Zenkraft Service

C# Library to access Zenkraft API

Documentation

# Zenkraft Library

Zenkraft Library is the main .dll through which developers can get access to the Zenkraft API. It contains all the methods such as **rate**, **ship**, **pickup**…

## Zenkraft Service

After the Zenkraft account is created, devs can use the **ApiKey** to initiate the **ZenkraftService** and access all the methods provided by Zenkraft.

### Constructors

1. ZkService zkService = new ZkService (); //To be used to create the Zenkraft Account
2. ZkService zkService = new ZkService (“your\_Api\_key\_here”); //To be used after obtaining the Api key

### Properties

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| ApiKey | string | The API Key for the account used |
| Test | Boolean | Is set to True if user wants to use the carriers’ test servers. Default: false |

### Sample Code

ZkService zkService = new ZkService (“your\_Api\_key\_here”);

zkService.Test = true; //If you want to test the methods, set it to true. Default is false.

### Methods

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Arguments** | **Return Type** | **Description** |
| GetCarriers()  *Sample:* string carriers = zkMetadataService.GetCarriers(); |  | string | Retrieves a list of Carriers supported by Zenkraft as a JSON String |
| GetCarrierDetails()  *Sample:* CarrierDetail fedexDetails = zkMetadataService.GetCarrierDetails("fedex"); | string carrierName | [CarrierDetail](#_CarrierDetail) | Retrieves the details for one specific carrier (Label Types, Package Types…) |
| AddCarrierAccount()  *Sample:* string carrierJson = "{\r\n \"shipping\_account\": {\r\n \"name\": \"My FedEx test account\",\r\n \"carrier\": \"fedex\",\r\n \"country\": \"US\",\r\n \"auth\": {\r\n\t \"account\_number\": \"630156343\",\r\n\t \"meter\_number\": \"119147957\",\r\n\t \"key\": \"SH2p9UyUWgbhdeAg\",\r\n\t \"password\": \"v6S5YjRhLEvSTdSwfjOY2uzyW\"\r\n }\r\n }\r\n}";  CarrierAccount carrierAccount = zkService.AddCarrierAccount(carrierJson); | string carrierJson | [CarrierAccount](#_CarrierAccount) | Creates a Carrier Account |
| GetRates()  *Sample:* zkService.Test = true;  Shipment shipment = new Shipment();  shipment.carrier = "fedex";  shipment.type = "outbound";  shipment.dim\_units = "CM";  shipment.weight\_units = "KG";  shipment.currency = "USD";  shipment.shipping\_account = fedexShippingAccountId;  Address sender = new Address();  sender.street1 = "10 FED EX PKWY";  sender.city = "COLLIERVILLE";  sender.state = "TN";  sender.postal\_code = "38017";  sender.country = "US";  shipment.sender = sender;  Address recipient = new Address();  recipient.street1 = "525 S. Lexington Ave";  recipient.city = "Burlington";  recipient.state = "NC";  recipient.postal\_code = "27215";  recipient.country = "US";  shipment.recipient = recipient;  Package package1 = new Package();  package1.weight = 33;  package1.value = 4;  package1.length = 4;  package1.width = 4;  package1.height = 4;  Package package2 = new Package();  package2.weight = 33;  package2.value = 4;  package2.length = 4;  package2.width = 4;  package2.height = 4;  List<Package> packages = new List<Package>() { package1, package2 };  shipment.packages = packages;  string ratesResponse = zkService.GetRates(shipment); | [Shipment](#_Shipment) shipment | string ratesJSON | Gets a list of all possible rates for desired shipment |
| TrackShipment()  *Sample:* string trackingNumber = "122816215025810";  CarrierAccount fedexCarrierAccount = new CarrierAccount();  fedexCarrierAccount.id = fedexShippingAccountId;  fedexCarrierAccount.carrier = "fedex";  zkService.Test = true;  ShipmentStatus shipmentStatus = zkService.TrackShipment(trackingNumber, fedexCarrierAccount); | string trackingNumber, [CarrierAccount](#_CarrierAccount) carrierAccount | [ShipmentStatus](#_ShipmentStatus) | Returns the status of the shipment |
| Cancel()  *Sample:* string trackingNumber = "794626099895";  CarrierAccount fedexCarrierAccount = new CarrierAccount();  fedexCarrierAccount.id = fedexShippingAccountId;  fedexCarrierAccount.carrier = "fedex";  zkService.Test = true;  ShipmentCancellationResponse cancellationResponse = zkService.Cancel(trackingNumber, fedexCarrierAccount); | string trackingNumber, [CarrierAccount](#_CarrierAccount) carrierAccount | [ShipmentCancellationResponse](#_ShipmentCancellationResponse) | Returns a response (success/error) along with a message |
