

SF-LWR Extension Framework : FedEx Shipping Integration

[Functional Overview](#)

[Technical Details](#)

[Sequence Diagram](#)

[Configuration details](#)

[Record - FedEx Shipping Provider\(Shipping Provider \)](#)

[Record - FedEx Shipping Service\(HTTP Service\)](#)

[Source Code](#)

Functional Overview

This document provides details specific to integrating with **FedEx**.

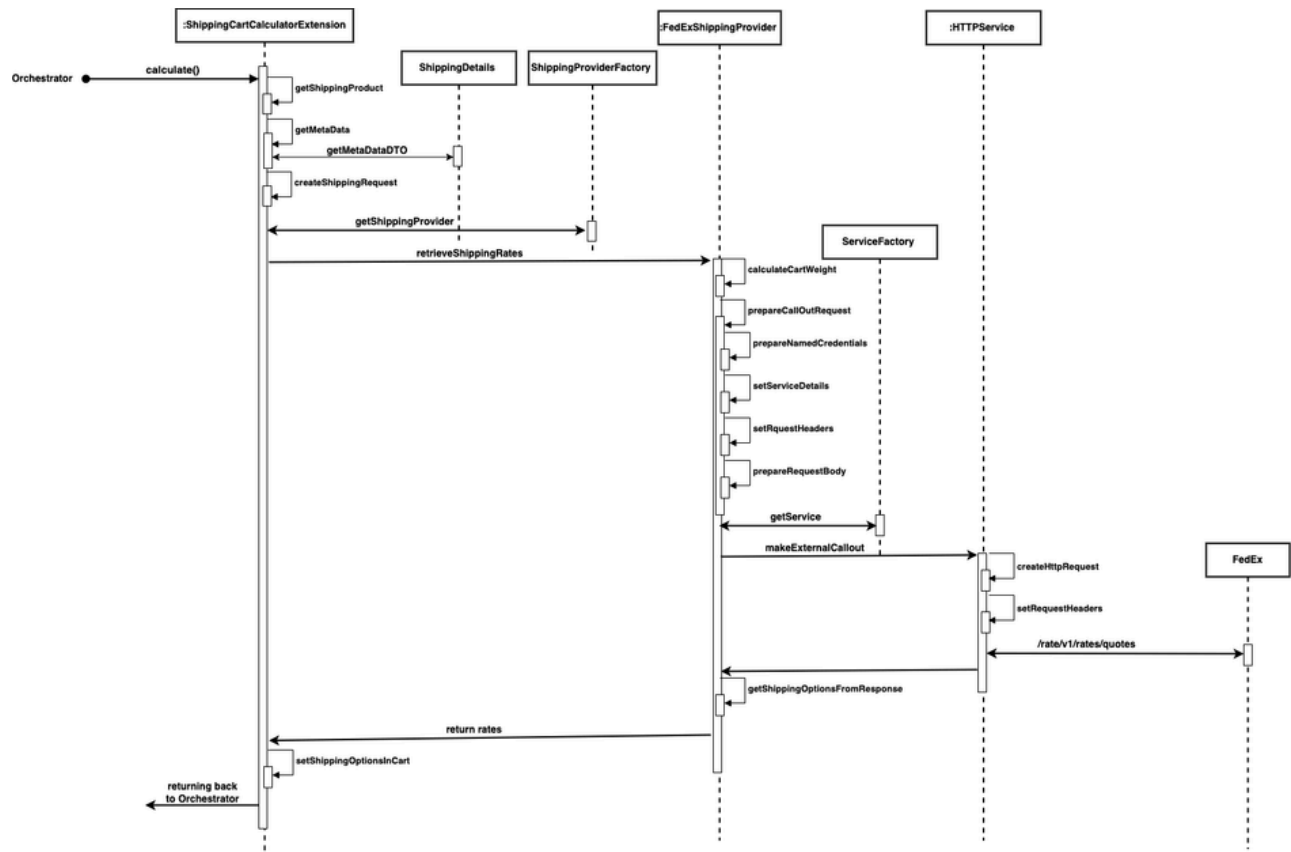
For common details you can refer to parent doc [here](#).

Technical Details

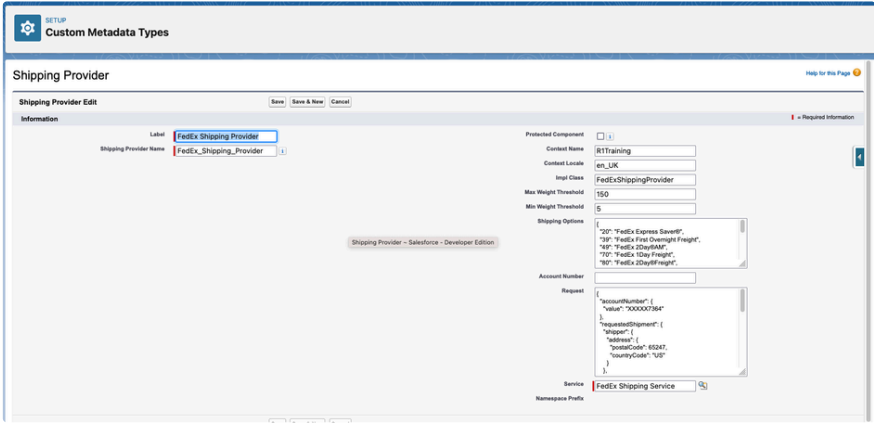
This integration implementation has two parts :

- **Configuration**
 - Custom Meta Data Records
 - FedEx Shipping Provider
 - FedEx Shipping Service
 - Named Credentials
 - FedEx Shipping
 - External Credentials
 - FedEx Authentication
 - FedExCredentials (Principal)
- **Source Code**
 - FedExShippingProvider
 - FedExShippingRateRequest
 - FedExShippingRateResponse

Sequence Diagram



Configuration details

	Section	Task	Details
1	Salesforce Org	Shipping Provider - Create a record for FedEx	<p>Create a record FedEx Shipping Provider for Custom Meta Data Type Shipping Provider</p> 
2	Salesforce Org	HTTP Service - Create a record for FedEx	<p>Create a record - FedEx Shipping Service</p>

Custom Metadata Types

HTTP Service

Help for this Page

HTTP Service Edit

Save
Save & New
Cancel

Information

Label
FedEx Shipping Service

HTTP Service Name
FedEx_Shipping_Service

HTTP Service - Salesforce - Developer Edition

Protected Component
1

Impl Class
FedEx_Shipping

Service Mode
Live

HTTP Method
POST

End Point
/paths/quotes

Service Timeout
5000

Success Codes
200,201

Error Codes
400,403,404,500,503

Auth Codes
401

Mocked Response

```
{
  "transactionId": "APF_SV_RATC_FGDH38154",
  "cust-4feb-bd65-313581b4cedf",
  "customerTransactionId": "customer test",
  "output": [
    "hello"
  ]
}
```

