

Introduction

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

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

Why should developers be interested in Web Services?

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

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

Document Overview

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

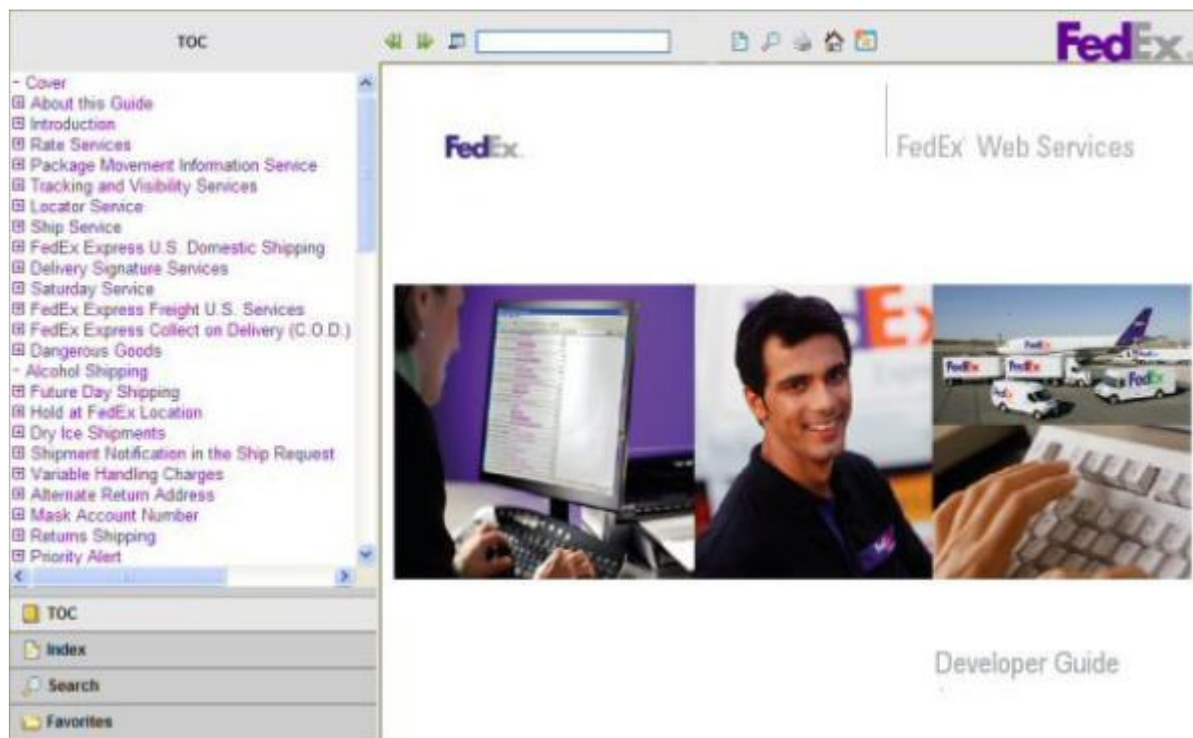
- Introduction (this chapter):
 - o Documentation overview and guidelines, including how to use the Help application and how to print this guide.
 - o Overview information about Web Services, including a high-level description of FedEx Web Services methods.
 - o Coding basics.
 - o Overview information about testing and certifying your application.
- [Rate Services](#) describes services to rate packages.
- [Package Movement Information Service](#) describes how to check service availability, postal codes and route information.
- [Tracking and Visibility Services](#) covers Track Services and includes:
 - o Elements for requesting tracking information, coding notification requests, and requesting [Signature Proof of Delivery \(SPOD\)](#) .
 - o The [Notification](#) service, which can be used to notify senders and recipients of significant shipment events.
 - o Elements for configuring [FedEx InSight®](#).
- [Locator Service](#) describes how to receive the addresses of the nearest FedEx package drop-off locations, including FedEx Office Ship and print Centers.
- [Ship Service](#) provides:
 - o Service details for shipping with FedEx Services.
 - o Service details and coding details for all shipping services, including [FedEx Express U.S. Domestic Shipping](#), [FedEx Ground U.S. Shipping](#), [FedEx Express International Shipping](#), and [FedEx Ground International Shipping](#).
- [Courier Dispatch Services](#) describes how to code the Courier Dispatch and Cancel Courier Dispatch, and check for availability of courier services.
- [SmartPost Shipping](#) describes how to configure FedEx SmartPost® shipping options.
- [Creating a Label](#) describes how to configure, customize and print shipping labels using a variety of options.
- [Address Validation Service](#) explains how to check your shipping addresses for accuracy before shipping.
- [Glossary](#) lists FedEx-specific terms, including acronyms in use in this guide.
- [Appendices A-U](#) describe information including [sample transactions](#), [error codes](#) and the [XML schema](#).