| Ship()  *Sample:* zkService.Test = true;  Shipment shipment = new Shipment();  shipment.carrier = "fedex";  shipment.type = "outbound";  shipment.dim\_units = "CM";  shipment.weight\_units = "KG";  shipment.currency = "USD";  shipment.shipping\_account = fedexShippingAccountId;  shipment.service = "fedex\_ground";  shipment.ship\_date = new DateTime(2019, 7, 20);  shipment.label\_type = "PDF";  shipment.packaging = "your\_packaging";  Address sender = new Address();  sender.street1 = "10 FED EX PKWY";  sender.city = "COLLIERVILLE";  sender.state = "TN";  sender.postal\_code = "38017";  sender.country = "US";  sender.company = "Zk sender";  sender.phone = "1234567890";  sender.name = "Zack King";  sender.email = "zk@zktest.com";  shipment.sender = sender;  Address recipient = new Address();  recipient.street1 = "525 S. Lexington Ave";  recipient.city = "Burlington";  recipient.state = "NC";  recipient.postal\_code = "27215";  recipient.country = "US";  recipient.company = "Zk recipient";  recipient.phone = "1234567898";  recipient.name = "John Doe";  recipient.email = "jd@jdtest.com";  shipment.recipient = recipient;  Package package1 = new Package();  package1.weight = 33;  package1.value = 4;  package1.length = 4;  package1.width = 4;  package1.height = 4;  package1.description = "Electronics";    Package package2 = new Package();  package2.weight = 33;  package2.value = 4;  package2.length = 4;  package2.width = 4;  package2.height = 4;  package2.description = "Electronics";  List<Package> packages = new List<Package>() { package1 };  ZkField reference1 = new ZkField();  reference1.type = "invoice\_number";  reference1.value = "INV-12345";  List<ZkField> references = new List<ZkField>() { reference1 };  shipment.packages = packages;  shipment.references = references;  Shipment submittedShipment = zkService.Ship(shipment); | [Shipment](#_Shipment) shipment | [Shipment](#_Shipment) | Returns the shipment object with Tracking number |
| Pickup()  *Sample:*  ZkTime zkTime = new ZkTime();  zkTime.date = "2019-07-18";  zkTime.ready\_time = "15:00";  zkTime.close\_time = "18:00";  Address pickupLocation = new Address();  pickupLocation.street1 = "207 Continental Drive";  pickupLocation.city = "Newark";  pickupLocation.state = "DE";  pickupLocation.postal\_code = "19720";  pickupLocation.country = "US";  pickupLocation.email = "tstemail@tst.com";  pickupLocation.phone = "293843995";  pickupLocation.name = "Bob Jones";  pickupLocation.company = "Tesla Inc";  pickupLocation.residential = false;  pickupLocation.location\_type = "front";  Shipment shipment = new Shipment();    Address recipient = new Address();  recipient.street1 = "525 S. Lexington Ave";  recipient.city = "Burlington";  recipient.state = "NC";  recipient.postal\_code = "27215";  recipient.country = "US";  recipient.company = "Zk recipient";  recipient.phone = "1234567898";  recipient.name = "John Doe";  recipient.email = "jd@jdtest.com";    shipment.recipient = recipient;  Package package1 = new Package();  package1.weight = 33;  package1.value = 4;  package1.length = 4;  package1.width = 4;  package1.height = 4;  shipment.packages = new List<Package>() { package1 };  Pickup pickupTest = new Pickup();  pickupTest.shipments = new List<Shipment>() { shipment };  pickupTest.carrier = "fedex";  pickupTest.shipping\_account = fedexShippingAccountId;  pickupTest.time = zkTime;  pickupTest.location = pickupLocation;  pickupTest.test = true;  pickupTest.debug = true;  pickupTest.dim\_units = "IN";  pickupTest.weight\_units = "LB";  pickupTest.description = "TEST ENVIRONMENT -PLEASE DO NOT PROCESS PICKUP";  List<Int64> shipments = new List<Int64>() { 0909090909090, 090909090910 };    zkService.Test = true;  Pickup pickupResponse = zkService.Pickup(pickupTest); | [Pickup](#_Pickup) pickup | [Pickup](#_Pickup) | Returns true if pickup is scheduled. False otherwise |