Namespace Prefix

Save
Save & New
Cancel

3

Salesforce Org

Create External Credentials

Create **External Credentials** for OAuth 2.0,

Named Credentials

Named Credentials
External Credentials

1 Items - Sorted by Label

New

Label	Authentication Protocol	Actions
UPS Authentication	OAuth 2.0	

Provide details , like

Name - FedExAuth

Authentication Protocol - OAuth 2.0

Authentication Flow Type - Client Credentials with Secret Flow

Identity Provider URL - <https://apis-sandbox.fedex.com/oauth/token>

Make sure to turn on -

Edit FedEx Authentication

* Label
FedEx Authentication

* Name
FedExAuth

* Authentication Protocol
OAuth 2.0

* Authentication Flow Type
Client Credentials with Client Secret Flow

* Identity Provider URL
https://apis-sandbox.fedex.com/oauth/token

Scope

☒ Pass client credentials in request body ⓘ

Cancel
Save

SETUP > NAMED CREDENTIALS

FedEx Authentication

EditDelete

LabelFedEx AuthenticationNameFedExAuth

Authentication ProtocolOAuth 2.0

Authentication Flow TypeClient Credentials with Client Secret FlowScope

Identity Provider URLhttps://apis-sandbox.fedex.com/oauth/tokenPass client credentials in request body ⓘ

Managed Package Access

Created By Namespace ⓘ

Related Named Credentials

Label	Name	URL
FedEx Shipping	FedEx_Shipping	https://apis-sandbox.fedex.com

Principals

Sequ...	Parameter Name	Client ID	Authentication Status	Actions
1	FedExCredentials	17057792dc3f304545abeb89db9625ab	Configured	ⓘ

Also create a Principal which will store Client ID & Client Secret

Name - FedExCredentials

Edit Principal

* Parameter NameUPSCredentials

* Sequence Number1

Client IDE6LNmA4GRLS5ccqYUtnfyF3tIRUdByAGxlcArGace5YGj

Client Secret.....

Principal Access

Name	Entity Type	ID
B2BBuyer	PermissionSet	OPSHn000001yYV7OAM

CancelSave

4

Salesforce Org

Create Named Credentials

Create a Named Credentials for storing FedEx API URL

Set URL - <https://apis-sandbox.fedex.com>

Under **External Credentials**

Select “**FedEx Authentication**” , to associate with External Credentials (created in previous step)

Under **Callout Options**

Turn on - Generate Authorization Header

Here are the details

SETUP Named Credentials					
Named Credentials External Credentials					
2 Items - Sorted by Label					
Label	Type	URL	External Credential	Actions	
FedEx Shipping	Secured Endpoint	https://apis-sandbox.fedex.com	FedEx Authentication	⋮	
UPS Shipping	Secured Endpoint	https://www.ups.com	UPS Authentication	⋮	

5

Assign Permission

Click **External Credential Principal Access**

B2BBuyer

Find Settings...

Clone

Edit Properties

Manage Assignments

View Summary (Beta)

Permission Set Overview

Description

Allows access to the store. Lets users see products and categories, make purchases, and add products to a wishlist.

License

B2B Buyer Permission Set One Seat

Session Activation Required

☐

Permission Set Groups Added To

0

API Name

B2BBuyer

Namespace Prefix

Created By

Mansi Tyagi, 2/8/2024, 11:25 AM

Last Modified By

Mansi Tyagi, 2/8/2024, 7:08 PM

Apps

Object Settings

Permissions to access objects and fields, and settings such as tab availability

App Permissions

Permissions to perform app-specific actions, such as "Manage Call Centers"

Flow Access

Permissions to execute Flows

External Credential Principal Access

Permissions to authenticate with external credential principal mappings

Custom Metadata Types

Permissions to access custom metadata types

Custom Setting Definitions

Permissions to access custom settings

Settings that apply to Salesforce apps, such as Sales, and custom apps built on the Lightning Platform

[Learn More](#)

Now click edit and select principal created for FedEx in above steps(step # 3)

SETUP

Permission Sets

Permission Set

B2BBuyer

Find Settings...

Clone

Edit Properties

Manage Assignments

View Summary (Beta)

View Tutorial/Help for this Page

Permission Set Overview

External Credential Principal Access

External Credential Principal Access

Save

Close

Available External Credential Principals

FedExAuth - FedExCredentials

UPSAuth - UPSCredentials

Add

Remove

Enabled External Credential Principals

--None--

SETUP

Permission Sets

Permission Set

B2BBuyer

Find Settings...

Clone

Edit Properties

Manage Assignments

View Summary (Beta)

View Tutorial/Help for this Page

Permission Set Overview

External Credential Principal Access

Items per page

50 Items

View

All

External Credential Principal Access

Edit

Page 1 of 1

External Credential Principal Name

Installed Package

FedExAuth - FedExCredentials

More details can be found here - https://help.salesforce.com/s/articleView?id=sf.nc_enable_ext_cred_principal.htm&type=5

6

Record - FedEx Shipping Provider(Shipping Provider)

Field Label	Field Value
Context Name	<Webstore name> e.g. R1Training
Context Locale	<locale> e.g. en_US

Impl Class	FedExShippingProvider
Max Weight Threshold	150
Min Weight Threshold	5
Service	FedEx Shipping Service
Account Number	<account number > provider by shipping provider with other credentials
Shipping Options	<pre> 1 { 2 "20": "FedEx Express Saver@", 3 "39": "FedEx First Overnight Freight", 4 "49": "FedEx 2Day@AM", 5 "70": "FedEx 1Day Freight", 6 "80": "FedEx 2Day@Freight", 7 "83": "FedEx 3Day@Freight", 8 "90": "FedEx Home Delivery@", 9 "92": "FedEx Ground@", 10 "111": "FedEx Freight@", 11 "112": "FedEx Freight@Priority", 12 "113": "FedEx Freight@Economy", 13 "01": "FedEx Priority Overnight@", 14 "03": "FedEx 2Day@ ", 15 "05": "FedEx Standard Overnight@", 16 "06": "FedEx First Overnight@", 17 "E1": "Extra Hours Priority Overnight", 18 "E5": "Extra Hours Standard Overnight", 19 "E6": "Extra Hours First Overnight" 20 } </pre>
Request	<pre> 1 { 2 "accountNumber": { 3 "value": "XXXXX7364" 4 }, 5 "requestedShipment": { 6 "shipper": { 7 "address": { 8 "postalCode": 65247, 9 "countryCode": "US" 10 } 11 }, 12 "recipient": { 13 "address": { 14 "postalCode": 75063, 15 "countryCode": "US" 16 } 17 }, 18 "pickupType": "DROPOFF_AT_FEDEX_LOCATION", 19 "rateRequestType": [20 "LIST" 21], 22 "requestedPackageLineItems": [23 { 24 "weight": { 25 "units": "LB", 26 "value": 0 27 } 28 } 29] </pre>

```
30    }  
31    }
```

Record - FedEx Shipping Service(HTTP Service)