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.


Using the Web Services Online Help

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








Web Services Help opens in your default browser, such as Internet Explorer or Firefox. The first topic—in this case, the cover page—appears in the main window. The Table of Contents (TOC) appears in the navigation column. Under the TOC you can choose the Index, Search, or Favorites options. Each of these features appears in the same column.

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

 Back: Returns you to the previously viewed topic.

 Forward: Goes to the next topic as listed in the TOC.

 Quick Search: Enter  to highlight the term in the current topic. This feature only searches the current topic.

-  Hide Navigation: Hides the left navigation column.
-  Search: Opens the full search tool in the navigation column.
-  Print: Opens the **Print** dialog box. See Printing this Guide or Online Help in the following section.
-  Home: Opens the default topic: in this case, the cover page.
-  Add Topic to Favorites: Saves the current topic to your **Favorites** list.

Printing This Guide or Online Help

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

Printing from the PDF Version

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

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

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

Printing from the WebHelp Version

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

Open WebHelp and click the printer icon .

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

Web Services, WSDL, and SOAP Overview

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

Web Services

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

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

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

WSDL

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

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

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

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

Note: For more information about the WSDL standard, refer to the W3C Web site at w3.org/TR/wsdl.

SOAP

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

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

SOAP Message

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

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

Example: Delete Tag Request (SOAP Message)

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://fedex.com/ws/ship/v8">
<SOAP-ENV:Body>
<DeleteTagRequest>
<WebAuthenticationDetail>
<UserCredential>
<Key>xxxxxxxxxxxxxxxx</Key>
<Password></Password>
</UserCredential>
</WebAuthenticationDetail>
<ClientDetail>
<AccountNumber>xxxxxxxx</AccountNumber>
<MeterNumber>xxxxxx</MeterNumber>
<ClientProductId>WBUS</ClientProductId>
<ClientProductVersion>0200</ClientProductVersion>
</ClientDetail>
<TransactionDetail>
<CustomerTransactionId>DE_Shakeout_wsvc</CustomerTransactionId>
</TransactionDetail>
<Version>
<ServiceId>ship</ServiceId>
<Major>8</Major>
<Intermediate>0</Intermediate>
<Minor>0</Minor>
</Version>
<DispatchLocationId>MQYA</DispatchLocationId>
<DispatchDate>2008-10-08</DispatchDate>
<Payment>
<PaymentType>SENDER</PaymentType>
<Payor>
<AccountNumber>xxxxxxxx</AccountNumber>
```

```

<CountryCode>US</CountryCode>
</Payor>
</Payment>
<ConfirmationNumber>997037200019454</ConfirmationNumber>
</DeleteTagRequest>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

