SF-LWR Extension Framework : FedEx Shipping Integration

Functional Overview

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

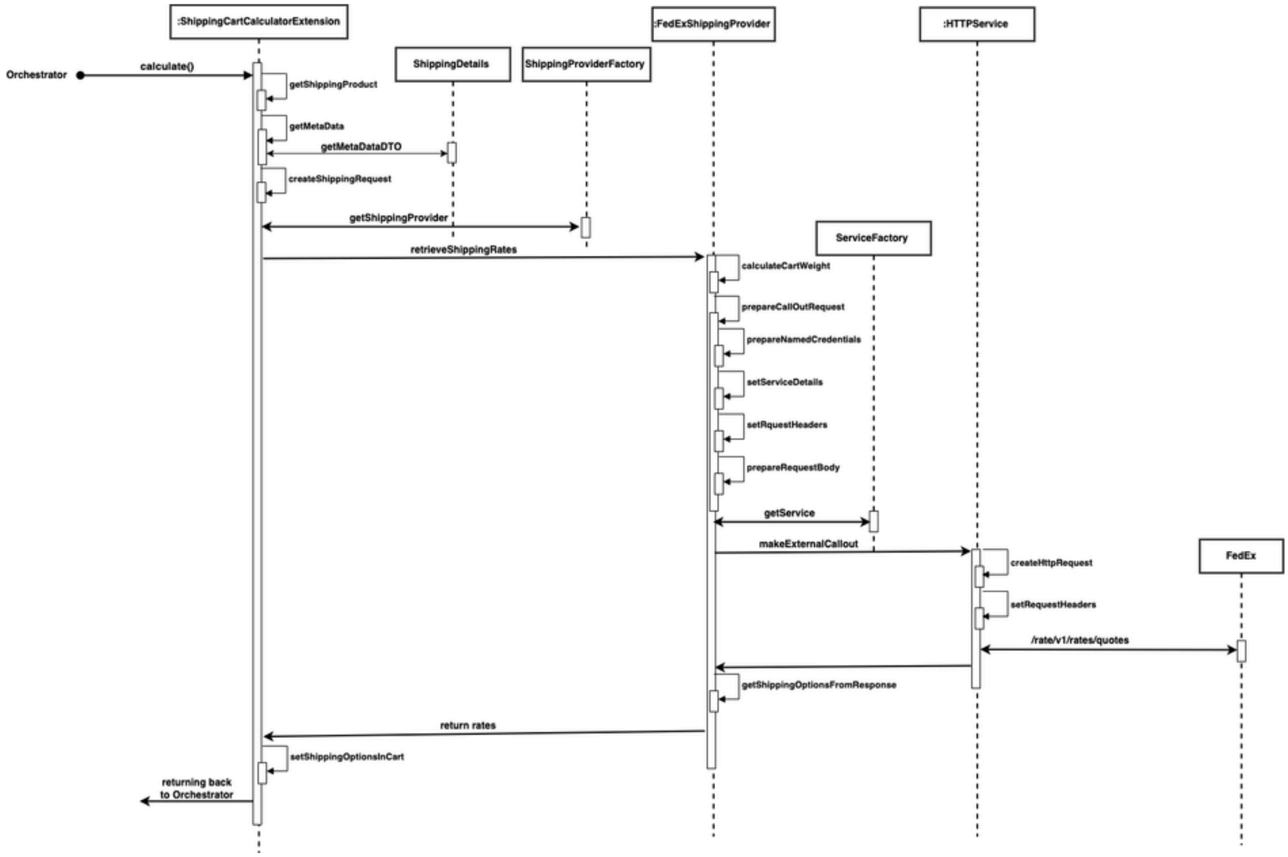
For common details you can refer to parent doc here.

Technical Details

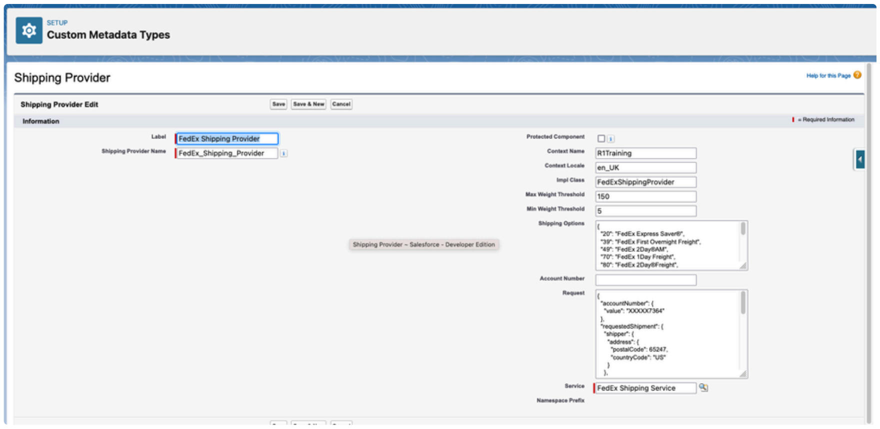
This integration implementation has two parts :

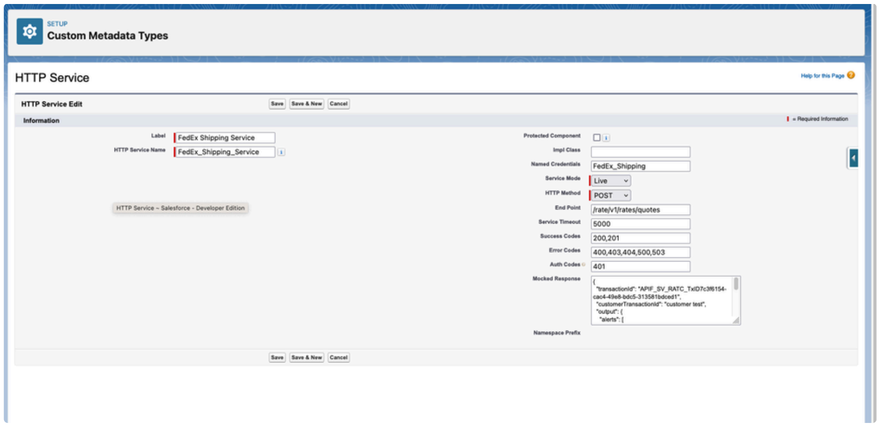
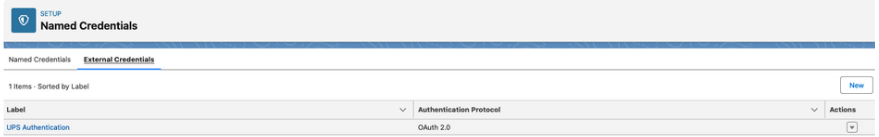
- **Configuration**
  - Custom Meta Data Records
    - FedEx Shipping Provider
    - FedEx Shipping Service
  - Named Credentials
    - FedEx Shipping
  - External Credentials
    - FedEx Authentication
    - FedExCredentials (Principal)
- **Source Code**
  - FedExShippingProvider
  - FedExShippingRateRequest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

Sequence Diagram



Configuration details

	Section	Task	Details
1	Salesforce Org	Shipping Provider - Create a record for FedEx	Create a record **FedEx Shipping Provider** for Custom Meta Data Type Shipping Provider  SETUP Custom Metadata Types Shipping Provider Shipping Provider Edit Information Label: FedEx Shipping Provider Shipping Provider Name: FedEx_Shipping_Provider Protected Component: □ Context Name: R1Training Context Locale: en_UK Impl Class: FedExShippingProvider Max Weight Threshold: 150 Min Weight Threshold: 5 Shipping Options: { "20": "FedEx Express Saver®", "39": "FedEx First Overnight Freight", "49": "FedEx 2Day®AM", "70": "FedEx 1Day Freight", "80": "FedEx 2Day®Freight", Account Number: Request: { "accountNumber": { "value": "XXXXX7364" }, "requestedShipment": { "shipper": { "address": { "postalCode": 65247, "countryCode": "US" } }, Service: FedEx Shipping Service Namespace Prefix: Shipping Provider ~ Salesforce - Developer Edition
2	Salesforce Org	HTTP Service - Create a record for FedEx	Create a record - FedEx Shipping Service