Field Label	Field Value
End Point	/api/rating/v1/Shop
HTTP Method	GET POST PUT DELETE PATCH HEAD TRACE
Impl Class	<i>This is optional so could be left blank</i>
Named Credentials	FedEx_Shipping
Service Mode	Live Mocked
Timeout	5000 (ms)
Success Codes	200,201
Error Codes	400,403,404,500,503
Auth Codes	401
Mocked Response	<pre>1 { 2 "transactionId": "APIF_SV_RATC_TxID7c3f6154-cac4-49e8-bdc5-313581bdced1", 3 "customerTransactionId": "customer test", 4 "output": { 5 "alerts": [6 { 7 "code": "VIRTUAL.RESPONSE", 8 "message": "This is a Virtual Response.", 9 "alertType": "NOTE" 10 }, 11 { 12 "code": "ORIGIN.STATEORPROVINCECODE.CHANGED", 13 "message": "The origin state/province code has been changed.", 14 "alertType": "NOTE" 15 }, 16 { 17 "code": "DESTINATION.STATEORPROVINCECODE.CHANGED", 18 "message": "The destination state/province code has been changed.", 19 "alertType": "NOTE" 20 } 21], 22 "rateReplyDetails": [23 { 24 "serviceType": "PRIORITY_OVERNIGHT",</pre>


```

25     "serviceName": "FedEx Priority Overnight@",
26     "packagingType": "YOUR_PACKAGING",
27     "ratedShipmentDetails": [
28         {
29             "rateType": "LIST",
30             "ratedWeightMethod": "ACTUAL",
31             "totalDiscounts": 0.0,
32             "totalBaseCharge": 53.23,
33             "totalNetCharge": 66.25,
34             "totalNetFedExCharge": 66.25,
35             "shipmentRateDetail": {
36                 "rateZone": "02",
37                 "dimDivisor": 0,
38                 "fuelSurchargePercent": 14.75,
39                 "totalSurcharges": 13.02,
40                 "totalFreightDiscount": 0.0,
41                 "surCharges": [
42                     {
43                         "type": "FUEL",
44                         "description": "Fuel Surcharge",
45                         "amount": 8.52
46                     },
47                     {
48                         "type": "DELIVERY_AREA",
49                         "description": "DAS Extended Comm",
50                         "amount": 4.5
51                     }
52                 ],
53                 "pricingCode": "PACKAGE",
54                 "totalBillingWeight": {
55                     "units": "LB",
56                     "value": 10.0
57                 },
58                 "currency": "USD",
59                 "rateScale": "1486"
60             },
61             "ratedPackages": [
62                 {
63                     "groupNumber": 0,
64                     "effectiveNetDiscount": 0.0,
65                     "packageRateDetail": {
66                         "rateType": "PAYOR_ACCOUNT_PACKAGE",
67                         "ratedWeightMethod": "ACTUAL",
68                         "baseCharge": 53.23,
69                         "netFreight": 53.23,
70                         "totalSurcharges": 13.02,
71                         "netFedExCharge": 66.25,
72                         "totalTaxes": 0.0,
73                         "netCharge": 66.25,
74                         "totalRebates": 0.0,
75                         "billingWeight": {
76                             "units": "LB",
77                             "value": 10.0
78                         },
79                         "totalFreightDiscounts": 0.0,
80                         "surcharges": [
81                             {
82                                 "type": "FUEL",
83                                 "description": "Fuel Surcharge",
84                                 "amount": 8.52

```

```

85         },
86         {
87             "type": "DELIVERY_AREA",
88             "description": "DAS Extended Comm",
89             "amount": 4.5
90         }
91     ],
92     "currency": "USD"
93 }
94 }
95 ],
96 "currency": "USD"
97 }
98 ],
99 "operationalDetail": {
100     "ineligibleForMoneyBackGuarantee": false,
101     "astraDescription": "P1",
102     "airportId": "TLH",
103     "serviceCode": "01"
104 },
105 "signatureOptionType": "SERVICE_DEFAULT",
106 "serviceDescription": {
107     "serviceId": "EP1000000002",
108     "serviceType": "PRIORITY_OVERNIGHT",
109     "code": "01",
110     "names": [
111         {
112             "type": "long",
113             "encoding": "utf-8",
114             "value": "FedEx Priority Overnight@"
115         },
116         {
117             "type": "long",
118             "encoding": "ascii",
119             "value": "FedEx Priority Overnight"
120         },
121         {
122             "type": "medium",
123             "encoding": "utf-8",
124             "value": "FedEx Priority Overnight@"
125         },
126         {
127             "type": "medium",
128             "encoding": "ascii",
129             "value": "FedEx Priority Overnight"
130         },
131         {
132             "type": "short",
133             "encoding": "utf-8",
134             "value": "P-1"
135         },
136         {
137             "type": "short",
138             "encoding": "ascii",
139             "value": "P-1"
140         },
141         {
142             "type": "abbrv",
143             "encoding": "ascii",
144             "value": "P0"

```

```

145         }
146     ],
147     "serviceCategory": "parcel",
148     "description": "Priority Overnight",
149     "astraDescription": "P1"
150 }
151 },
152 {
153     "serviceType": "FEDEX_2_DAY",
154     "serviceName": "FedEx 2Day®",
155     "packagingType": "YOUR_PACKAGING",
156     "ratedShipmentDetails": [
157         {
158             "rateType": "LIST",
159             "ratedWeightMethod": "ACTUAL",
160             "totalDiscounts": 0.0,
161             "totalBaseCharge": 30.4,
162             "totalNetCharge": 40.05,
163             "totalNetFedExCharge": 40.05,
164             "shipmentRateDetail": {
165                 "rateZone": "02",
166                 "dimDivisor": 0,
167                 "fuelSurchargePercent": 14.75,
168                 "totalSurcharges": 9.65,
169                 "totalFreightDiscount": 0.0,
170                 "surCharges": [
171                     {
172                         "type": "FUEL",
173                         "description": "Fuel Surcharge",
174                         "amount": 5.15
175                     },
176                     {
177                         "type": "DELIVERY_AREA",
178                         "description": "DAS Extended Comm",
179                         "amount": 4.5
180                     }
181                 ],
182                 "pricingCode": "PACKAGE",
183                 "totalBillingWeight": {
184                     "units": "LB",
185                     "value": 10.0
186                 },
187                 "currency": "USD",
188                 "rateScale": "5980"
189             },
190             "ratedPackages": [
191                 {
192                     "groupNumber": 0,
193                     "effectiveNetDiscount": 0.0,
194                     "packageRateDetail": {
195                         "rateType": "PAYOR_ACCOUNT_PACKAGE",
196                         "ratedWeightMethod": "ACTUAL",
197                         "baseCharge": 30.4,
198                         "netFreight": 30.4,
199                         "totalSurcharges": 9.65,
200                         "netFedExCharge": 40.05,
201                         "totalTaxes": 0.0,
202                         "netCharge": 40.05,
203                         "totalRebates": 0.0,
204                         "billingWeight": {

```

```

205         "units": "LB",
206         "value": 10.0
207     },
208     "totalFreightDiscounts": 0.0,
209     "surcharges": [
210         {
211             "type": "FUEL",
212             "description": "Fuel Surcharge",
213             "amount": 5.15
214         },
215         {
216             "type": "DELIVERY_AREA",
217             "description": "DAS Extended Comm",
218             "amount": 4.5
219         }
220     ],
221     "currency": "USD"
222 }
223 }
224 ],
225 "currency": "USD"
226 }
227 ],
228 "operationalDetail": {
229     "ineligibleForMoneyBackGuarantee": false,
230     "astraDescription": "E2",
231     "airportId": "TLH",
232     "serviceCode": "03"
233 },
234 "signatureOptionType": "SERVICE_DEFAULT",
235 "serviceDescription": {
236     "serviceId": "EP1000000003",
237     "serviceType": "FEDEX_2_DAY",
238     "code": "03",
239     "names": [
240         {
241             "type": "long",
242             "encoding": "utf-8",
243             "value": "FedEx 2Day®"
244         },
245         {
246             "type": "long",
247             "encoding": "ascii",
248             "value": "FedEx 2Day"
249         },
250         {
251             "type": "medium",
252             "encoding": "utf-8",
253             "value": "FedEx 2Day®"
254         },
255         {
256             "type": "medium",
257             "encoding": "ascii",
258             "value": "FedEx 2Day"
259         },
260         {
261             "type": "short",
262             "encoding": "utf-8",
263             "value": "P-2"
264         }
265     ]
266 }

