs:enumeration value="FORMATTED_FOR_COUNTRY">
	        <xs:annotation>
	          <xs:documentation>
	                                        (EAS 208) Formatting performed for country (non-US,
	non-Canada).
	                                    </xs:documentation>
	        </xs:annotation>
	      </xs:enumeration>
	      <xs:enumeration value="APO_OR_FPO_MATCH">
	        <xs:annotation>
	          <xs:documentation>
	                                        (EAS 209) Match to military address (e.g. APO/FPO).
	                                    </xs:documentation>
	        </xs:annotation>
	      </xs:enumeration>
	      <xs:enumeration value="GENERAL_DELIVERY_MATCH">
	        <xs:annotation>
	          <xs:documentation>
	                                        (EAS 210) Match to general delivery.
	                                    </xs:documentation>
	        </xs:annotation>
	      </xs:enumeration>
	      <xs:enumeration value="FIELD_TRUNCATED">
	        <xs:annotation>
	          <xs:documentation>
	                                        (EAS 211) Address exceeded 35 character plug-in limit.
	                                    </xs:documentation>
	        </xs:annotation>
	      </xs:enumeration>
	      <xs:enumeration value="UNABLE_TO_APPEND_NON_ADDRESS_DATA">
	        <xs:annotation>
	          <xs:documentation>
	                                        (EAS 212) Unable to append non-address; data 35 character limit
	imposed.
	                                    </xs:documentation>
	```

```
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="INSUFFICIENT_DATA">
        <xs:annotation>
          <xs:documentation>
                                                    (EAS 300) Insufficient data for address verification.
                                                </xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="HOUSE_OR_BOX_NUMBER_NOT_FOUND">
        <xs:annotation>
          <xs:documentation>
                                                    (EAS 301) Address (house or box number) not found.
                                                </xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="POSTAL_CODE_NOT_FOUND">
        <xs:annotation>
          <xs:documentation>
                                                    (EAS 303) Postal code not found.
                                                </xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="INVALID_COUNTRY">
        <xs:annotation>
          <xs:documentation>
                                                    (EAS 305) Invalid country.
                                                </xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="SERVICE_UNAVAILABLE_FOR_ADDRESS">
        <xs:annotation>
          <xs:documentation>
                                                    (EAS 400) Service unavailable for request address.
                                                </xs:documentation>
        </xs:annotation>
      </xs:enumeration>
    </xs:restriction>
  </xs:simpleType>
```

simpleType **DeliveryPointValidationType**

namespace	http://fedex.com/ws/addressvalidation/v2
type	restriction of **xs:string**
used by	element **ProposedAddressDetail/DeliveryPointValidation**
facets	Kind / Value / annotation enumeration CONFIRMED enumeration UNCONFIRMED enumeration UNAVAILABLE
source	see below

Kind	Value	annotation
enumeration	CONFIRMED	
enumeration	UNCONFIRMED	
enumeration	UNAVAILABLE	

```
<xs:simpleType name="DeliveryPointValidationType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="CONFIRMED"/>
    <xs:enumeration value="UNCONFIRMED"/>
    <xs:enumeration value="UNAVAILABLE"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType **NotificationSeverityType**

namespace	http://fedex.com/ws/addressvalidation/v2
type	restriction of **xs:string**
used by	elements **AddressValidationReply/HighestSeverity Notification/Severity**
annotation	documentation Identifies the set of severity values for a Notification.

facets:

Kind	Value	annotation
enumeration	ERROR	
enumeration	FAILURE	
enumeration	NOTE	
enumeration	SUCCESS	
enumeration	WARNING	

source:

```
<xs:simpleType name="NotificationSeverityType">
  <xs:annotation>
    <xs:documentation>Identifies the set of severity values for a Notification.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="ERROR"/>
    <xs:enumeration value="FAILURE"/>
    <xs:enumeration value="NOTE"/>
```

	`<xs:enumeration value="SUCCESS"/>`
	`<xs:enumeration value="WARNING"/>`
	`</xs:restriction>`
	`</xs:simpleType>`

simpleType **ResidentialStatusType**

namespace	http://fedex.com/ws/addressvalidation/v2
type	restriction of **xs:string**
used by	element **ProposedAddressDetail/ResidentialStatus**
facets	Kind / Value / annotation: enumeration UNDETERMINED enumeration BUSINESS enumeration RESIDENTIAL enumeration INSUFFICIENT_DATA enumeration UNAVAILABLE enumeration NOT_APPLICABLE_TO_COUNTRY
source	see below

Kind	Value	annotation
enumeration	UNDETERMINED	
enumeration	BUSINESS	
enumeration	RESIDENTIAL	
enumeration	INSUFFICIENT_DATA	
enumeration	UNAVAILABLE	
enumeration	NOT_APPLICABLE_TO_COUNTRY	

```
<xs:simpleType name="ResidentialStatusType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="UNDETERMINED"/>
    <xs:enumeration value="BUSINESS"/>
    <xs:enumeration value="RESIDENTIAL"/>
    <xs:enumeration value="INSUFFICIENT_DATA"/>
    <xs:enumeration value="UNAVAILABLE"/>
    <xs:enumeration value="NOT_APPLICABLE_TO_COUNTRY"/>
  </xs:restriction>
</xs:simpleType>
```