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

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 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. An example of a request using the non-SOAP interface looks like this:

```

<ns:TrackRequest xmlns:ns="http://fedex.com/ws/track/v4"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://fedex.com/ws/track/v4 TrackService_v4.xsd ">
<ns:WebAuthenticationDetail>
<ns:UserCredential>
<ns:Key>xxxxxxxxxxxxxxxx</ns:Key>
<ns>Password>xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx</ns>Password>
</ns:UserCredential>
</ns:WebAuthenticationDetail>
<ns:ClientDetail>
<ns:AccountNumber>000000000</ns:AccountNumber>
<ns:MeterNumber>0000000</ns:MeterNumber>
</ns:ClientDetail>
<ns:TransactionDetail>
<ns:CustomerTransactionId>User Customizable
Field</ns:CustomerTransactionId></ns:TransactionDetail>
<ns:Version>
<ns:ServiceId>trck</ns:ServiceId>
<ns:Major>4</ns:Major>
<ns:Intermediate>0</ns:Intermediate>
<ns:Minor>0</ns:Minor>

```

```

</ns:Version>
<ns:PackageIdentifier>
<ns:Value>tttttttttttttttttt</ns:Value>
<ns:Type>TRACKING_NUMBER_OR_DOORTAG</ns:Type>
</ns:PackageIdentifier>
<ns:IncludeDetailedScans>true</ns:IncludeDetailedScans>
</ns:TrackRequest>