```



```

325         "ratedWeightMethod": "ACTUAL",
326         "baseCharge": 13.51,
327         "netFreight": 13.51,
328         "totalSurcharges": 7.02,
329         "netFedExCharge": 20.53,
330         "totalTaxes": 0.0,
331         "netCharge": 20.53,
332         "totalRebates": 0.0,
333         "billingWeight": {
334             "units": "LB",
335             "value": 10.0
336         },
337         "totalFreightDiscounts": 0.0,
338         "surcharges": [
339             {
340                 "type": "FUEL",
341                 "description": "Fuel Surcharge",
342                 "level": "PACKAGE",
343                 "amount": 2.52
344             },
345             {
346                 "type": "DELIVERY_AREA",
347                 "description": "DAS Extended Comm",
348                 "level": "PACKAGE",
349                 "amount": 4.5
350             }
351         ],
352         "currency": "USD"
353     }
354 }
355 ],
356 "currency": "USD"
357 }
358 ],
359 "operationalDetail": {
360     "ineligibleForMoneyBackGuarantee": false,
361     "astraDescription": "FXG",
362     "airportId": "TLH",
363     "serviceCode": "92"
364 },
365 "signatureOptionType": "SERVICE_DEFAULT",
366 "serviceDescription": {
367     "serviceId": "EP1000000134",
368     "serviceType": "FEDEX_GROUND",
369     "code": "92",
370     "names": [
371         {
372             "type": "long",
373             "encoding": "utf-8",
374             "value": "FedEx Ground®"
375         },
376         {
377             "type": "long",
378             "encoding": "ascii",
379             "value": "FedEx Ground"
380         },
381         {
382             "type": "medium",
383             "encoding": "utf-8",
384             "value": "Ground®"

```

```
385         },
386         {
387             "type": "medium",
388             "encoding": "ascii",
389             "value": "Ground"
390         },
391         {
392             "type": "short",
393             "encoding": "utf-8",
394             "value": "FG"
395         },
396         {
397             "type": "short",
398             "encoding": "ascii",
399             "value": "FG"
400         },
401         {
402             "type": "abbrv",
403             "encoding": "ascii",
404             "value": "SG"
405         }
406     ],
407     "description": "FedEx Ground",
408     "astraDescription": "FXG"
409 }
410 }
411 ],
412 "quoteDate": "2024-02-26",
413 "encoded": false
414 }
415 }
```

Source Code

Class	Details	Code
FedExShippingProvider	<p>This class extends ShippingProvider class & implements following methods -</p> <p>prepareRequestBody</p> <p>getShippingOptionsFromResponse</p>	<pre>1 public with sharing class FedExShippingProvider extends ShippingProvider { 2 public FedExShippingProvider() { 3 } 4 public override void prepareRequestBody(5 ShippingProviderRequest shippingRequest, 6 Map<String, String> callOutRequest 7) { 8 FedExShippingRateRequest requestObject = FedExShippingRateRequest.newInstance(9 shippingRequest.shippingMetaData.requestJSON 10); 11 if (requestObject != null) { 12 requestObject.requestedShipment.recipient.address = setShippingRequestAddress(13 shippingRequest 14); 15 FedExShippingRateRequest.RequestedPackageLineItems packageLineItems = 16 new FedExShippingRateRequest.RequestedPackageLineItems(17 0 18); 19 FedExShippingRateRequest.Weight weight = new FedExShippingRateRequest.Weight(20 shippingRequest.packageWeight</pre>

```

21     String strRatingRequestBody = JSON.serialize(requestObject);
22     callOutRequest.put(Constants.SERVICE_REQUEST_BODY, strRatingRequestBody);
23 }
24 }
25
26 public virtual FedExShippingRateRequest.Address setShipToAddress(
27     ShippingProviderRequest shippingRequest) {
28 } {
29     FedExShippingRateRequest.Address address = new FedExShippingRateRequest.Address();
30     address.postalCode = shippingRequest.postalCode;
31     address.countryCode = shippingRequest.country;
32     return address;
33 }
34
35 public override Map<String, ShippingProviderResponse> getShippingMethods(
36     List<String> responseList,
37     ShippingProviderRequest shippingRequest) {
38 } {
39     Map<String, Object> shippingMethods = new Map<String, Object>();
40     Map<String, ShippingProviderResponse> shippingMethodsWithRates = new Map<String, ShippingProviderResponse>();
41     for (String response : responseList) {
42         FedExShippingRateResponse responseObj = JSON.parse(response);
43         shippingMethods.put(responseObj.getServiceCode(), responseObj);
44     }
45     return shippingMethodsWithRates;
46 }
47
48 public virtual Map<String, ShippingProviderResponse> parseFedExShippingRateResponse(
49     List<String> lstFedExResponseBody) {
50 } {
51     Map<String, ShippingProviderResponse> shippingMethodsWithRates = new Map<String, ShippingProviderResponse>();
52     for (String strFedExResponseBody : lstFedExResponseBody) {
53         FedExShippingRateResponse fedexShippingRateResponseObj = JSON.parse(strFedExResponseBody);
54         shippingMethodsWithRates.put(fedexShippingRateResponseObj.getServiceCode(), fedexShippingRateResponseObj);
55     }
56     return shippingMethodsWithRates;
57 }
58
59 public virtual Map<String, ShippingProviderResponse> getShippingMethodsWithRates(
60     List<String> responseList,
61     ShippingProviderRequest shippingRequest) {
62 } {
63     Map<String, Object> shippingMethods = new Map<String, Object>();
64     Map<String, ShippingProviderResponse> shippingMethodsWithRates = new Map<String, ShippingProviderResponse>();
65     for (String response : responseList) {
66         FedExShippingRateResponse responseObj = JSON.parse(response);
67         shippingMethods.put(responseObj.getServiceCode(), responseObj);
68     }
69     return shippingMethodsWithRates;
70 }
71
72 public virtual Map<String, ShippingProviderResponse> getShippingMethodsWithRatesAndCost(
73     List<String> responseList,
74     ShippingProviderRequest shippingRequest) {
75 } {
76     Map<String, Object> shippingMethods = new Map<String, Object>();
77     Map<String, ShippingProviderResponse> shippingMethodsWithRates = new Map<String, ShippingProviderResponse>();
78     Map<String, ShippingProviderResponse> shippingMethodsWithRatesAndCost = new Map<String, ShippingProviderResponse>();
79     for (String response : responseList) {
80         FedExShippingRateResponse responseObj = JSON.parse(response);
81         shippingMethods.put(responseObj.getServiceCode(), responseObj);
82         shippingMethodsWithRates.put(responseObj.getServiceCode(), responseObj);
83         shippingMethodsWithRatesAndCost.put(responseObj.getServiceCode(), responseObj);
84     }
85     return shippingMethodsWithRatesAndCost;
86 }