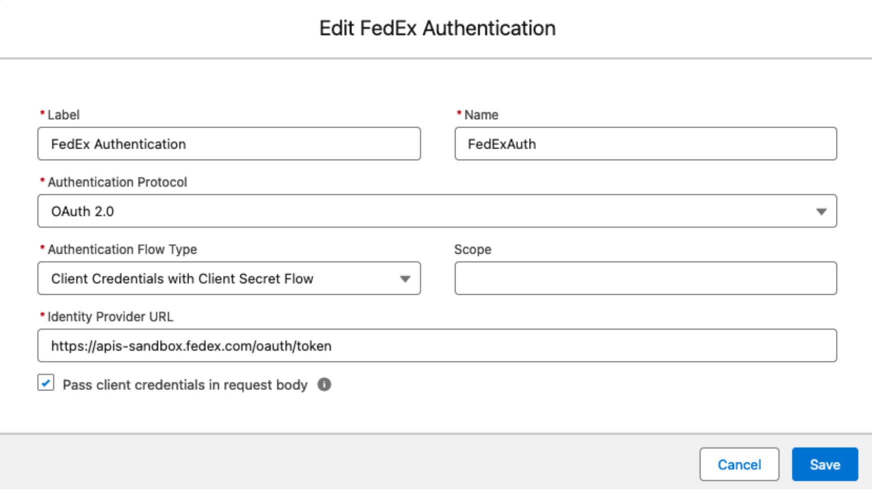
<table>
<tr>
<td></td>
<td></td>
<td></td>
<td>
SETUP
Custom Metadata Types
HTTP Service
Help for this Page
HTTP Service Edit
Save Save & New Cancel
Information
= Required Information
Label: FedEx Shipping Service
HTTP Service Name: FedEx_Shipping_Service
HTTP Service ~ Salesforce - Developer Edition
Protected Component: □
Impl Class:
Named Credentials: FedEx_Shipping
Service Mode: Live
HTTP Method: POST
End Point: /rate/v1/rates/quotes
Service Timeout: 5000
Success Codes: 200,201
Error Codes: 400,403,404,500,503
Auth Codes: 401
Mocked Response: {
"transactionId": "APIF_SV_RATC_TxID7c3f6154-cac4-49e8-bdc5-313581bdced1",
"customerTransactionId": "customer test",
"output": {
"alerts": [
Namespace Prefix
Save Save & New Cancel</td>
</tr>
<tr>
<td>3</td>
<td>Salesforce Org</td>
<td>Create External Credentials</td>
<td>Create <b>External Credentials</b> for OAuth 2.0,

SETUP
Named Credentials
Named Credentials | External Credentials
1 Items - Sorted by Label
New
<table><tr><th>Label</th><th>Authentication Protocol</th><th>Actions</th></tr><tr><td>UPS Authentication</td><td>OAuth 2.0</td><td></td></tr></table>
Provide details , like

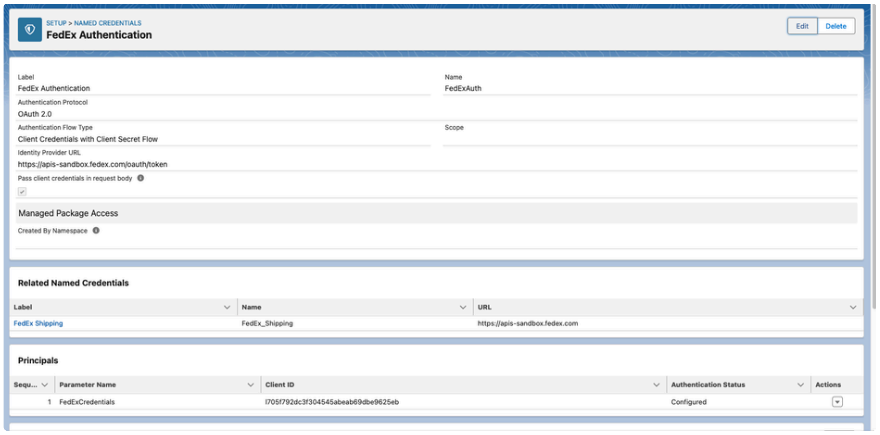
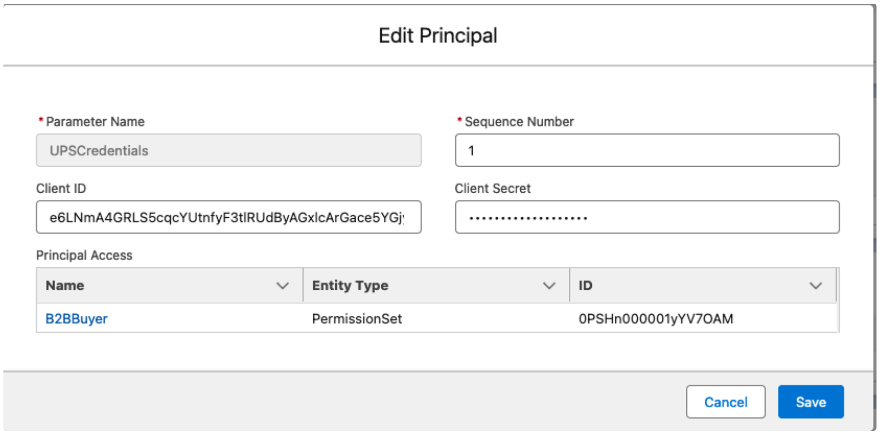
Name - FedExAuth

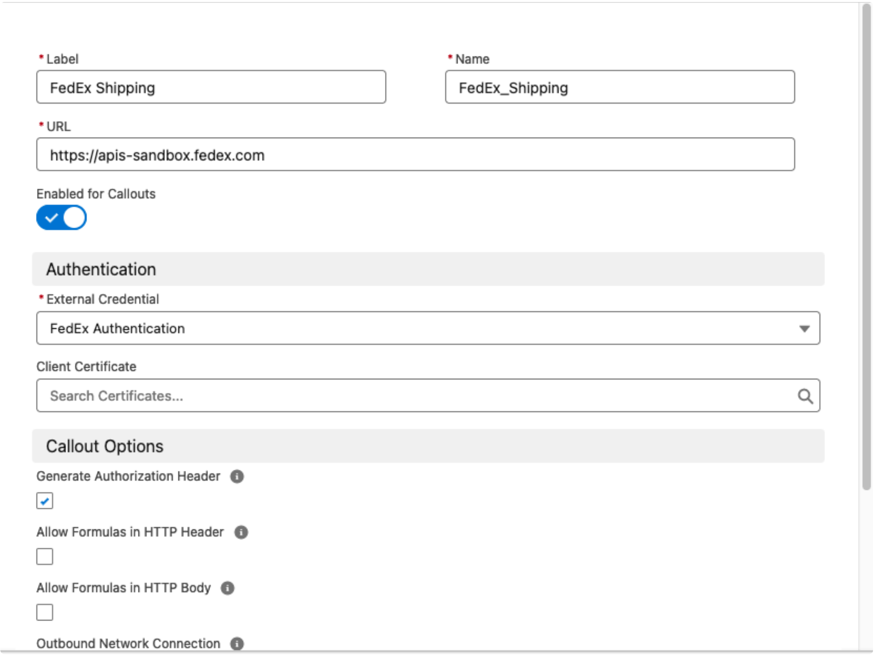
Authentication Protocol - AOuth 2.0

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
Scope:
* Identity Provider URL: https://apis-sandbox.fedex.com/oauth/token
☑ Pass client credentials in request body
Cancel Save</td>
</tr>
</table>

			 SETUP > NAMED CREDENTIALS **FedEx Authentication** Label: FedEx Authentication Name: FedExAuth Authentication Protocol: OAuth 2.0 Authentication Flow Type: Client Credentials with Client Secret Flow Scope: Identity Provider URL: https://apis-sandbox.fedex.com/oauth/token Pass client credentials in request body: ☑ Managed Package Access Created By Namespace: **Related Named Credentials** Label: FedEx Shipping — Name: FedEx_Shipping — URL: https://apis-sandbox.fedex.com **Principals** Sequ...: 1 — Parameter Name: FedExCredentials — Client ID: l705f792dc3f304545abeab69dbe9625eb — Authentication Status: Configured Also create a Principal which will store Client ID & Client Secret Name - FedExCredentials  Edit Principal * Parameter Name: UPSCredentials * Sequence Number: 1 Client ID: e6LNmA4GRLS5cqcYUtnfyF3tlRUdByAGxlcArGace5YGj Client Secret: •••••••••••••••••••• Principal Access Name: B2BBuyer — Entity Type: PermissionSet — ID: 0PSHn000001yYV7OAM
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



* Label: FedEx Shipping

* Name: FedEx_Shipping

* URL: https://apis-sandbox.fedex.com

Enabled for Callouts: ✓

Authentication

* External Credential: FedEx Authentication

Client Certificate: Search Certificates...