#### Objects

##### CarrierAccount

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| name | string | Gets the name of the carrier account |
| id | Int64 | Gets the Id of the carrier account |
| carrier | string | Gets the name of the carrier |
| dim\_units | string | Gets the Dimension Units (i.e. “IN”) |
| weight\_units | string | Gets the Weight Units (i.e. “KG”) |
| currency | string | Gets the name of the currency (i.e. “USD”) |
| auth | string (Json) | Gets a JSON object representing the authentication for the carrier (i.e. {  "account\_number": "XXXXXXXX",  "meter\_number": "XXXXXXXX",  "key": "XXXXXXXXXXXXXXXXXXXXXXXX",  "password": "XXXXXXXXXXXXXXXXXXXXXXXX"  }) |

##### ShipmentStatus

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| carrier | string | Gets the Carrier Name |
| status | string | Gets the Status Text |
| checkpoints | List<[Checkpoint](#_Checkpoint)>() | Gets the List of checkpoints for the current shipment |
| delivered | boolean | Returns whether the shipment is delivered |
| packages | List<[PackageCheckpoint](#_PackageCheckpoint)>() | Returns a list of package checkpoints |
| shipment | [Shipment](#_Shipment_1) | Returns the shipment |
| tracking\_stage | string | Returns the Tracking Stage of the shipment |

##### Checkpoint

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| description | string | Gets the Description of the checkpoint |
| location | [Location](#_Location) | Gets the location object of the checkpoint |
| time | DateTime | Gets the time when the shipment reached/will reach the checkpoint |
| status | string | Gets the status of the checkpoint |
| tracking\_stage | string | Gets the tracking stage |

##### PackageCheckpoint

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| checkpoints | List<[Checkpoint](#_Checkpoint)>() | Gets the list of checkpoints for this package |
| tracking\_number | string | Gets the tracking number of the package |

##### Location

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| city | string | Gets the city |
| country | string | Gets the Country |
| postal\_code | string | Gets the Postal Code |
| state | string | Gets the State |

##### ShipmentCancellationResponse

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| success | Boolean | Returns true if shipment was cancelled |
| message | string | Returns the message from API |

##### CarrierDetail

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| auth | List<[ZkField](#_ZkField)>() | Returns the list of auth fields for the current Carrier |
| billing\_fields | List<[ZkField](#_ZkField)>() | Returns the list of billing fields for the current Carrier |
| custom\_fields | List<[ZkField](#_ZkField)>() | Returns the list of custom fields for the current Carrier |
| customs | List<[ZkField](#_ZkField)>() | Returns the list of customs fields for the current Carrier |
| display\_name | string | Returns the Display name of the carrier |
| label\_types | List<[ZkField](#_ZkField)>() | Returns the list of label types supported by the current Carrier |
| multi\_package | Boolean | Returns whether the carrier supports multiple packages |
| notifications | List<[ZkField](#_ZkField)>() | Returns notification methods |
| package\_special\_services | List<[ZkField](#_ZkField)>() | Returns the list of package special services supported by the current Carrier |
| packaging\_types | List<[ZkField](#_ZkField)>() | Returns the list of packaging types supported by the current Carrier |
| rate | [CarrierRate](#_CarrierRate) | Returns rate fields for this carrier |
| references | List<[ZkField](#_ZkField)>() | Returns the list of references for the current Carrier |
| servicetype\_availability | Boolean | Returns true/false if service types are available |
| service\_types | List<[ZkField](#_ZkField)>() | Returns the list of service types supported by this Carrier |
| special\_services | List<[ZkField](#_ZkField)>() | Returns the list of special services supported by this Carrier |