```


		<pre> 81 } 82 } 83 } 84 } 85 return shippingMethodsWithRate; 86 } 87 } 88 89 </pre>
FedExShippingRateRequest	This class is a DTO used to prepare FedEx Shipping rate request which is posted to FedEx to get shipping details	<pre> 1 public class FedExShippingRateRequest { 2 public AccountNumber accountNumber; 3 public RequestedShipment requestedShipment; 4 5 public class Address { 6 public String postalCode; 7 public String countryCode; 8 } 9 10 public class Shipper { 11 public Address address; 12 } 13 14 public class RequestedPackageLineItems { 15 public Weight weight; 16 } 17 18 public class RequestedShipment { 19 public Shipper shipper; 20 public Shipper recipient; 21 public String pickupType; 22 public List<String> rateRequestType; 23 public List<RequestedPackageLineItems> requestedPackageLine 24 } 25 26 public class Weight { 27 public String units; 28 public Decimal value; 29 } 30 31 public class AccountNumber { 32 public String value; 33 } 34 35 public static FedExShippingRateRequest parse(String json) { 36 return (FedExShippingRateRequest) System.JSON.deserialize(37 json, 38 FedExShippingRateRequest.class 39); 40 } 41 } 42 43 </pre>
FedExShippingRateResponse	This class is a DTO used to parse shipping rate response returned by FedEx	<pre> 1 public class FedExShippingRateResponse { 2 public class ShipmentRateDetail_Z { 3 public String rateZone { get; set; } 4 public Integer dimDivisor { get; set; } </pre>

```

5     public Double fuelSurchargePercent { get; set; }
6     public Double totalSurcharges { get; set; }
7     public Double totalFreightDiscount { get; set; }
8     public List<SurCharges_Z> surCharges { get; set; }
9     public TotalBillingWeight totalBillingWeight { get; set; }
10    public String currency_Z { get; set; } // in json: currenc
11
12    public ShipmentRateDetail_Z(JSONParser parser) {
13        while (parser.nextToken() != System.JSONToken.END_OBJECT
14            if (parser.getCurrentToken() == System.JSONToken.FIELD
15                String text = parser.getText();
16                if (parser.nextToken() != System.JSONToken.VALUE_NUL
17                    if (text == 'rateZone') {
18                        rateZone = parser.getText();
19                    } else if (text == 'dimDivisor') {
20                        dimDivisor = parser.getIntegerValue();
21                    } else if (text == 'fuelSurchargePercent') {
22                        fuelSurchargePercent = parser.getDoubleValue();
23                    } else if (text == 'totalSurcharges') {
24                        totalSurcharges = parser.getDoubleValue();
25                    } else if (text == 'totalFreightDiscount') {
26                        totalFreightDiscount = parser.getDoubleValue();
27                    } else if (text == 'surCharges') {
28                        surCharges = arrayOfSurCharges_Z(parser);
29                    } else if (text == 'totalBillingWeight') {
30                        totalBillingWeight = new TotalBillingWeight(pars
31                    } else if (text == 'currency') {
32                        currency_Z = parser.getText();
33                    } else {
34                        System.debug(
35                            LoggingLevel.WARN,
36                            'ShipmentRateDetail_Z consuming unrecognized p
37                        );
38                        consumeObject(parser);
39                    }
40                }
41            }
42        }
43    }
44 }
45
46 public class RatedPackages_Z {
47     public Integer groupNumber { get; set; }
48     public Double effectiveNetDiscount { get; set; }
49     public PackageRateDetail_Z packageRateDetail { get; set; }
50
51     public RatedPackages_Z(JSONParser parser) {
52         while (parser.nextToken() != System.JSONToken.END_OBJECT
53             if (parser.getCurrentToken() == System.JSONToken.FIELD
54                 String text = parser.getText();
55                 if (parser.nextToken() != System.JSONToken.VALUE_NUL
56                     if (text == 'groupNumber') {
57                         groupNumber = parser.getIntegerValue();
58                     } else if (text == 'effectiveNetDiscount') {
59                         effectiveNetDiscount = parser.getDoubleValue();
60                     } else if (text == 'packageRateDetail') {
61                         packageRateDetail = new PackageRateDetail_Z(pars
62                     } else {
63                         System.debug(
64                             LoggingLevel.WARN,

```



```

125         description = parser.getText();
126     } else if (text == 'astraDescription') {
127         astraDescription = parser.getText();
128     } else {
129         System.debug(
130             LoggingLevel.WARN,
131             'ServiceDescription_Z consuming unrecognized p
132         );
133         consumeObject(parser);
134     }
135 }
136 }
137 }
138 }
139 }
140
141 public class ServiceDescription {
142     public String serviceId { get; set; }
143     public String serviceType { get; set; }
144     public String code { get; set; }
145     public List<Names> names { get; set; }
146     public String serviceCategory { get; set; }
147     public String description { get; set; }
148     public String astraDescription { get; set; }
149
150     public ServiceDescription(JSONParser parser) {
151         while (parser.nextToken() != System.JSONToken.END_OBJECT
152             if (parser.getCurrentToken() == System.JSONToken.FIELD
153                 String text = parser.getText();
154                 if (parser.nextToken() != System.JSONToken.VALUE_NUL
155                     if (text == 'serviceId') {
156                         serviceId = parser.getText();
157                     } else if (text == 'serviceType') {
158                         serviceType = parser.getText();
159                     } else if (text == 'code') {
160                         code = parser.getText();
161                     } else if (text == 'names') {
162                         names = arrayOfNames(parser);
163                     } else if (text == 'serviceCategory') {
164                         serviceCategory = parser.getText();
165                     } else if (text == 'description') {
166                         description = parser.getText();
167                     } else if (text == 'astraDescription') {
168                         astraDescription = parser.getText();
169                     } else {
170                         System.debug(
171                             LoggingLevel.WARN,
172                             'ServiceDescription consuming unrecognized pro
173                         );
174                         consumeObject(parser);
175                     }
176                 }
177             }
178         }
179     }
180 }
181
182 public class PackageRateDetail_Z {
183     public String rateType { get; set; }
184     public String ratedWeightMethod { get; set; }