Callout Options

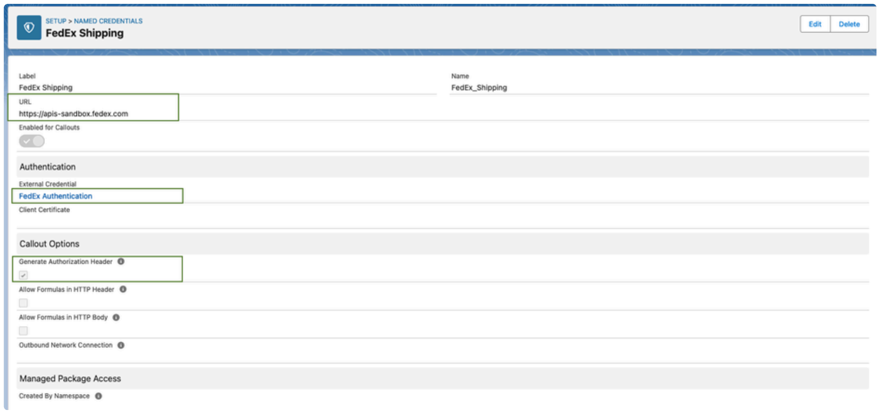
Generate Authorization Header: ☑

Allow Formulas in HTTP Header: ☐

Allow Formulas in HTTP Body: ☐

Outbound Network Connection

Here are the details



SETUP > NAMED CREDENTIALS

FedEx Shipping

Label: FedEx Shipping

Name: FedEx_Shipping

URL: https://apis-sandbox.fedex.com

Enabled for Callouts: ✓

Authentication

External Credential: FedEx Authentication

Client Certificate

Callout Options

Generate Authorization Header: ☑

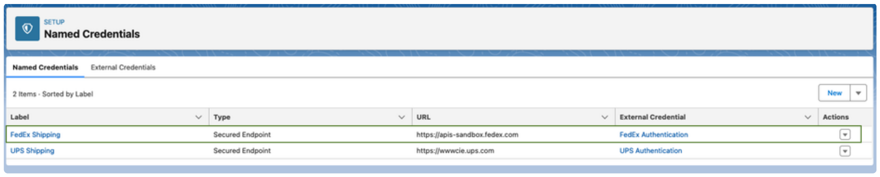
Allow Formulas in HTTP Header: ☐

Allow Formulas in HTTP Body: ☐

Outbound Network Connection

Managed Package Access

Created By Namespace



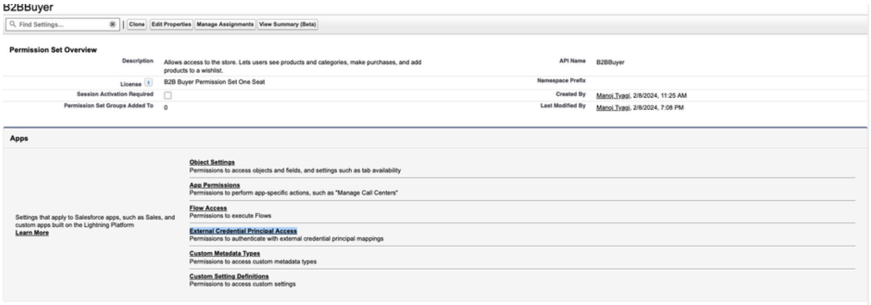
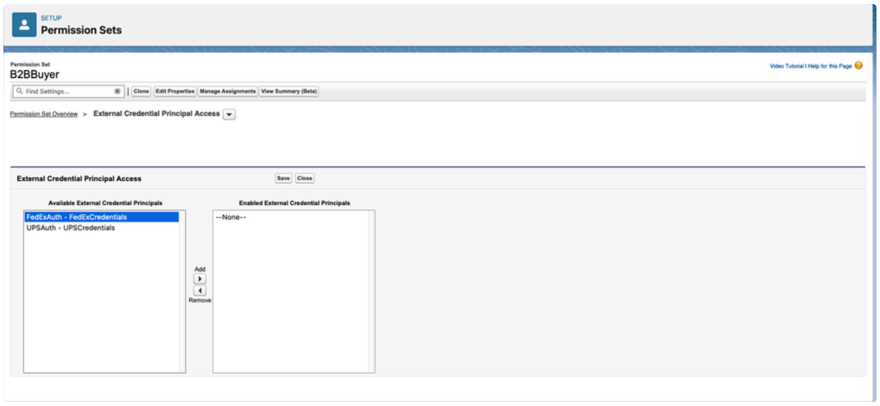
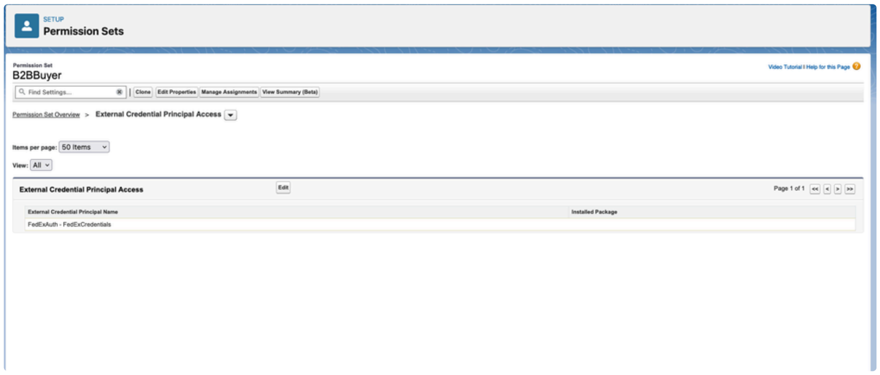
SETUP

Named 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://wwwcie.ups.com	UPS Authentication	▾

5	Salesforce Org	Assign Permission	To make callouts that use a named credential, you must enable principal access to permission sets or profiles for the users who need access. Setup > Permission Sets > B2BBuyer Click **External Credential Principal Access**

			 Find Settings... Clone Edit Properties Manage Assignments View Summary (Beta) B2BBuyer Permission Set Overview Description: Allows access to the store. Lets users see products and categories, make purchases, and add products to a wishlist. API Name: B2BBuyer License: B2B Buyer Permission Set One Seat Namespace Prefix: Session Activation Required: □ Created By: Manoj Tyagi, 2/8/2024, 11:25 AM Permission Set Groups Added To: 0 Last Modified By: Manoj Tyagi, 2/8/2024, 7:08 PM Apps Settings that apply to Salesforce apps, such as Sales, and custom apps built on the Lightning Platform Learn More 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 Now click edit and select principal created for FedEx in above steps(step # 3)  SETUP Permission Sets Video Tutorial I Help for this Page Find Settings... Clone Edit Properties Manage Assignments View Summary (Beta) Save Close Add Remove Permission Set B2BBuyer Permission Set Overview > External Credential Principal Access External Credential Principal Access Available External Credential Principals: FedExAuth - FedExCredentials; UPSAuth - UPSCredentials Enabled External Credential Principals: --None--  SETUP Permission Sets Video Tutorial I Help for this Page Find Settings... Clone Edit Properties Manage Assignments View Summary (Beta) Edit Page 1 of 1 << < > >> Permission Set B2BBuyer Permission Set Overview > External Credential Principal Access Items per page: 50 Items View: All External Credential Principal Access External Credential Principal Name: FedExAuth - FedExCredentials Installed Package: 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

```
{
  "20": "FedEx Express Saver®",
  "39": "FedEx First Overnight Freight",
  "49": "FedEx 2Day®AM",
  "70": "FedEx 1Day Freight",
  "80": "FedEx 2Day®Freight",
  "83": "FedEx 3Day®Freight",
  "90": "FedEx Home Delivery®",
  "92": "FedEx Ground®",
  "111": "FedEx Freight®",
  "112": "FedEx Freight®Priority",
  "113": "FedEx Freight®Economy",
  "01": "FedEx Priority Overnight®",
  "03": "FedEx 2Day® ",
  "05": "FedEx Standard Overnight®",
  "06": "FedEx First Overnight®",
  "E1": "Extra Hours Priority Overnight",
  "E5": "Extra Hours Standard Overnight",
  "E6": "Extra Hours First Overnight"
}
```

Request

```
{
  "accountNumber": {
    "value": "XXXXX7364"
  },
  "requestedShipment": {
    "shipper": {
      "address": {
        "postalCode": 65247,
        "countryCode": "US"
      }
    },
    "recipient": {
      "address": {
        "postalCode": 75063,
        "countryCode": "US"
      }
    },
    "pickupType": "DROPOFF_AT_FEDEX_LOCATION",
    "rateRequestType": [
      "LIST"
    ],
    "requestedPackageLineItems": [
      {
        "weight": {
          "units": "LB",
          "value": 0
        }
      }
    ]
  }
}
```