##### ZkField

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| name | string | Name of the field |
| required | boolean | Returns whether the field is required or not |
| slug | string | Returns the slug of the field |
| type | string | Returns the type of the field |
| ui | boolean | Returns whether this field should appear on the ui or not |
| collection | List<[ZkField](#_ZkField)>() | Returns a collection of ZkFields |
| code | string | Returns the code of the field (similar to slug) |
| maxlength | Int | Returns the maxLength of the field |
| minlength | Int | Returns the minLength of the field |
| node | string | Returns the node to which the field belongs |
| sizes | List<string> | Returns a list of sizes |
| required\_weight | Boolean | Returns whether the weight is required or not |
| value | string | Returns the value of the field |

##### Shipment

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| carrier | string | Returns the name of the carrier for this shipment |
| costs | [Cost](#_Cost) | Returns the cost properties |
| currency | string | Returns the currency used for the costs |
| dim\_units | string | Returns the Dimension units |
| debug | boolean | Returns true/false if debugging is enabled |
| id | string | Returns the id of the shipment |
| label\_type | string | Returns the Format of the label returned |
| packages | List<[Package](#_Package)> | Returns a list of packages in the shipment |
| recipient | [Address](#_Address) | Returns the Recipient of the Shipment |
| sender | [Address](#_Address) | Returns the sender of the shipment |
| service | string | Returns the service name |
| ship\_date | DateTime | Returns the ship date |
| shipping\_account | Int64 | Returns the shipping account used to ship |
| special\_services | List<[ZkField](#_ZkField)>() | Returns a list of special services |
| tracking\_number | String | Returns the tracking number of the shipment |
| type | String | Returns the type of the shipment (return or normal) |
| weight\_units | String | Return the weight units used |
| references | List<[ZkField](#_ZkField)>() | Returns the list of references |
| packaging | string | Returns the packaging type of the shipment |
| test | boolean | Returns if we’re using the test servers for this shipment |
| shipping\_documents | List<[ZkField](#_ZkField)>() | Returns the list of shipping documents |
| additional\_fields | string | Represents a Json object with keys/values that are custom for specific carriers. These will be added to the shipment object and sent to the /ship request |

##### Cost

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| base\_charge | string | Returns the base charge amount |
| discounts | string | Returns the discounts amount |
| net\_charge | string | Returns the net charge amount |
| surcharges | string | Returns the surcharges amount |
| taxes | string | Returns the taxes amount |
| currency | string | Returns the currency |

##### Package

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| height | int | Returns the height of the package |
| label | string | Returns the label of the package as Base64 String |
| label\_type | string | Returns the label format |
| length | int | Returns the length of the package |
| tracking\_number | string | Returns the tracking number of the package |
| value | string | Returns the value (money) of the package |
| weight | string | Returns the weight of the package |
| width | int | Returns the width of the package |
| description | string | Returns the Description of the package |
| carrier\_specific | string | Returns a JSON string of carrier specific fields |

##### Address

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| city | string | Returns the city |
| company | string | Returns the company |
| country | string | Returns the country |
| email | string | Returns the email address |
| name | string | Returns the name |
| phone | string | Returns the Phone number |
| postal\_code | string | Returns the Postal Code |
| state | string | Returns the State |
| street1 | string | Returns the Street |
| residential | boolean | Returns true if address is residential |
| location\_type | string | Returns the location type |

##### CarrierRate

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| fields | List<[ZkField](#_ZkField)>() | Gets the list of fields for the carrier rate |
| service\_type\_field | string | Gets the name of the service type field |

##### Pickup

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| carrier | string | Returns the carrier |
| shipping\_account | Int64 | Returns the shipping account |
| time | [ZkTime](#_ZkTime) | Returns the time object for pickup |
| location | [Address](#_Address) | Returns the address |
| shipments | List<[Shipment](#_Shipment_1)>() | Returns the shipment object |
| test | boolean | Returns whether this is a test call |
| debug | boolean | Returns whether debug is enabled or not |
| weight\_units | string | Returns the weight units for the shipments |
| dim\_units | string | Returns the dim units for the shipments |
| description | string | Returns the description of the pickup |
| confirmation\_number | string | Returns the pickup confirmation number |
| id | string | Returns the pickup id |
| pickup\_location\_id | string | Returns the pickup location Id |