```

```

185     public Double baseCharge { get; set; }
186     public Double netFreight { get; set; }
187     public Double totalSurcharges { get; set; }
188     public Double netFedExCharge { get; set; }
189     public Double totalTaxes { get; set; }
190     public Double netCharge { get; set; }
191     public Double totalRebates { get; set; }
192     public TotalBillingWeight billingWeight { get; set; }
193     public Double totalFreightDiscounts { get; set; }
194     public List<SurCharges_Z> surcharges { get; set; }
195     public String currency_Z { get; set; } // in json: currenc
196
197     public PackageRateDetail_Z(JSONParser parser) {
198         while (parser.nextToken() != System.JSONToken.END_OBJECT
199             if (parser.getCurrentToken() == System.JSONToken.FIELD
200                 String text = parser.getText();
201                 if (parser.nextToken() != System.JSONToken.VALUE_NUL
202                     if (text == 'rateType') {
203                         rateType = parser.getText();
204                     } else if (text == 'ratedWeightMethod') {
205                         ratedWeightMethod = parser.getText();
206                     } else if (text == 'baseCharge') {
207                         baseCharge = parser.getDoubleValue();
208                     } else if (text == 'netFreight') {
209                         netFreight = parser.getDoubleValue();
210                     } else if (text == 'totalSurcharges') {
211                         totalSurcharges = parser.getDoubleValue();
212                     } else if (text == 'netFedExCharge') {
213                         netFedExCharge = parser.getDoubleValue();
214                     } else if (text == 'totalTaxes') {
215                         totalTaxes = parser.getDoubleValue();
216                     } else if (text == 'netCharge') {
217                         netCharge = parser.getDoubleValue();
218                     } else if (text == 'totalRebates') {
219                         totalRebates = parser.getDoubleValue();
220                     } else if (text == 'billingWeight') {
221                         billingWeight = new TotalBillingWeight(parser);
222                     } else if (text == 'totalFreightDiscounts') {
223                         totalFreightDiscounts = parser.getDoubleValue();
224                     } else if (text == 'surcharges') {
225                         surcharges = arrayOfSurCharges_Z(parser);
226                     } else if (text == 'currency') {
227                         currency_Z = parser.getText();
228                     } else {
229                         System.debug(
230                             LoggingLevel.WARN,
231                             'PackageRateDetail_Z consuming unrecognized pr
232                         );
233                         consumeObject(parser);
234                     }
235                 }
236             }
237         }
238     }
239 }
240
241 public class OperationalDetail {
242     public Boolean ineligibleForMoneyBackGuarantee { get; set; }
243     public String astraDescription { get; set; }
244     public String airportId { get; set; }

```

```

245     public String serviceCode { get; set; }
246
247     public OperationalDetail(JSONParser parser) {
248         while (parser.nextToken() != System.JSONToken.END_OBJECT
249             if (parser.getCurrentToken() == System.JSONToken.FIELD
250                 String text = parser.getText();
251                 if (parser.nextToken() != System.JSONToken.VALUE_NULL
252                     if (text == 'ineligibleForMoneyBackGuarantee') {
253                         ineligibleForMoneyBackGuarantee = parser.getBool
254                     } else if (text == 'astraDescription') {
255                         astraDescription = parser.getText();
256                     } else if (text == 'airportId') {
257                         airportId = parser.getText();
258                     } else if (text == 'serviceCode') {
259                         serviceCode = parser.getText();
260                     } else {
261                         System.debug(
262                             LoggingLevel.WARN,
263                             'OperationalDetail consuming unrecognized prop
264                         );
265                         consumeObject(parser);
266                     }
267                 }
268             }
269         }
270     }
271 }
272
273 public class RatedPackages {
274     public Integer groupNumber { get; set; }
275     public Double effectiveNetDiscount { get; set; }
276     public PackageRateDetail packageRateDetail { get; set; }
277
278     public RatedPackages(JSONParser parser) {
279         while (parser.nextToken() != System.JSONToken.END_OBJECT
280             if (parser.getCurrentToken() == System.JSONToken.FIELD
281                 String text = parser.getText();
282                 if (parser.nextToken() != System.JSONToken.VALUE_NULL
283                     if (text == 'groupNumber') {
284                         groupNumber = parser.getIntegerValue();
285                     } else if (text == 'effectiveNetDiscount') {
286                         effectiveNetDiscount = parser.getDoubleValue();
287                     } else if (text == 'packageRateDetail') {
288                         packageRateDetail = new PackageRateDetail(parser
289                     } else {
290                         System.debug(
291                             LoggingLevel.WARN,
292                             'RatedPackages consuming unrecognized property
293                         );
294                         consumeObject(parser);
295                     }
296                 }
297             }
298         }
299     }
300 }
301
302 public class RatedShipmentDetails {
303     public String rateType { get; set; }
304     public String ratedWeightMethod { get; set; }

```

```

305     public Double totalDiscounts { get; set; }
306     public Double totalBaseCharge { get; set; }
307     public Double totalNetCharge { get; set; }
308     public Double totalNetFedExCharge { get; set; }
309     public ShipmentRateDetail shipmentRateDetail { get; set; }
310     public List<RatedPackages> ratedPackages { get; set; }
311     public String currency_Z { get; set; } // in json: currenc
312
313     public RatedShipmentDetails(JSONParser parser) {
314         while (parser.nextToken() != System.JSONToken.END_OBJECT
315             if (parser.getCurrentToken() == System.JSONToken.FIELD
316                 String text = parser.getText();
317                 if (parser.nextToken() != System.JSONToken.VALUE_NUL
318                     if (text == 'rateType') {
319                         rateType = parser.getText();
320                     } else if (text == 'ratedWeightMethod') {
321                         ratedWeightMethod = parser.getText();
322                     } else if (text == 'totalDiscounts') {
323                         totalDiscounts = parser.getDoubleValue();
324                     } else if (text == 'totalBaseCharge') {
325                         totalBaseCharge = parser.getDoubleValue();
326                     } else if (text == 'totalNetCharge') {
327                         totalNetCharge = parser.getDoubleValue();
328                     } else if (text == 'totalNetFedExCharge') {
329                         totalNetFedExCharge = parser.getDoubleValue();
330                     } else if (text == 'shipmentRateDetail') {
331                         shipmentRateDetail = new ShipmentRateDetail(pars
332                     } else if (text == 'ratedPackages') {
333                         ratedPackages = arrayOfRatedPackages(parser);
334                     } else if (text == 'currency') {
335                         currency_Z = parser.getText();
336                     } else {
337                         System.debug(
338                             LoggingLevel.WARN,
339                             'RatedShipmentDetails consuming unrecognized p
340                         );
341                         consumeObject(parser);
342                     }
343                 }
344             }
345         }
346     }
347 }
348
349 public class RatedShipmentDetails_Z {
350     public String rateType { get; set; }
351     public String ratedWeightMethod { get; set; }
352     public Double totalDiscounts { get; set; }
353     public Double totalBaseCharge { get; set; }
354     public Double totalNetCharge { get; set; }
355     public Double totalNetFedExCharge { get; set; }
356     public ShipmentRateDetail_Z shipmentRateDetail { get; set; }
357     public List<RatedPackages_Z> ratedPackages { get; set; }
358     public String currency_Z { get; set; } // in json: currenc
359
360     public RatedShipmentDetails_Z(JSONParser parser) {
361         while (parser.nextToken() != System.JSONToken.END_OBJECT
362             if (parser.getCurrentToken() == System.JSONToken.FIELD
363                 String text = parser.getText();
364                 if (parser.nextToken() != System.JSONToken.VALUE_NUL