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	*This is optional so could be left blank*
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	

```
{
  "transactionId": "APIF_SV_RATC_TxID7c3f6154-cac4-49e8-bdc5-313581bdced1",
  "customerTransactionId": "customer test",
  "output": {
    "alerts": [
      {
        "code": "VIRTUAL.RESPONSE",
        "message": "This is a Virtual Response.",
        "alertType": "NOTE"
      },
      {
        "code": "ORIGIN.STATEORPROVINCECODE.CHANGED",
        "message": "The origin state/province code has been changed.",
        "alertType": "NOTE"
      },
      {
        "code": "DESTINATION.STATEORPROVINCECODE.CHANGED",
        "message": "The destination state/province code has been changed.",
        "alertType": "NOTE"
      }
    ],
    "rateReplyDetails": [
      {
        "serviceType": "PRIORITY_OVERNIGHT",
        "serviceName": "FedEx Priority Overnight®",
        "packagingType": "YOUR_PACKAGING",
        "ratedShipmentDetails": [
          {
            "rateType": "LIST",
            "ratedWeightMethod": "ACTUAL",
            "totalDiscounts": 0.0,
            "totalBaseCharge": 53.23,
            "totalNetCharge": 66.25,
            "totalNetFedExCharge": 66.25,
            "shipmentRateDetail": {
              "rateZone": "02",
              "dimDivisor": 0,
              "fuelSurchargePercent": 14.75,
              "totalSurcharges": 13.02,
              "totalFreightDiscount": 0.0,
              "surCharges": [
                {
                  "type": "FUEL",
                  "description": "Fuel Surcharge",
                  "amount": 8.52
                },
                {
                  "type": "DELIVERY_AREA",
                  "description": "DAS Extended Comm",
                  "amount": 4.5
                }
              ],
              "pricingCode": "PACKAGE",
              "totalBillingWeight": {
                "units": "LB",
                "value": 10.0
              },
              "currency": "USD",
              "rateScale": "1486"
            },
            "ratedPackages": [
              {
                "groupNumber": 0,
                "effectiveNetDiscount": 0.0,
                "packageRateDetail": {
                  "rateType": "PAYOR_ACCOUNT_PACKAGE",
                  "ratedWeightMethod": "ACTUAL",
                  "baseCharge": 53.23,
                  "netFreight": 53.23,
                  "totalSurcharges": 13.02,
                  "netFedExCharge": 66.25,
                  "totalTaxes": 0.0,
                  "netCharge": 66.25,
                  "totalRebates": 0.0,
                  "billingWeight": {
                    "units": "LB",
                    "value": 10.0
                  },
                  "totalFreightDiscounts": 0.0,
                  "surcharges": [
                    {
                      "type": "FUEL",
                      "description": "Fuel Surcharge",
                      "amount": 8.52
                      },
                      {
                        "type": "DELIVERY_AREA",
                        "description": "DAS Extended Comm",
                        "amount": 4.5
                      }
                    ],
                    "currency": "USD"
                  }
                }
              ],
              "currency": "USD"
            }
          ],
          "operationalDetail": {
            "ineligibleForMoneyBackGuarantee": false,
            "astraDescription": "P1",
            "airportId": "TLH",
            "serviceCode": "01"
          },
          "signatureOptionType": "SERVICE_DEFAULT",
          "serviceDescription": {
            "serviceId": "EP1000000002",
            "serviceType": "PRIORITY_OVERNIGHT",
            "code": "01",
            "names": [
              {
                "type": "long",
                "encoding": "utf-8",
                "value": "FedEx Priority Overnight®"
              },
              {
                "type": "long",
                "encoding": "ascii",
                "value": "FedEx Priority Overnight"
              },
              {
                "type": "medium",
                "encoding": "utf-8",
                "value": "FedEx Priority Overnight®"
              },
              {
                "type": "medium",
                "encoding": "ascii",
                "value": "FedEx Priority Overnight"
              },
              {
                "type": "short",
                "encoding": "utf-8",
                "value": "P-1"
              },
              {
                "type": "short",
                "encoding": "ascii",
                "value": "P-1"
              },
              {
                "type": "abbrv",
                "encoding": "ascii",
                "value": "PO"
            }
          ],
          "serviceCategory": "parcel",
          "description": "Priority Overnight",
          "astraDescription": "P1"
        }
      },
      {
        "serviceType": "FEDEX_2_DAY",
        "serviceName": "FedEx 2Day®",
        "packagingType": "YOUR_PACKAGING",
        "ratedShipmentDetails": [
          {
            "rateType": "LIST",
            "ratedWeightMethod": "ACTUAL",
            "totalDiscounts": 0.0,
            "totalBaseCharge": 30.4,
            "totalNetCharge": 40.05,
            "totalNetFedExCharge": 40.05,
            "shipmentRateDetail": {
              "rateZone": "02",
              "dimDivisor": 0,
              "fuelSurchargePercent": 14.75,
              "totalSurcharges": 9.65,
              "totalFreightDiscount": 0.0,
              "surCharges": [
                {
                  "type": "FUEL",
                  "description": "Fuel Surcharge",
                  "amount": 5.15
                },
                {
                  "type": "DELIVERY_AREA",
                  "description": "DAS Extended Comm",
                  "amount": 4.5
                }
              ],
              "pricingCode": "PACKAGE",
              "totalBillingWeight": {
                "units": "LB",
                "value": 10.0
              },
              "currency": "USD",
              "rateScale": "5980"
            },
            "ratedPackages": [
              {
                "groupNumber": 0,
                "effectiveNetDiscount": 0.0,
                "packageRateDetail": {
                  "rateType": "PAYOR_ACCOUNT_PACKAGE",
                  "ratedWeightMethod": "ACTUAL",
                  "baseCharge": 30.4,
                  "netFreight": 30.4,
                  "totalSurcharges": 9.65,
                  "netFedExCharge": 40.05,
                  "totalTaxes": 0.0,
                  "netCharge": 40.05,
                  "totalRebates": 0.0,
                  "billingWeight": {
                    "units": "LB",
                    "value": 10.0
                  },
                  "totalFreightDiscounts": 0.0,
                  "surcharges": [
                    {
                      "type": "FUEL",
                      "description": "Fuel Surcharge",
                      "amount": 5.15
                    },
                    {
                      "type": "DELIVERY_AREA",
                      "description": "DAS Extended Comm",
                      "amount": 4.5
                    }
                  ],
                  "currency": "USD"
                }
              }
            ],
            "currency": "USD"
          }
        ],
        "operationalDetail": {
          "ineligibleForMoneyBackGuarantee": false,
          "astraDescription": "E2",
          "airportId": "TLH",
          "serviceCode": "03"
        },
        "signatureOptionType": "SERVICE_DEFAULT",
        "serviceDescription": {
          "serviceId": "EP1000000003",
          "serviceType": "FEDEX_2_DAY",
          "code": "03",
          "names": [
            {
              "type": "long",
              "encoding": "utf-8",
              "value": "FedEx 2Day®"
            },
            {
              "type": "long",
              "encoding": "ascii",
              "value": "FedEx 2Day"
            },
            {
              "type": "medium",
              "encoding": "utf-8",
              "value": "FedEx 2Day®"
            },
            {
              "type": "medium",
              "encoding": "ascii",
              "value": "FedEx 2Day"
            },
            {
              "type": "short",
              "encoding": "utf-8",
              "value": "P-2"
            },
            {
              "type": "short",
              "encoding": "ascii",
              "value": "P-2"
            },
            {
              "type": "abbrv",
              "encoding": "ascii",
              "value": "ES"
            }
          ],
          "serviceCategory": "parcel",
          "description": "2Day",
          "astraDescription": "E2"
        }
      },
      {
        "serviceType": "FEDEX_GROUND",
        "serviceName": "FedEx Ground®",
        "packagingType": "YOUR_PACKAGING",
        "ratedShipmentDetails": [
          {
            "rateType": "LIST",
            "ratedWeightMethod": "ACTUAL",
            "totalDiscounts": 0.0,
            "totalBaseCharge": 13.51,
            "totalNetCharge": 20.53,
            "totalNetFedExCharge": 20.53,
            "shipmentRateDetail": {
              "rateZone": "2",
              "dimDivisor": 0,
              "fuelSurchargePercent": 14.0,
              "totalSurcharges": 7.02,
              "totalFreightDiscount": 0.0,
              "surCharges": [
                {
                  "type": "FUEL",
                  "description": "Fuel Surcharge",
                  "level": "PACKAGE",
                  "amount": 2.52
                },
                {
                  "type": "DELIVERY_AREA",
                  "description": "DAS Extended Comm",
                  "level": "PACKAGE",
                  "amount": 4.5
                }
              ],
              "totalBillingWeight": {
                "units": "LB",
                "value": 10.0
              },
              "currency": "USD"
            },
            "ratedPackages": [
              {
                "groupNumber": 0,
                "effectiveNetDiscount": 0.0,
                "packageRateDetail": {
                  "rateType": "PAYOR_ACCOUNT_PACKAGE",
                  "ratedWeightMethod": "ACTUAL",
                  "baseCharge": 13.51,
                  "netFreight": 13.51,
                  "totalSurcharges": 7.02,
                  "netFedExCharge": 20.53,
                  "totalTaxes": 0.0,
                  "netCharge": 20.53,
                  "totalRebates": 0.0,
                  "billingWeight": {
                    "units": "LB",
                    "value": 10.0
                  },
                  "totalFreightDiscounts": 0.0,
                  "surcharges": [
                    {
                      "type": "FUEL",
                      "description": "Fuel Surcharge",
                      "level": "PACKAGE",
                      "amount": 2.52
                    },
                    {
                      "type": "DELIVERY_AREA",
                      "description": "DAS Extended Comm",
                      "level": "PACKAGE",
                      "amount": 4.5
                    }
                  ],
                  "currency": "USD"
                }
              }
            ],
            "currency": "USD"
          }
        ],
        "operationalDetail": {
          "ineligibleForMoneyBackGuarantee": false,
          "astraDescription": "FXG",
          "airportId": "TLH",
          "serviceCode": "92"
        },
        "signatureOptionType": "SERVICE_DEFAULT",
        "serviceDescription": {
          "serviceId": "EP1000000134",
          "serviceType": "FEDEX_GROUND",
          "code": "92",
          "names": [
            {
              "type": "long",
              "encoding": "utf-8",
              "value": "FedEx Ground®"
            },
            {
              "type": "long",
              "encoding": "ascii",
              "value": "FedEx Ground"
            },
            {
              "type": "medium",
              "encoding": "utf-8",
              "value": "Ground®"
            },
            {
              "type": "medium",
              "encoding": "ascii",
              "value": "Ground"
            },
            {
              "type": "short",
              "encoding": "utf-8",
              "value": "FG"
            },
            {
              "type": "short",
              "encoding": "ascii",
              "value": "FG"
            },
            {
              "type": "abbrv",
              "encoding": "ascii",
              "value": "SG"
            }
          ],
          "description": "FedEx Ground",
          "astraDescription": "FXG"
        }
      }
    ],
    "quoteDate": "2024-02-26",
    "encoded": false
  }
}
```