```

Error Handling

Error handling for non-SOAP 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/v7">000000000</ship:AccountNumberx>
</con1:xmlLocation>
<con1:message>Expected element 'AccountNumber@http://fedex.com/ws/ship/v7' before
the end of the content in element
ClientDetail@http://fedex.com/ws/ship/v8</con1:message>
<con1:xmlLocation>
<ship:ClientDetail xmlns:ship="http://fedex.com/ws/ship/v8">
<ship:AccountNumberx>000000000000000000</ship:AccountNumberx>
<ship:MeterNumber>0000000</ship:MeterNumber>
</ship:ClientDetail>
</con1:xmlLocation>
</con1:ValidationFailureDetail>
</con:details>
<con:location>
<con:node>Validate</con:node>
<con:pipeline>Validate_request</con:pipeline>
<con:stage>ValidateRequest</con:stage>
<con:path>request-pipeline</con:path>
</con:location>
</con:fault>
</detail>
</soapenv:Fault>

```


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

Keep in mind that if you use either the SOAP or non-SOAP version of FedEx Web Services, labels are returned as Base64 encoded. To print shipping labels, you must decode labels before sending them to your printer. For more information on Base64 decoding, see [Creating a Label](#).

Non-SOAP HTTP POST Example

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

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

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

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

After formatting your non-SOAP transaction and placing it in a HTTP POST request, 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 element is not set, even after setting it in the code. When you set a Boolean type element to true, you may also need to set the specified element to True.

Implementing FedEx Web Services

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

- FedEx Web Services is designed for use by skilled developers who are familiar with 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 GUI. Instead, Web Services share business logic, data, and processes through a programmatic interface across a network.
- To perform a particular FedEx task such as tracking a package, you need to use a class, module, or function that creates your request, sends it to the FedEx platform, and handles the response.

- Web Services are designed to support any operating system and coding language. Downloadable sample code is available in Java, C#, VB, .Net, and PHP languages.

Understanding the XML Schema

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

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

An XML schema defines:

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

Some important facts about the XML schema:

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

Guide to the XML Schema

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

The top of each service schema includes:

- Schema location and schema file name that ends in an ".xsd" suffix.
- Alphabetical listing of complex types for the documented service.

- Alphabetical listing of schema simple types for the documented service.
- Input or request data type for the documented service.
- Output or reply data type for the documented service.

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

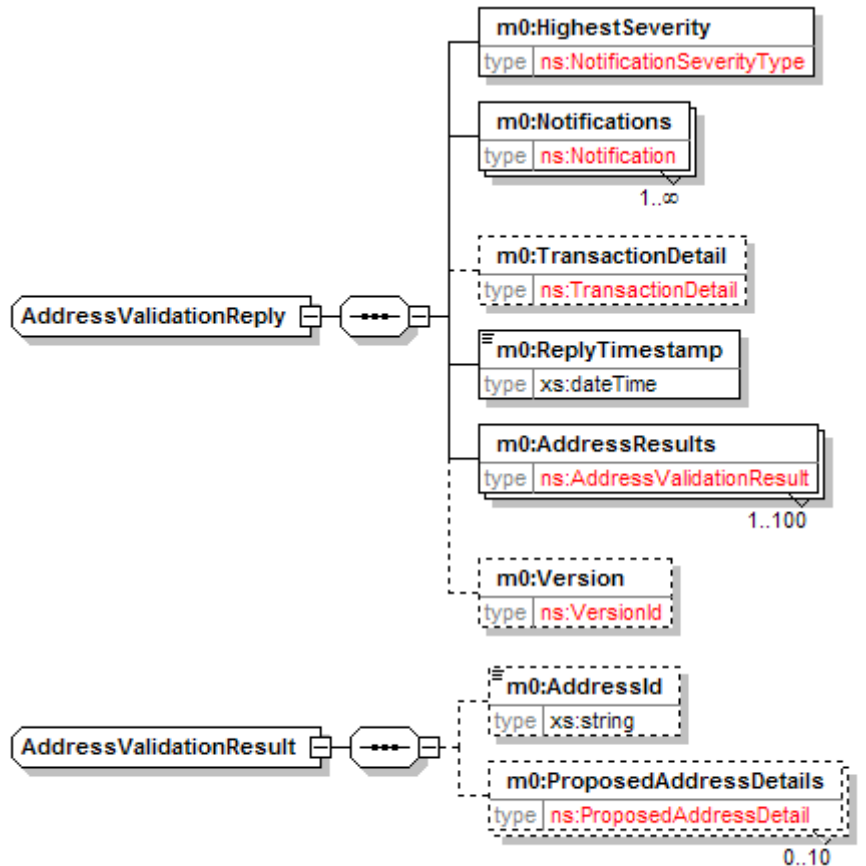
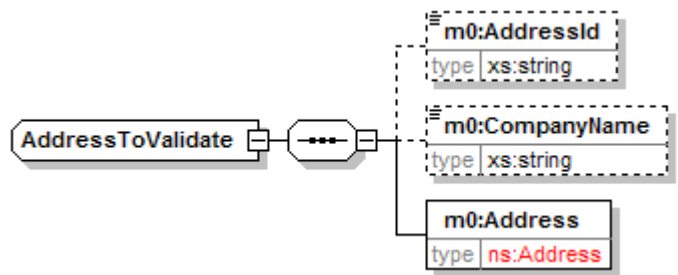
Each table consists of some or all of the following sections: diagram, namespace, children, type, properties, used by, facets, and source.

XML Schema Diagrams

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

Diagram	Description
	<p>Diagrams of a parent element, such as AddressValidationRequest, include connections to the child elements. Child elements can be simple or complex types.</p> <p>A child element connected with a solid line and surrounded by a box with a solid border represents a required type, such as ClientDetail.</p> <p>A child element connected by a dotted line and surrounded by a dotted border represents an optional type (minOccurs="0"), such as TransactionDetail.</p> <p>Note: An element that is defined as minOccurs="0" may be required for some calls.</p> <p>Types that are documented include the documentation directly below the box.</p> <p>All children are linked by name below the diagram.</p>
<p>m0:WebAuthenticationDetail type ns:WebAuthenticationDetail</p> <p>The descriptive data to be used in authentication of the sender's identity (and right to use FedEx web services).</p>	
<p>m0:ClientDetail type ns:ClientDetail</p> <p>Descriptive data identifying the client submitting the transaction.</p>	
<p>m0:TransactionDetail type ns:TransactionDetail</p> <p>Descriptive data for this customer transaction. The TransactionDetail from the request is echoed back to the caller in the corresponding reply.</p>	
<p>m0:Version type ns:VersionId</p> <p>Identifies the version/level of a service operation expected by a caller (in each request) and performed by the callee (in each reply).</p>	
<p>m0:RequestTimestamp type xs:dateTime</p>	
<p>m0:Options type ns:AddressValidationOptions</p>	
<p>m0:AddressesToValidate type ns:AddressToValidate</p> <p>1..100</p>	

<p>children m0:WebAuthenticationDetail m0:ClientDetail m0:TransactionDetail m0:Version m0:RequestTimestamp m0:Options m0:AddressesToValidate</p>	
<p>m0: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).</p>	<p>A box with a single solid border represents a single element that is required.</p> <p>The type can be simple or complex.</p>
<p>m0: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.</p>	<p>A box with a dotted border indicates a single element that is optional.</p> <p>The type can be simple or complex.</p>
	<p>A layered box represents a multiple occurrence element. A solid line represents a required</p>