##### ZkTime

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| date | string | Returns the date: format “2018-07-17” |
| ready\_time | string | Returns the ready time: format: “15:00” |
| close\_time | string | Returns the close time: format: “18:00” |

##### ZkSuccess

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| error\_message | string | Returns the error message if success is false |
| detail | string | Returns the error detail if any |
| success | boolean | Returns whether an api call succeeded or not |

## Dynamics 365 Sample

Our Service can be used in Dynamics 365 CE Plugins and workflows to integrate Zenkraft Api directly in your Customer engagement instance.

Please note that Initial set-up is required in order to get things up and running. The code in the sample below is meant to serve as a reference for Dynamics 365 Developers, not a ready-to-deploy code. This is an example

### Creating a return shipment from a case

#### Pre-requisites

1. Have a Shipment entity created (in our case: zk\_shipment)
2. Have a button on the Case entity to create a draft shipment
3. Have a workflow/plugin/JavaScript code setup to map all the info needed into the shipment entity (Sender info, recipient info…)
4. Have a Package entity created (in our case: zk\_package) with an N:1 relationship to the shipment
5. Have a button on the shipment to get Rates
6. Have a “Two-options” field on the shipment entity to specify whether it’s a return shipment or not.
7. Have Custom Actions that call the custom Workflow Activities so they can be called from JavaScript functions

#### Get Rates

##### Custom Activity to GetRates

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

using Microsoft.Xrm.Sdk.Workflow;

using Newtonsoft.Json;

using Newtonsoft.Json.Linq;

using System;

using System.Activities;

using System.Collections.Generic;

using System.Dynamic;

using System.Linq;

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

using ZenkraftService;

namespace ZenkraftWorkflows

{

public class GetRatesForShipment : CodeActivity

{

#region Input Parameters

[Input("Shipment Id")]

public InArgument<String> ShipmentId { get; set; }

#endregion

#region Output Parameters

[Output("Rates JSON")]

public OutArgument<string> RatesJSON { get; set; }

[Output("Success")]

public OutArgument<Boolean> Success { get; set; }

[Output("Error Message")]

public OutArgument<string> ErrorMessage { get; set; }

#endregion

protected override void Execute(CodeActivityContext executionContext)

{

try

{

//Create the tracing service

ITracingService tracingService = executionContext.GetExtension<ITracingService>();

//Create the context

IWorkflowContext context = executionContext.GetExtension<IWorkflowContext>();

IOrganizationServiceFactory serviceFactory = executionContext.GetExtension<IOrganizationServiceFactory>();

IOrganizationService service = serviceFactory.CreateOrganizationService(context.UserId);

tracingService.Trace("Starting Workflow: GetRates");

//get input params

string shipmentId = ShipmentId.Get<String>(executionContext);

tracingService.Trace("Retrieved Input Param: {0}", shipmentId);

string apiKey = "your\_api\_key\_goes\_here";

tracingService.Trace("Retrieved API Key: {0}", apiKey);

if (string.IsNullOrEmpty(apiKey))

{

Success.Set(executionContext, false);

ErrorMessage.Set(executionContext, "No configuration with API key was found.");

return;

}

//Get details from shipment

Guid shipmentGuid = Guid.Parse(shipmentId);

Entity shipmentEntity = service.Retrieve("zk\_shipment", shipmentGuid, new ColumnSet("zk\_carrierid", "zk\_shippingaccount", "zk\_returnshipment", "zk\_dimensionunits", "zk\_weightunits", "zk\_currencyid", "zk\_shipperstreet", "zk\_shippercity", "zk\_shipperstatecode", "zk\_shipperpostalcode", "zk\_shippercountrycode", "zk\_recipientstreet", "zk\_recipientcity", "zk\_recipientstatecode", "zk\_recipientpostalcode", "zk\_recipientcountrycode"));

tracingService.Trace("Successfully retrieved shipment entity...");