Source Code

<table>
<tr><th>Class</th><th>Details</th><th>Code</th></tr>
<tr><td>FedExShippingProvider</td><td>This class extends ShippingProvider class & implements following methods -
<b>prepareRequestBody</b>
<b>getShippingOptionsFromResponse</b></td><td>

```
public with sharing class FedExShippingProvider extends Shippin
  public FedExShippingProvider() {
  }
  public override void prepareRequestBody(
    ShippingProviderRequest shippingRequest,
    Map<String, String> callOutRequest
  ) {
    FedExShippingRateRequest requestObject = FedExShippingRateR
      shippingRequest.shippingMetaData.requestJSON
    );
    if (requestObject != null) {
      requestObject.requestedShipment.recipient.address = setSh
        shippingRequest
      );
      FedExShippingRateRequest.RequestedPackageLineItems packag
        0
      );
      FedExShippingRateRequest.Weight weight = new FedExShippin
      packageLItem.weight.value = shippingRequest.packageWeight

```

</td></tr>
</table>

```
      String strRatingRequestBody = JSON.serialize(requestObjec
      callOutRequest.put(Constants.SERVICE_REQUEST_BODY, strRat
    }
  }

  public virtual FedExShippingRateRequest.Address setShipToAddr
    ShippingProviderRequest shippingRequest
  ) {
    FedExShippingRateRequest.Address address = new FedExShippin
    address.postalCode = shippingRequest.postalCode;
    address.countryCode = shippingRequest.country;
    return address;
  }

  public override Map<String, ShippingProviderResponse> getShip
    List<String> responseList,
    ShippingProviderRequest shippingRequest
  ) {
    Map<String, Object> shippingMethods = new Map<String, Objec
    Map<String, ShippingProviderResponse> shippingMethodsWithRa
      responseList
    );
    return shippingMethodsWithRate;
  }

  public virtual Map<String, ShippingProviderResponse> parseFed
    List<String> lstFedExResponseBody
  ) {
    Map<String, ShippingProviderResponse> shippingMethodsWithRa
    ShippingProviderResponse shippingResponse;
    if (lstFedExResponseBody != null && lstFedExResponseBody.si
      for (String strFedExResponseBody : lstFedExResponseBody)
        FedExShippingRateResponse fedexShippingRateResponseObj
          strFedExResponseBody
        );
        FedExShippingRateResponse.Output fedexShippingRateRespo
        for (
          FedExShippingRateResponse.RateReplyDetails shipmentOb
        ) {
          Decimal shippingAmount = shipmentObj.ratedShipmentDet
              .totalNetCharge != null
            ? Decimal.valueOf(
                shipmentObj.ratedShipmentDetails[0].totalNetCha
              )
            : 0;
          if (
            shippingMethodsWithRate.containsKey(
              shipmentObj.operationalDetail.serviceCode
            )
          ) {
            shippingResponse = shippingMethodsWithRate.get(ship
            shippingResponse.cost = shippingResponse.cost + shi
          } else {
             shippingResponse = new ShippingProviderResponse();
             shippingResponse.serviceCode = shipmentObj.operati
             shippingResponse.cost = shippingAmount
            shippingMethodsWithRate.put(
              shipmentObj.operationalDetail.serviceCode,
              shippingResponse
            );
```

<table>
<tr><td></td><td></td><td><pre>
     }
    }
   }
  }
  return shippingMethodsWithRate;
 }
}

</pre></td></tr>
<tr><td>FedExShippingRateRequest</td><td>This class is a DTO used to prepare FedEx Shipping rate request which is posted to FedEx to get shipping details</td><td><pre>
public class FedExShippingRateRequest {
 public AccountNumber accountNumber;
 public RequestedShipment requestedShipment;

 public class Address {
  public String postalCode;
  public String countryCode;
 }

 public class Shipper {
  public Address address;
 }

 public class RequestedPackageLineItems {
  public Weight weight;
 }

 public class RequestedShipment {
  public Shipper shipper;
  public Shipper recipient;
  public String pickupType;
  public List<String> rateRequestType;
  public List<RequestedPackageLineItems> requestedPackageLine
 }

 public class Weight {
  public String units;
  public Decimal value;
 }

 public class AccountNumber {
  public String value;
 }

 public static FedExShippingRateRequest parse(String json) {
  return (FedExShippingRateRequest) System.JSON.deserialize(
   json,
   FedExShippingRateRequest.class
  );
 }
}