```

```

365         if (text == 'rateType') {
366             rateType = parser.getText();
367         } else if (text == 'ratedWeightMethod') {
368             ratedWeightMethod = parser.getText();
369         } else if (text == 'totalDiscounts') {
370             totalDiscounts = parser.getDoubleValue();
371         } else if (text == 'totalBaseCharge') {
372             totalBaseCharge = parser.getDoubleValue();
373         } else if (text == 'totalNetCharge') {
374             totalNetCharge = parser.getDoubleValue();
375         } else if (text == 'totalNetFedExCharge') {
376             totalNetFedExCharge = parser.getDoubleValue();
377         } else if (text == 'shipmentRateDetail') {
378             shipmentRateDetail = new ShipmentRateDetail_Z(pa
379         } else if (text == 'ratedPackages') {
380             ratedPackages = arrayOfRatedPackages_Z(parser);
381         } else if (text == 'currency') {
382             currency_Z = parser.getText();
383         } else {
384             System.debug(
385                 LoggingLevel.WARN,
386                 'RatedShipmentDetails_Z consuming unrecognized
387                 text
388             );
389             consumeObject(parser);
390         }
391     }
392 }
393 }
394 }
395 }
396
397 public class SurCharges {
398     public String type { get; set; }
399     public String description { get; set; }
400     public Double amount { get; set; }
401
402     public SurCharges(JSONParser parser) {
403         while (parser.nextToken() != System.JSONToken.END_OBJECT
404             if (parser.getCurrentToken() == System.JSONToken.FIELD
405                 String text = parser.getText();
406                 if (parser.nextToken() != System.JSONToken.VALUE_NUL
407                     if (text == 'type') {
408                         type = parser.getText();
409                     } else if (text == 'description') {
410                         description = parser.getText();
411                     } else if (text == 'amount') {
412                         amount = parser.getDoubleValue();
413                     } else {
414                         System.debug(
415                             LoggingLevel.WARN,
416                             'SurCharges consuming unrecognized property: '
417                         );
418                         consumeObject(parser);
419                     }
420                 }
421             }
422         }
423     }
424 }

```



```

485         pricingCode = parser.getText();
486     } else if (text == 'totalBillingWeight') {
487         totalBillingWeight = new TotalBillingWeight(pars
488     } else if (text == 'currency') {
489         currency_Z = parser.getText();
490     } else if (text == 'rateScale') {
491         rateScale = parser.getText();
492     } else {
493         System.debug(
494             LoggingLevel.WARN,
495             'ShipmentRateDetail consuming unrecognized pro
496         );
497         consumeObject(parser);
498     }
499 }
500 }
501 }
502 }
503 }
504
505 public class PackageRateDetail {
506     public String rateType { get; set; }
507     public String ratedWeightMethod { get; set; }
508     public Double baseCharge { get; set; }
509     public Double netFreight { get; set; }
510     public Double totalSurcharges { get; set; }
511     public Double netFedExCharge { get; set; }
512     public Double totalTaxes { get; set; }
513     public Double netCharge { get; set; }
514     public Double totalRebates { get; set; }
515     public TotalBillingWeight billingWeight { get; set; }
516     public Double totalFreightDiscounts { get; set; }
517     public List<SurCharges> surcharges { get; set; }
518     public String currency_Z { get; set; } // in json: currenc
519
520     public PackageRateDetail(JSONParser parser) {
521         while (parser.nextToken() != System.JSONToken.END_OBJECT
522             if (parser.getCurrentToken() == System.JSONToken.FIELD
523                 String text = parser.getText();
524                 if (parser.nextToken() != System.JSONToken.VALUE_NUL
525                     if (text == 'rateType') {
526                         rateType = parser.getText();
527                     } else if (text == 'ratedWeightMethod') {
528                         ratedWeightMethod = parser.getText();
529                     } else if (text == 'baseCharge') {
530                         baseCharge = parser.getDoubleValue();
531                     } else if (text == 'netFreight') {
532                         netFreight = parser.getDoubleValue();
533                     } else if (text == 'totalSurcharges') {
534                         totalSurcharges = parser.getDoubleValue();
535                     } else if (text == 'netFedExCharge') {
536                         netFedExCharge = parser.getDoubleValue();
537                     } else if (text == 'totalTaxes') {
538                         totalTaxes = parser.getDoubleValue();
539                     } else if (text == 'netCharge') {
540                         netCharge = parser.getDoubleValue();
541                     } else if (text == 'totalRebates') {
542                         totalRebates = parser.getDoubleValue();
543                     } else if (text == 'billingWeight') {
544                         billingWeight = new TotalBillingWeight(parser);

```

```

545         } else if (text == 'totalFreightDiscounts') {
546             totalFreightDiscounts = parser.getDoubleValue();
547         } else if (text == 'surcharges') {
548             surcharges = arrayOfSurCharges(parser);
549         } else if (text == 'currency') {
550             currency_Z = parser.getText();
551         } else {
552             System.debug(
553                 LoggingLevel.WARN,
554                 'PackageRateDetail consuming unrecognized prop
555             );
556             consumeObject(parser);
557         }
558     }
559 }
560 }
561 }
562 }
563
564 public class RateReplyDetails {
565     public String serviceType { get; set; }
566     public String serviceName { get; set; }
567     public String packagingType { get; set; }
568     public List<RatedShipmentDetails> ratedShipmentDetails { g
569     public OperationalDetail operationalDetail { get; set; }
570     public String signatureOptionType { get; set; }
571     public ServiceDescription serviceDescription { get; set; }
572
573     public RateReplyDetails(JSONParser parser) {
574         while (parser.nextToken() != System.JSONToken.END_OBJECT
575             if (parser.getCurrentToken() == System.JSONToken.FIELD
576                 String text = parser.getText();
577                 if (parser.nextToken() != System.JSONToken.VALUE_NUL
578                     if (text == 'serviceType') {
579                         serviceType = parser.getText();
580                     } else if (text == 'serviceName') {
581                         serviceName = parser.getText();
582                     } else if (text == 'packagingType') {
583                         packagingType = parser.getText();
584                     } else if (text == 'ratedShipmentDetails') {
585                         ratedShipmentDetails = arrayOfRatedShipmentDetail
586                     } else if (text == 'operationalDetail') {
587                         operationalDetail = new OperationalDetail(parser
588                     } else if (text == 'signatureOptionType') {
589                         signatureOptionType = parser.getText();
590                     } else if (text == 'serviceDescription') {
591                         serviceDescription = new ServiceDescription(pars
592                     } else {
593                         System.debug(
594                             LoggingLevel.WARN,
595                             'RateReplyDetails consuming unrecognized prop
596                         );
597                         consumeObject(parser);
598                     }
599                 }
600             }
601         }
602     }
603 }
604