	<p>multiple occurrence element.</p> <p>The number of possible occurrences appears below the box, as depicted by the AddressResults element.</p> <p>An unbounded number of occurrences is represented by the infinity ∞ symbol (maxOccurs="unbounded"), as depicted by the Notifications type.</p> <p>A layered box with a dotted border represents an optional multiple occurrence type (minOccurs="0"), such as ProposedAddressResults.</p> <p>Note: An element that is defined as minOccurs="0" may be required for some calls.</p>
	<p>A standard type such as "string" appears in black text below element name.</p> <p>A FedEx-specific type such as "Address" appears in red text below the element name.</p>

Required Elements

Most requests to FedEx require the following complex elements:

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

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

Element	Description
WebAuthenticationDetail	The descriptive data to be used in authentication of the sender's identity and right to use FedEx web services.

UserCredential	Credential used to authenticate a specific software application. This value is provided by FedEx after registration.
WebAuthenticationCredential	Two-part authentication string used to verify sender identity.
WebAuthenticationCredential/Key	Unique identifier assigned to each customer as part of their authentication credentials.
WebAuthenticationCredential/Password	Second part of the authentication credential which has a direct relationship with the credential key.

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

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

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

Note: When building a Web-based application for shipping that will be used at multiple locations, include the local FedEx Express® account and meter in the ClientDetail section of the ship transaction. Create a database to hold multiple account and meter numbers for the shipping locations.

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

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

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

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

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

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

Reply Notifications

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

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

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

See [Appendix O: Error Codes](#) for a list of errors by service.

Notification Examples

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

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

Case:

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

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

Case:

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

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

Case:

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

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

Case:

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

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

Case:

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

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

Case:

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

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

Case:

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

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

Case:

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

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

Implementation Process

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

To obtain FedEx Web Services and begin integrating with an application, you will need to access documentation, sample code, and sample service requests and replies with the WSDLs from the FedEx Developer Resource Center Technical Resources. You will also need to obtain a test meter number to engage in real-time online testing in the FedEx-hosted test environment. Note that 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. In order to execute test interactions, you must first include a test account number, test meter number, authentication key, and password in your code. These credentials are provided to registered developers.

Certification

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

Go to Production

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

Requirements for Corporate and Non-Commercial Developers

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

Requirements and Resources for Corporate Developers

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

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

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 through Friday, 7 a.m. to 9 p.m. (CST), 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 Solution Program. If improved time-to-market, cost containment, or specialized knowledge is needed, corporate development planners may want to review the available third-party solutions. To see a list of the solutions provided by the CSP providers, go to the Available CSP Solutions page at fedex.allegis.com/LeadReg.asp.

Requirements for Consultants

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

Requirements and Resources for Commercial Developers

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

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

If you are a commercial developer interested in becoming a FedEx Compatible Solution Provider, visit fedex.com/us/compatiblesolutions/provider/ 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 point to production, perform the following:

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