</pre></td></tr>
<tr><td>FedExShippingRateResponse</td><td>This class is a DTO used to parse shipping rate response returned by FedEx</td><td><pre>
public class FedExShippingRateResponse {
 public class ShipmentRateDetail_Z {
  public String rateZone { get; set; }
  public Integer dimDivisor { get; set; }
</pre></td></tr>
</table>

```
    public Double fuelSurchargePercent { get; set; }
    public Double totalSurcharges { get; set; }
    public Double totalFreightDiscount { get; set; }
    public List<SurCharges_Z> surCharges { get; set; }
    public TotalBillingWeight totalBillingWeight { get; set; }
    public String currency_Z { get; set; } // in json: currenc

    public ShipmentRateDetail_Z(JSONParser parser) {
      while (parser.nextToken() != System.JSONToken.END_OBJECT
        if (parser.getCurrentToken() == System.JSONToken.FIELD
          String text = parser.getText();
          if (parser.nextToken() != System.JSONToken.VALUE_NUL
            if (text == 'rateZone') {
              rateZone = parser.getText();
            } else if (text == 'dimDivisor') {
              dimDivisor = parser.getIntegerValue();
            } else if (text == 'fuelSurchargePercent') {
              fuelSurchargePercent = parser.getDoubleValue();
            } else if (text == 'totalSurcharges') {
              totalSurcharges = parser.getDoubleValue();
            } else if (text == 'totalFreightDiscount') {
              totalFreightDiscount = parser.getDoubleValue();
            } else if (text == 'surCharges') {
              surCharges = arrayOfSurCharges_Z(parser);
            } else if (text == 'totalBillingWeight') {
              totalBillingWeight = new TotalBillingWeight(pars
            } else if (text == 'currency') {
              currency_Z = parser.getText();
            } else {
              System.debug(
                LoggingLevel.WARN,
                'ShipmentRateDetail_Z consuming unrecognized p
              );
              consumeObject(parser);
            }
          }
        }
      }
    }
  }

  public class RatedPackages_Z {
    public Integer groupNumber { get; set; }
    public Double effectiveNetDiscount { get; set; }
    public PackageRateDetail_Z packageRateDetail { get; set; }

    public RatedPackages_Z(JSONParser parser) {
      while (parser.nextToken() != System.JSONToken.END_OBJECT
        if (parser.getCurrentToken() == System.JSONToken.FIELD
          String text = parser.getText();
          if (parser.nextToken() != System.JSONToken.VALUE_NUL
            if (text == 'groupNumber') {
              groupNumber = parser.getIntegerValue();
            } else if (text == 'effectiveNetDiscount') {
              effectiveNetDiscount = parser.getDoubleValue();
            } else if (text == 'packageRateDetail') {
              packageRateDetail = new PackageRateDetail_Z(pars
            } else {
              System.debug(
                LoggingLevel.WARN,
                'RatedPackages_Z consuming unrecognized proper
              );
              consumeObject(parser);
            }
          }
        }
      }
    }
  }

  public String transactionId { get; set; }
  public String customerTransactionId { get; set; }
  public Output output { get; set; }

  public FedExShippingRateResponse(JSONParser parser) {
    while (parser.nextToken() != System.JSONToken.END_OBJECT)
      if (parser.getCurrentToken() == System.JSONToken.FIELD_N
        String text = parser.getText();
        if (parser.nextToken() != System.JSONToken.VALUE_NULL)
          if (text == 'transactionId') {
            transactionId = parser.getText();
          } else if (text == 'customerTransactionId') {
            customerTransactionId = parser.getText();
          } else if (text == 'output') {
            output = new Output(parser);
          } else {
            System.debug(
              LoggingLevel.WARN,
              'FedExShippingRateResponse consuming unrecognize
              text
            );
            consumeObject(parser);
          }
        }
      }
    }
  }

  public class ServiceDescription_Z {
    public String serviceId { get; set; }
    public String serviceType { get; set; }
    public String code { get; set; }
    public List<Names> names { get; set; }
    public String description { get; set; }
    public String astraDescription { get; set; }

    public ServiceDescription_Z(JSONParser parser) {
      while (parser.nextToken() != System.JSONToken.END_OBJECT
        if (parser.getCurrentToken() == System.JSONToken.FIELD
          String text = parser.getText();
          if (parser.nextToken() != System.JSONToken.VALUE_NUL
            if (text == 'serviceId') {
              serviceId = parser.getText();
            } else if (text == 'serviceType') {
              serviceType = parser.getText();
            } else if (text == 'code') {
              code = parser.getText();
            } else if (text == 'names') {
              names = arrayOfNames(parser);
            } else if (text == 'description') {
```

```
            description = parser.getText();
          } else if (text == 'astraDescription') {
            astraDescription = parser.getText();
          } else {
            System.debug(
              LoggingLevel.WARN,
              'ServiceDescription_Z consuming unrecognized p
            );
            consumeObject(parser);
          }
        }
      }
    }
  }
}

public class ServiceDescription {
  public String serviceId { get; set; }
  public String serviceType { get; set; }
  public String code { get; set; }
  public List<Names> names { get; set; }
  public String serviceCategory { get; set; }
  public String description { get; set; }
  public String astraDescription { get; set; }

  public ServiceDescription(JSONParser parser) {
    while (parser.nextToken() != System.JSONToken.END_OBJECT
      if (parser.getCurrentToken() == System.JSONToken.FIELD
        String text = parser.getText();
        if (parser.nextToken() != System.JSONToken.VALUE_NUL
          if (text == 'serviceId') {
            serviceId = parser.getText();
          } else if (text == 'serviceType') {
            serviceType = parser.getText();
          } else if (text == 'code') {
            code = parser.getText();
          } else if (text == 'names') {
            names = arrayOfNames(parser);
          } else if (text == 'serviceCategory') {
            serviceCategory = parser.getText();
          } else if (text == 'description') {
            description = parser.getText();
          } else if (text == 'astraDescription') {
            astraDescription = parser.getText();
          } else {
            System.debug(
              LoggingLevel.WARN,
              'ServiceDescription consuming unrecognized pro
            );
            consumeObject(parser);
          }
        }
      }
    }
  }
}

public class PackageRateDetail_Z {
  public String rateType { get; set; }
  public String ratedWeightMethod { get; set; }
```

```
    public Double baseCharge { get; set; }
    public Double netFreight { get; set; }
    public Double totalSurcharges { get; set; }
    public Double netFedExCharge { get; set; }
    public Double totalTaxes { get; set; }
    public Double netCharge { get; set; }
    public Double totalRebates { get; set; }
    public TotalBillingWeight billingWeight { get; set; }
    public Double totalFreightDiscounts { get; set; }
    public List<SurCharges_Z> surcharges { get; set; }
    public String currency_Z { get; set; } // in json: currenc

    public PackageRateDetail_Z(JSONParser parser) {
      while (parser.nextToken() != System.JSONToken.END_OBJECT
        if (parser.getCurrentToken() == System.JSONToken.FIELD
          String text = parser.getText();
          if (parser.nextToken() != System.JSONToken.VALUE_NUL
            if (text == 'rateType') {
              rateType = parser.getText();
            } else if (text == 'ratedWeightMethod') {
              ratedWeightMethod = parser.getText();
            } else if (text == 'baseCharge') {
              baseCharge = parser.getDoubleValue();
            } else if (text == 'netFreight') {
              netFreight = parser.getDoubleValue();
            } else if (text == 'totalSurcharges') {
              totalSurcharges = parser.getDoubleValue();
            } else if (text == 'netFedExCharge') {
              netFedExCharge = parser.getDoubleValue();
            } else if (text == 'totalTaxes') {
              totalTaxes = parser.getDoubleValue();
            } else if (text == 'netCharge') {
              netCharge = parser.getDoubleValue();
            } else if (text == 'totalRebates') {
              totalRebates = parser.getDoubleValue();
            } else if (text == 'billingWeight') {
              billingWeight = new TotalBillingWeight(parser);
            } else if (text == 'totalFreightDiscounts') {
              totalFreightDiscounts = parser.getDoubleValue();
            } else if (text == 'surcharges') {
              surcharges = arrayOfSurCharges_Z(parser);
            } else if (text == 'currency') {
              currency_Z = parser.getText();
            } else {
              System.debug(
                LoggingLevel.WARN,
                'PackageRateDetail_Z consuming unrecognized pr
              );
              consumeObject(parser);
            }
          }
        }
      }
    }
  }

  public class OperationalDetail {
    public Boolean ineligibleForMoneyBackGuarantee { get; set;
    public String astraDescription { get; set; }
    public String airportId { get; set; }
    public String serviceCode { get; set; }

    public OperationalDetail(JSONParser parser) {
      while (parser.nextToken() != System.JSONToken.END_OBJECT
        if (parser.getCurrentToken() == System.JSONToken.FIELD
          String text = parser.getText();
          if (parser.nextToken() != System.JSONToken.VALUE_NUL
            if (text == 'ineligibleForMoneyBackGuarantee') {
              ineligibleForMoneyBackGuarantee = parser.getBool
            } else if (text == 'astraDescription') {
              astraDescription = parser.getText();
            } else if (text == 'airportId') {
              airportId = parser.getText();
            } else if (text == 'serviceCode') {
              serviceCode = parser.getText();
            } else {
              System.debug(
                LoggingLevel.WARN,
                'OperationalDetail consuming unrecognized prop
              );
              consumeObject(parser);
            }
          }
        }
      }
    }
  }

  public class RatedPackages {
    public Integer groupNumber { get; set; }
    public Double effectiveNetDiscount { get; set; }
    public PackageRateDetail packageRateDetail { get; set; }

    public RatedPackages(JSONParser parser) {
      while (parser.nextToken() != System.JSONToken.END_OBJECT
        if (parser.getCurrentToken() == System.JSONToken.FIELD
          String text = parser.getText();
          if (parser.nextToken() != System.JSONToken.VALUE_NUL
            if (text == 'groupNumber') {
              groupNumber = parser.getIntegerValue();
            } else if (text == 'effectiveNetDiscount') {
              effectiveNetDiscount = parser.getDoubleValue();
            } else if (text == 'packageRateDetail') {
              packageRateDetail = new PackageRateDetail(parser
            } else {
              System.debug(
                LoggingLevel.WARN,
                'RatedPackages consuming unrecognized property
              );
              consumeObject(parser);
            }
          }
        }
      }
    }
  }

  public class RatedShipmentDetails {
    public String rateType { get; set; }
    public String ratedWeightMethod { get; set; }