EntityReference carrierRef = (EntityReference)shipmentEntity.Attributes["zk\_carrierid"];

Entity carrierEntity = service.Retrieve("zk\_carrier", carrierRef.Id, new ColumnSet("zk\_code"));

CarrierAccount carrierAccount = new CarrierAccount();

carrierAccount.carrier = carrierEntity.Attributes["zk\_code"].ToString();

EntityReference shippingAccountRef = (EntityReference)shipmentEntity.Attributes["zk\_shippingaccount"];

Entity shippingAccountEntity = service.Retrieve("zk\_shippingaccount", shippingAccountRef.Id, new ColumnSet("zk\_spid"));

Int64 shipping\_account = Int64.Parse(shippingAccountEntity.Attributes["zk\_spid"].ToString());

bool returnShipment = (bool)shipmentEntity.Attributes["zk\_returnshipment"];

string type = "";

if (returnShipment)

type = "return";

else

type = "outbound";

string dim\_units = Common.getOptionSetLabel(service, ((OptionSetValue)shipmentEntity.Attributes["zk\_dimensionunits"]).Value, "zk\_dimensionunits", "zk\_shipment");

string weight\_units = Common.getOptionSetLabel(service, ((OptionSetValue)shipmentEntity.Attributes["zk\_weightunits"]).Value, "zk\_weightunits", "zk\_shipment");

EntityReference currencyRef = (EntityReference)shipmentEntity.Attributes["zk\_currencyid"];

Entity currencyEntity = service.Retrieve(currencyRef.LogicalName, currencyRef.Id, new ColumnSet("isocurrencycode"));

string currency = currencyEntity.Attributes["isocurrencycode"].ToString();

carrierAccount.tyoe = type;

carrierAccount.dim\_units = dim\_units;

carrierAccoint.weight\_units = weight\_units;

carrierAccount.currency = currency;

//Sender details

string sender\_street1 = shipmentEntity.Attributes["zk\_shipperstreet"].ToString();

string sender\_city = shipmentEntity.Attributes["zk\_shippercity"].ToString();

string sender\_state = shipmentEntity.Attributes["zk\_shipperstatecode"].ToString();

string sender\_postalcode = shipmentEntity.Attributes["zk\_shipperpostalcode"].ToString();

string sender\_country = shipmentEntity.Attributes["zk\_shippercountrycode"].ToString();

//recipient details

string recipient\_street1 = shipmentEntity.Attributes["zk\_recipientstreet"].ToString();

string recipient\_city = shipmentEntity.Attributes["zk\_recipientcity"].ToString();

string recipient\_state = shipmentEntity.Attributes["zk\_recipientstatecode"].ToString();

string recipient\_postalcode = shipmentEntity.Attributes["zk\_recipientpostalcode"].ToString();

string recipient\_country = shipmentEntity.Attributes["zk\_recipientcountrycode"].ToString();

EntityCollection packagesCollection = getPackagesForShipment(shipmentGuid, service, tracingService);

List<Package> packagesArray = new List<Package>();

if (packagesCollection != null)

{

if(packagesCollection.Entities.Count == 0)

{

Success.Set(executionContext, false);

ErrorMessage.Set(executionContext, "Couldn't find packages for that shipment.");

return;

}

else

{

foreach (Entity packageEntity in packagesCollection.Entities)

{

Package package = new Package();

package.weight = (decimal)packageEntity.Attributes["zk\_weightvalue"];

package.value = (decimal)packageEntity.Attributes["zk\_value"];

package.length = (decimal)packageEntity.Attributes["zk\_length"];

package.width = (decimal)packageEntity.Attributes["zk\_width"];

package.height = (decimal)packageEntity.Attributes["zk\_height"];

packagesArray.Add(package);

}

}

}

else

{

Success.Set(executionContext, false);

ErrorMessage.Set(executionContext, "Couldn't retrieve packages for shipment.");

return;

}

tracingService.Trace("Successfully retrieved all attributes from shipment and related packages...");

tracingService.Trace("Building Json object to send to API");