```

```

605 public class TotalBillingWeight {
606     public String units { get; set; }
607     public Double value { get; set; }
608
609     public TotalBillingWeight(JSONParser parser) {
610         while (parser.nextToken() != System.JSONToken.END_OBJECT
611             if (parser.getCurrentToken() == System.JSONToken.FIELD
612                 String text = parser.getText();
613                 if (parser.nextToken() != System.JSONToken.VALUE_NUL
614                     if (text == 'units') {
615                         units = parser.getText();
616                     } else if (text == 'value') {
617                         value = parser.getDoubleValue();
618                     } else {
619                         System.debug(
620                             LoggingLevel.WARN,
621                             'TotalBillingWeight consuming unrecognized pro
622                         );
623                         consumeObject(parser);
624                     }
625                 }
626             }
627         }
628     }
629 }
630
631 public class Output {
632     public List<Alerts> alerts { get; set; }
633     public List<RateReplyDetails> rateReplyDetails { get; set; }
634     public String quoteDate { get; set; }
635     public Boolean encoded { get; set; }
636
637     public Output(JSONParser parser) {
638         while (parser.nextToken() != System.JSONToken.END_OBJECT
639             if (parser.getCurrentToken() == System.JSONToken.FIELD
640                 String text = parser.getText();
641                 if (parser.nextToken() != System.JSONToken.VALUE_NUL
642                     if (text == 'alerts') {
643                         alerts = arrayOfAlerts(parser);
644                     } else if (text == 'rateReplyDetails') {
645                         rateReplyDetails = arrayOfRateReplyDetails(parse
646                     } else if (text == 'quoteDate') {
647                         quoteDate = parser.getText();
648                     } else if (text == 'encoded') {
649                         encoded = parser.getBooleanValue();
650                     } else {
651                         System.debug(
652                             LoggingLevel.WARN,
653                             'Output consuming unrecognized property: ' + t
654                         );
655                         consumeObject(parser);
656                     }
657                 }
658             }
659         }
660     }
661 }
662
663 public class Alerts {
664     public String code { get; set; }

```

```

665     public String message { get; set; }
666     public String alertType { get; set; }
667
668     public Alerts(JSONParser parser) {
669         while (parser.nextToken() != System.JSONToken.END_OBJECT
670             if (parser.getCurrentToken() == System.JSONToken.FIELD
671                 String text = parser.getText();
672                 if (parser.nextToken() != System.JSONToken.VALUE_NUL
673                     if (text == 'code') {
674                         code = parser.getText();
675                     } else if (text == 'message') {
676                         message = parser.getText();
677                     } else if (text == 'alertType') {
678                         alertType = parser.getText();
679                     } else {
680                         System.debug(
681                             LoggingLevel.WARN,
682                             'Alerts consuming unrecognized property: ' + t
683                         );
684                         consumeObject(parser);
685                     }
686                 }
687             }
688         }
689     }
690 }
691
692 public class SurCharges_Z {
693     public String type { get; set; }
694     public String description { get; set; }
695     public String level { get; set; }
696     public Double amount { get; set; }
697
698     public SurCharges_Z(JSONParser parser) {
699         while (parser.nextToken() != System.JSONToken.END_OBJECT
700             if (parser.getCurrentToken() == System.JSONToken.FIELD
701                 String text = parser.getText();
702                 if (parser.nextToken() != System.JSONToken.VALUE_NUL
703                     if (text == 'type') {
704                         type = parser.getText();
705                     } else if (text == 'description') {
706                         description = parser.getText();
707                     } else if (text == 'level') {
708                         level = parser.getText();
709                     } else if (text == 'amount') {
710                         amount = parser.getDoubleValue();
711                     } else {
712                         System.debug(
713                             LoggingLevel.WARN,
714                             'SurCharges_Z consuming unrecognized property:
715                         );
716                         consumeObject(parser);
717                     }
718                 }
719             }
720         }
721     }
722 }
723
724 public static FedExShippingRateResponse parse(String json) {

```

```

725     System.JSONParser parser = System.JSON.createParser(json);
726     return new FedExShippingRateResponse(parser);
727 }
728
729 public static void consumeObject(System.JSONParser parser) {
730     Integer depth = 0;
731     do {
732         System.JSONToken curr = parser.getCurrentToken();
733         if (
734             curr == System.JSONToken.START_OBJECT ||
735             curr == System.JSONToken.START_ARRAY
736         ) {
737             depth++;
738         } else if (
739             curr == System.JSONToken.END_OBJECT ||
740             curr == System.JSONToken.END_ARRAY
741         ) {
742             depth--;
743         }
744     } while (depth > 0 && parser.nextToken() != null);
745 }
746
747 private static List<Alerts> arrayOfAlerts(System.JSONParser
748     List<Alerts> res = new List<Alerts>();
749     if (p.getCurrentToken() == null)
750         p.nextToken();
751     while (p.nextToken() != System.JSONToken.END_ARRAY) {
752         res.add(new Alerts(p));
753     }
754     return res;
755 }
756
757 private static List<RatedPackages_Z> arrayOfRatedPackages_Z(
758     System.JSONParser p
759 ) {
760     List<RatedPackages_Z> res = new List<RatedPackages_Z>();
761     if (p.getCurrentToken() == null)
762         p.nextToken();
763     while (p.nextToken() != System.JSONToken.END_ARRAY) {
764         res.add(new RatedPackages_Z(p));
765     }
766     return res;
767 }
768
769 private static List<SurCharges> arrayOfSurCharges(System.JSON
770     List<SurCharges> res = new List<SurCharges>();
771     if (p.getCurrentToken() == null)
772         p.nextToken();
773     while (p.nextToken() != System.JSONToken.END_ARRAY) {
774         res.add(new SurCharges(p));
775     }
776     return res;
777 }
778
779 private static List<RatedShipmentDetails_Z> arrayOfRatedShip
780     System.JSONParser p
781 ) {
782     List<RatedShipmentDetails_Z> res = new List<RatedShipmentD
783     if (p.getCurrentToken() == null)
784         p.nextToken();

```

```

785     while (p.nextToken() != System.JSONToken.END_ARRAY) {
786         res.add(new RatedShipmentDetails_Z(p));
787     }
788     return res;
789 }
790
791 private static List<Names> arrayOfNames(System.JSONParser p)
792     List<Names> res = new List<Names>();
793     if (p.getCurrentToken() == null)
794         p.nextToken();
795     while (p.nextToken() != System.JSONToken.END_ARRAY) {
796         res.add(new Names(p));
797     }
798     return res;
799 }
800
801 private static List<RatedShipmentDetails> arrayOfRatedShipme
802     System.JSONParser p
803 ) {
804     List<RatedShipmentDetails> res = new List<RatedShipmentDet
805     if (p.getCurrentToken() == null)
806         p.nextToken();
807     while (p.nextToken() != System.JSONToken.END_ARRAY) {
808         res.add(new RatedShipmentDetails(p));
809     }
810     return res;
811 }
812
813 private static List<RateReplyDetails> arrayOfRateReplyDetail
814     System.JSONParser p
815 ) {
816     List<RateReplyDetails> res = new List<RateReplyDetails>();
817     if (p.getCurrentToken() == null)
818         p.nextToken();
819     while (p.nextToken() != System.JSONToken.END_ARRAY) {
820         res.add(new RateReplyDetails(p));
821     }
822     return res;
823 }
824
825 private static List<SurCharges_Z> arrayOfSurCharges_Z(System
826     List<SurCharges_Z> res = new List<SurCharges_Z>();
827     if (p.getCurrentToken() == null)
828         p.nextToken();
829     while (p.nextToken() != System.JSONToken.END_ARRAY) {
830         res.add(new SurCharges_Z(p));
831     }
832     return res;
833 }
834
835 private static List<RatedPackages> arrayOfRatedPackages(Syst
836     List<RatedPackages> res = new List<RatedPackages>();
837     if (p.getCurrentToken() == null)
838         p.nextToken();
839     while (p.nextToken() != System.JSONToken.END_ARRAY) {
840         res.add(new RatedPackages(p));
841     }
842     return res;
843 }
844 }

```