```

```
    public Double totalDiscounts { get; set; }
    public Double totalBaseCharge { get; set; }
    public Double totalNetCharge { get; set; }
    public Double totalNetFedExCharge { get; set; }
    public ShipmentRateDetail shipmentRateDetail { get; set; }
    public List<RatedPackages> ratedPackages { get; set; }
    public String currency_Z { get; set; } // in json: currenc

    public RatedShipmentDetails(JSONParser parser) {
      while (parser.nextToken() != System.JSONToken.END_OBJECT
        if (parser.getCurrentToken() == System.JSONToken.FIELD
          String text = parser.getText();
          if (parser.nextToken() != System.JSONToken.VALUE_NUL
            if (text == 'rateType') {
              rateType = parser.getText();
            } else if (text == 'ratedWeightMethod') {
              ratedWeightMethod = parser.getText();
            } else if (text == 'totalDiscounts') {
              totalDiscounts = parser.getDoubleValue();
            } else if (text == 'totalBaseCharge') {
              totalBaseCharge = parser.getDoubleValue();
            } else if (text == 'totalNetCharge') {
              totalNetCharge = parser.getDoubleValue();
            } else if (text == 'totalNetFedExCharge') {
              totalNetFedExCharge = parser.getDoubleValue();
            } else if (text == 'shipmentRateDetail') {
              shipmentRateDetail = new ShipmentRateDetail(pars
            } else if (text == 'ratedPackages') {
              ratedPackages = arrayOfRatedPackages(parser);
            } else if (text == 'currency') {
              currency_Z = parser.getText();
            } else {
              System.debug(
                LoggingLevel.WARN,
                'RatedShipmentDetails consuming unrecognized p
              );
              consumeObject(parser);
            }
          }
        }
      }
    }
  }

  public class RatedShipmentDetails_Z {
    public String rateType { get; set; }
    public String ratedWeightMethod { get; set; }
    public Double totalDiscounts { get; set; }
    public Double totalBaseCharge { get; set; }
    public Double totalNetCharge { get; set; }
    public Double totalNetFedExCharge { get; set; }
    public ShipmentRateDetail_Z shipmentRateDetail { get; set;
    public List<RatedPackages_Z> ratedPackages { get; set; }
    public String currency_Z { get; set; } // in json: currenc

    public RatedShipmentDetails_Z(JSONParser parser) {
      while (parser.nextToken() != System.JSONToken.END_OBJECT
        if (parser.getCurrentToken() == System.JSONToken.FIELD
          String text = parser.getText();
          if (parser.nextToken() != System.JSONToken.VALUE_NUL
            if (text == 'rateType') {
              rateType = parser.getText();
            } else if (text == 'ratedWeightMethod') {
              ratedWeightMethod = parser.getText();
            } else if (text == 'totalDiscounts') {
              totalDiscounts = parser.getDoubleValue();
            } else if (text == 'totalBaseCharge') {
              totalBaseCharge = parser.getDoubleValue();
            } else if (text == 'totalNetCharge') {
              totalNetCharge = parser.getDoubleValue();
            } else if (text == 'totalNetFedExCharge') {
              totalNetFedExCharge = parser.getDoubleValue();
            } else if (text == 'shipmentRateDetail') {
              shipmentRateDetail = new ShipmentRateDetail_Z(pa
            } else if (text == 'ratedPackages') {
              ratedPackages = arrayOfRatedPackages_Z(parser);
            } else if (text == 'currency') {
              currency_Z = parser.getText();
            } else {
              System.debug(
                LoggingLevel.WARN,
                'RatedShipmentDetails_Z consuming unrecognized
                text
              );
              consumeObject(parser);
            }
          }
        }
      }
    }
  }

  public class SurCharges {
    public String type { get; set; }
    public String description { get; set; }
    public Double amount { get; set; }

    public SurCharges(JSONParser parser) {
      while (parser.nextToken() != System.JSONToken.END_OBJECT
        if (parser.getCurrentToken() == System.JSONToken.FIELD
          String text = parser.getText();
          if (parser.nextToken() != System.JSONToken.VALUE_NUL
            if (text == 'type') {
              type = parser.getText();
            } else if (text == 'description') {
              description = parser.getText();
            } else if (text == 'amount') {
              amount = parser.getDoubleValue();
            } else {
              System.debug(
                LoggingLevel.WARN,
                'SurCharges consuming unrecognized property: '
              );
              consumeObject(parser);
            }
          }
        }
      }
    }
  }
```

```

public class Names {
  public String type { get; set; }
  public String encoding { get; set; }
  public String value { get; set; }

  public Names(JSONParser parser) {
    while (parser.nextToken() != System.JSONToken.END_OBJECT
      if (parser.getCurrentToken() == System.JSONToken.FIELD
        String text = parser.getText();
        if (parser.nextToken() != System.JSONToken.VALUE_NUL
          if (text == 'type') {
            type = parser.getText();
          } else if (text == 'encoding') {
            encoding = parser.getText();
          } else if (text == 'value') {
            value = parser.getText();
          } else {
            System.debug(
              LoggingLevel.WARN,
              'Names consuming unrecognized property: ' + te
            );
            consumeObject(parser);
          }
        }
      }
    }
  }
}

public class ShipmentRateDetail {
  public String rateZone { get; set; }
  public Integer dimDivisor { get; set; }
  public Double fuelSurchargePercent { get; set; }
  public Double totalSurcharges { get; set; }
  public Double totalFreightDiscount { get; set; }
  public List<SurCharges> surCharges { get; set; }
  public String pricingCode { get; set; }
  public TotalBillingWeight totalBillingWeight { get; set; }
  public String currency_Z { get; set; } // in json: currenc
  public String rateScale { get; set; }

  public ShipmentRateDetail(JSONParser parser) {
    while (parser.nextToken() != System.JSONToken.END_OBJECT
      if (parser.getCurrentToken() == System.JSONToken.FIELD
        String text = parser.getText();
        if (parser.nextToken() != System.JSONToken.VALUE_NUL
          if (text == 'rateZone') {
            rateZone = parser.getText();
          } else if (text == 'dimDivisor') {
            dimDivisor = parser.getIntegerValue();
          } else if (text == 'fuelSurchargePercent') {
            fuelSurchargePercent = parser.getDoubleValue();
          } else if (text == 'totalSurcharges') {
            totalSurcharges = parser.getDoubleValue();
          } else if (text == 'totalFreightDiscount') {
            totalFreightDiscount = parser.getDoubleValue();
          } else if (text == 'surCharges') {
            surCharges = arrayOfSurCharges(parser);
          } else if (text == 'pricingCode') {
            pricingCode = parser.getText();
          } else if (text == 'totalBillingWeight') {
            totalBillingWeight = new TotalBillingWeight(pars
          } else if (text == 'currency') {
            currency_Z = parser.getText();
          } else if (text == 'rateScale') {
            rateScale = parser.getText();
          } else {
            System.debug(
              LoggingLevel.WARN,
              'ShipmentRateDetail consuming unrecognized pro
            );
            consumeObject(parser);
          }
        }
      }
    }
  }
}

public class PackageRateDetail {
  public String rateType { get; set; }
  public String ratedWeightMethod { get; set; }
  public Double baseCharge { get; set; }
  public Double netFreight { get; set; }
  public Double totalSurcharges { get; set; }
  public Double netFedExCharge { get; set; }
  public Double totalTaxes { get; set; }
  public Double netCharge { get; set; }
  public Double totalRebates { get; set; }
  public TotalBillingWeight billingWeight { get; set; }
  public Double totalFreightDiscounts { get; set; }
  public List<SurCharges> surcharges { get; set; }
  public String currency_Z { get; set; } // in json: currenc

  public PackageRateDetail(JSONParser parser) {
    while (parser.nextToken() != System.JSONToken.END_OBJECT
      if (parser.getCurrentToken() == System.JSONToken.FIELD
        String text = parser.getText();
        if (parser.nextToken() != System.JSONToken.VALUE_NUL
          if (text == 'rateType') {
            rateType = parser.getText();
          } else if (text == 'ratedWeightMethod') {
            ratedWeightMethod = parser.getText();
          } else if (text == 'baseCharge') {
            baseCharge = parser.getDoubleValue();
          } else if (text == 'netFreight') {
            netFreight = parser.getDoubleValue();
          } else if (text == 'totalSurcharges') {
            totalSurcharges = parser.getDoubleValue();
          } else if (text == 'netFedExCharge') {
            netFedExCharge = parser.getDoubleValue();
          } else if (text == 'totalTaxes') {
            totalTaxes = parser.getDoubleValue();
          } else if (text == 'netCharge') {
            netCharge = parser.getDoubleValue();
          } else if (text == 'totalRebates') {
            totalRebates = parser.getDoubleValue();
          } else if (text == 'billingWeight') {
            billingWeight = new TotalBillingWeight(parser);
          } else if (text == 'totalFreightDiscounts') {
            totalFreightDiscounts = parser.getDoubleValue();
          } else if (text == 'surcharges') {
            surcharges = arrayOfSurCharges(parser);
          } else if (text == 'currency') {
            currency_Z = parser.getText();
          } else {
            System.debug(
              LoggingLevel.WARN,
              'PackageRateDetail consuming unrecognized prop
            );
            consumeObject(parser);
          }
        }
      }
    }
  }
}

public class RateReplyDetails {
  public String serviceType { get; set; }
  public String serviceName { get; set; }
  public String packagingType { get; set; }
  public List<RatedShipmentDetails> ratedShipmentDetails { g
  public OperationalDetail operationalDetail { get; set; }
  public String signatureOptionType { get; set; }
  public ServiceDescription serviceDescription { get; set; }

  public RateReplyDetails(JSONParser parser) {
    while (parser.nextToken() != System.JSONToken.END_OBJECT
      if (parser.getCurrentToken() == System.JSONToken.FIELD
        String text = parser.getText();
        if (parser.nextToken() != System.JSONToken.VALUE_NUL
          if (text == 'serviceType') {
            serviceType = parser.getText();
          } else if (text == 'serviceName') {
            serviceName = parser.getText();
          } else if (text == 'packagingType') {
            packagingType = parser.getText();
          } else if (text == 'ratedShipmentDetails') {
            ratedShipmentDetails = arrayOfRatedShipmentDetai
          } else if (text == 'operationalDetail') {
            operationalDetail = new OperationalDetail(parser
          } else if (text == 'signatureOptionType') {
            signatureOptionType = parser.getText();
          } else if (text == 'serviceDescription') {
            serviceDescription = new ServiceDescription(pars
          } else {
            System.debug(
              LoggingLevel.WARN,
              'RateReplyDetails consuming unrecognized prope
            );
            consumeObject(parser);
          }
        }
      }
    }
  }
}