//Shipment details

Shipment shipmentObj = new Shipment();

shipmentObj.carrier = carrier;

shipmentObj.type = type;

shipmentObj.dim\_units = dim\_units;

shipmentObj.weight\_units = weight\_units;

shipmentObj.currency = currency;

shipmentObj.shipping\_account = shipping\_account;

//sender

Address senderObj = new Address();

senderObj.street1 = sender\_street1;

senderObj.city = sender\_city;

senderObj.state = sender\_state;

senderObj.postal\_code = sender\_postalcode;

senderObj.country = sender\_country;

//recipient

Address recipientObj = new Address();

recipientObj.street1 = recipient\_street1;

recipientObj.city = recipient\_city;

recipientObj.state = recipient\_state;

recipientObj.postal\_code = recipient\_postalcode;

recipientObj.country = recipient\_country;

shipmentObj.sender = senderObj;

shipmentObj.recipient = recipientObj;

shipmentObj.packages = packagesArray;

shipmentObj.shipping\_account = 1233423432432 //Insert your shipping Account Id here

ZkService zkService = new ZkService(apiKey);

string result = zkService.GetRates(shipmentObj);

Success.Set(executionContext, true);

RatesJSON.Set(executionContext, result);

}

catch (Exception ex)

{

Success.Set(executionContext, false);

string errorMsg = "Failed to get Rates.";

if (ex != null)

errorMsg = ex.ToString();

ErrorMessage.Set(executionContext, errorMsg);

throw new InvalidWorkflowException(errorMsg);

}

}

private EntityCollection getPackagesForShipment(Guid shipmentId, IOrganizationService \_service, ITracingService \_tracingService)

{

\_tracingService.Trace("Entered Method 'getPackagesForShipment'");

QueryExpression qe = new QueryExpression("zk\_package");

qe.ColumnSet = new ColumnSet("zk\_weightvalue", "zk\_value", "zk\_length", "zk\_width", "zk\_height");

ConditionExpression condition1 = new ConditionExpression("statecode", ConditionOperator.Equal, 0); //Active

ConditionExpression condition2 = new ConditionExpression("zk\_shipmentid", ConditionOperator.Equal, shipmentId);

qe.Criteria.AddCondition(condition1);

qe.Criteria.AddCondition(condition2);

EntityCollection ec = \_service.RetrieveMultiple(qe);

return ec;

}

}

}

##### Rates Web Resource (To open when “get Rates” is clicked)

###### HTML

|  |
| --- |
| <!DOCTYPE html> |
|  |  |
|  | <html lang="en" xmlns="http://www.w3.org/1999/xhtml"> |
|  | <head> |
|  | <meta charset="utf-8" /> |
|  | <title>Zenkraft - Get Shipment Rates</title> |
|  |  |
|  | <link rel="stylesheet" type="text/css" href="zk\_bootstrap" /> |
|  | <script type="text/javascript" src="ClientGlobalContext.js.aspx"></script> |
|  | <script type="text/javascript" src="zk\_jquery.js"></script> |
|  | <script type="text/javascript" src="zk\_bootstrap.js"></script> |
|  | <script type="text/javascript" src="zk\_getrates.js"></script> |
|  | </head> |
|  | <body> |
|  | <!-- Modal --> |
|  | <div id="loadingModal" class="modal fade" role="dialog" > |
|  | <div class="modal-dialog"> |
|  |  |
|  | <!-- Modal content--> |
|  | <div class="modal-content"> |
|  | <div class="modal-header"> |
|  | <h4 class="modal-title">Please Wait...</h4> |
|  | <br /> |
|  |  |
|  | <img src="zk\_loadinggif" id="loading" alt="Loading..." height="42" width="42" style="margin:0 auto;"> |
|  | </div> |
|  |  |
|  | </div> |
|  |  |
|  | </div> |
|  | </div> |
|  | <div style="position:relative"> |
|  | <div id="tableContainer"></div> |
|  | <br /> |
|  | <button id="submitShipment" class="btn btn-primary" style="display:none; position:absolute; bottom:0; right: 10px;" onclick="submitShipment()">Submit</button> |
|  | </div> |
|  | </body> |
|  | </html> |