```

```
public class TotalBillingWeight {
  public String units { get; set; }
  public Double value { get; set; }

  public TotalBillingWeight(JSONParser parser) {
    while (parser.nextToken() != System.JSONToken.END_OBJECT
      if (parser.getCurrentToken() == System.JSONToken.FIELD
        String text = parser.getText();
        if (parser.nextToken() != System.JSONToken.VALUE_NUL
          if (text == 'units') {
            units = parser.getText();
          } else if (text == 'value') {
            value = parser.getDoubleValue();
          } else {
            System.debug(
              LoggingLevel.WARN,
              'TotalBillingWeight consuming unrecognized pro
            );
            consumeObject(parser);
          }
        }
      }
    }
  }
}

public class Output {
  public List<Alerts> alerts { get; set; }
  public List<RateReplyDetails> rateReplyDetails { get; set;
  public String quoteDate { get; set; }
  public Boolean encoded { get; set; }

  public Output(JSONParser parser) {
    while (parser.nextToken() != System.JSONToken.END_OBJECT
      if (parser.getCurrentToken() == System.JSONToken.FIELD
        String text = parser.getText();
        if (parser.nextToken() != System.JSONToken.VALUE_NUL
          if (text == 'alerts') {
            alerts = arrayOfAlerts(parser);
          } else if (text == 'rateReplyDetails') {
            rateReplyDetails = arrayOfRateReplyDetails(parse
          } else if (text == 'quoteDate') {
            quoteDate = parser.getText();
          } else if (text == 'encoded') {
            encoded = parser.getBooleanValue();
          } else {
            System.debug(
              LoggingLevel.WARN,
              'Output consuming unrecognized property: ' + t
            );
            consumeObject(parser);
          }
        }
      }
    }
  }
}

public class Alerts {
  public String code { get; set; }
```

```
    public String message { get; set; }
    public String alertType { get; set; }

    public Alerts(JSONParser parser) {
      while (parser.nextToken() != System.JSONToken.END_OBJECT
        if (parser.getCurrentToken() == System.JSONToken.FIELD
          String text = parser.getText();
          if (parser.nextToken() != System.JSONToken.VALUE_NUL
            if (text == 'code') {
              code = parser.getText();
            } else if (text == 'message') {
              message = parser.getText();
            } else if (text == 'alertType') {
              alertType = parser.getText();
            } else {
              System.debug(
                LoggingLevel.WARN,
                'Alerts consuming unrecognized property: ' + t
              );
              consumeObject(parser);
            }
          }
        }
      }
    }
  }

  public class SurCharges_Z {
    public String type { get; set; }
    public String description { get; set; }
    public String level { get; set; }
    public Double amount { get; set; }

    public SurCharges_Z(JSONParser parser) {
      while (parser.nextToken() != System.JSONToken.END_OBJECT
        if (parser.getCurrentToken() == System.JSONToken.FIELD
          String text = parser.getText();
          if (parser.nextToken() != System.JSONToken.VALUE_NUL
            if (text == 'type') {
              type = parser.getText();
            } else if (text == 'description') {
              description = parser.getText();
            } else if (text == 'level') {
              level = parser.getText();
            } else if (text == 'amount') {
              amount = parser.getDoubleValue();
            } else {
              System.debug(
                LoggingLevel.WARN,
                'SurCharges_Z consuming unrecognized property:
              );
              consumeObject(parser);
            }
          }
        }
      }
    }
  }

  public static FedExShippingRateResponse parse(String json) {
    System.JSONParser parser = System.JSON.createParser(json);
    return new FedExShippingRateResponse(parser);
  }

  public static void consumeObject(System.JSONParser parser) {
    Integer depth = 0;
    do {
      System.JSONToken curr = parser.getCurrentToken();
      if (
        curr == System.JSONToken.START_OBJECT ||
        curr == System.JSONToken.START_ARRAY
      ) {
        depth++;
      } else if (
        curr == System.JSONToken.END_OBJECT ||
        curr == System.JSONToken.END_ARRAY
      ) {
        depth--;
      }
    } while (depth > 0 && parser.nextToken() != null);
  }

  private static List<Alerts> arrayOfAlerts(System.JSONParser
    List<Alerts> res = new List<Alerts>();
    if (p.getCurrentToken() == null)
      p.nextToken();
    while (p.nextToken() != System.JSONToken.END_ARRAY) {
      res.add(new Alerts(p));
    }
    return res;
  }

  private static List<RatedPackages_Z> arrayOfRatedPackages_Z(
    System.JSONParser p
  ) {
    List<RatedPackages_Z> res = new List<RatedPackages_Z>();
    if (p.getCurrentToken() == null)
      p.nextToken();
    while (p.nextToken() != System.JSONToken.END_ARRAY) {
      res.add(new RatedPackages_Z(p));
    }
    return res;
  }

  private static List<SurCharges> arrayOfSurCharges(System.JSO
    List<SurCharges> res = new List<SurCharges>();
    if (p.getCurrentToken() == null)
      p.nextToken();
    while (p.nextToken() != System.JSONToken.END_ARRAY) {
      res.add(new SurCharges(p));
    }
    return res;
  }

  private static List<RatedShipmentDetails_Z> arrayOfRatedShip
    System.JSONParser p
  ) {
    List<RatedShipmentDetails_Z> res = new List<RatedShipmentD
    if (p.getCurrentToken() == null)
      p.nextToken();
    while (p.nextToken() != System.JSONToken.END_ARRAY) {
      res.add(new RatedShipmentDetails_Z(p));
    }
    return res;
  }

  private static List<Names> arrayOfNames(System.JSONParser p)
    List<Names> res = new List<Names>();
    if (p.getCurrentToken() == null)
      p.nextToken();
    while (p.nextToken() != System.JSONToken.END_ARRAY) {
      res.add(new Names(p));
    }
    return res;
  }

  private static List<RatedShipmentDetails> arrayOfRatedShipme
    System.JSONParser p
  ) {
    List<RatedShipmentDetails> res = new List<RatedShipmentDet
    if (p.getCurrentToken() == null)
      p.nextToken();
    while (p.nextToken() != System.JSONToken.END_ARRAY) {
      res.add(new RatedShipmentDetails(p));
    }
    return res;
  }

  private static List<RateReplyDetails> arrayOfRateReplyDetail
    System.JSONParser p
  ) {
    List<RateReplyDetails> res = new List<RateReplyDetails>();
    if (p.getCurrentToken() == null)
      p.nextToken();
    while (p.nextToken() != System.JSONToken.END_ARRAY) {
      res.add(new RateReplyDetails(p));
    }
    return res;
  }

  private static List<SurCharges_Z> arrayOfSurCharges_Z(System
    List<SurCharges_Z> res = new List<SurCharges_Z>();
    if (p.getCurrentToken() == null)
      p.nextToken();
    while (p.nextToken() != System.JSONToken.END_ARRAY) {
      res.add(new SurCharges_Z(p));
    }
    return res;
  }

  private static List<RatedPackages> arrayOfRatedPackages(Syst
    List<RatedPackages> res = new List<RatedPackages>();
    if (p.getCurrentToken() == null)
      p.nextToken();
    while (p.nextToken() != System.JSONToken.END_ARRAY) {
      res.add(new RatedPackages(p));
    }
    return res;
  }
}
```