###### JavaScript

|  |
| --- |
| window.onload = onLoad; |
|  |  |
|  |  |
|  | function onLoad() |
|  | { |
|  | showLoading(true); |
|  | var shipmentId = getShipmentId(); |
|  |  |
|  |  |
|  |  |
|  | if (shipmentId != "") { |
|  | var parameters = {}; |
|  | parameters.shipmentid = shipmentId; |
|  |  |
|  | var req = new XMLHttpRequest(); |
|  | req.open("POST", Xrm.Page.context.getClientUrl() + "/api/data/v8.2/zk\_getrates", true); |
|  | req.setRequestHeader("OData-MaxVersion", "4.0"); |
|  | req.setRequestHeader("OData-Version", "4.0"); |
|  | req.setRequestHeader("Accept", "application/json"); |
|  | req.setRequestHeader("Content-Type", "application/json; charset=utf-8"); |
|  | req.onreadystatechange = function () { |
|  | if (this.readyState === 4) { |
|  | req.onreadystatechange = null; |
|  | if (this.status === 200) { |
|  |  |
|  | var results = JSON.parse(this.response); |
|  | var getRatesSucceeded = results.success; |
|  | if (getRatesSucceeded) { |
|  | if (results.ratesjson) { |
|  |  |
|  | var ratesResult = {}; |
|  | try { |
|  | ratesResult = JSON.parse(results.ratesjson); |
|  | } |
|  | catch(e) |
|  | { |
|  | ratesResult = {}; |
|  | } |
|  |  |
|  | if (ratesResult.rates) { |
|  | if (ratesResult.rates.length > 0) { |
|  | createTableFromRates(ratesResult.rates); |
|  | } |
|  | else { |
|  | alert("No rates were returned...make sure the addresses are correct."); |
|  | showLoading(false); |
|  | } |
|  | } |
|  | else { |
|  | alert("No rates were returned...make sure the addresses are correct."); |
|  | showLoading(false); |
|  | } |
|  | } |
|  | else { |
|  | alert("An error occured while getting rates. Rates returned null"); |
|  | showLoading(false); |
|  | } |
|  | } |
|  | else { |
|  | alert("An error occured while getting rates. " + results.errormessage); |
|  | showLoading(false); |
|  | } |
|  | } else { |
|  | Xrm.Utility.alertDialog("An error occured while getting rates. " + this.statusText); |
|  | showLoading(false); |
|  | } |
|  | } |
|  | }; |
|  | req.send(JSON.stringify(parameters)); |
|  | } |
|  | else { |
|  | alert("Could not retrieve shipment ID. Please try again later.") |
|  | showLoading(false); |
|  | } |
|  |  |
|  |  |
|  | } |
|  |  |
|  |  |
|  |  |
|  | function createTableFromRates(rates) { |
|  |  |
|  | // EXTRACT VALUE FOR HTML HEADER. |
|  | var col = []; |
|  | for (var i = 0; i < rates.length; i++) { |
|  | for (var key in rates[i]) { |
|  | if (col.indexOf(key) === -1) { |
|  | col.push(key); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | // CREATE DYNAMIC TABLE. |
|  | var table = document.createElement("table"); |
|  | table.style.border = "1px solid black"; |
|  | table.className = "table table-bordered"; |
|  | table.setAttribute("id", "ratesTable"); |
|  |  |
|  | // CREATE HTML TABLE HEADER ROW USING THE EXTRACTED HEADERS ABOVE. |
|  |  |
|  | var tr = table.insertRow(-1); // TABLE ROW. |
|  |  |
|  | for (var i = 0; i < col.length; i++) { |
|  | var th = document.createElement("th"); // TABLE HEADER. |
|  |  |
|  | if (i != 0) |
|  | th.innerHTML = getHeaderDisplayName(col[i]); |
|  |  |
|  | tr.appendChild(th); |
|  | } |
|  |  |
|  | // ADD JSON DATA TO THE TABLE AS ROWS. |
|  | for (var i = 0; i < rates.length; i++) { |
|  |  |
|  | tr = table.insertRow(-1); |
|  |  |
|  | var tabCellChk = tr.insertCell(-1); |
|  | tabCellChk.innerHTML = "<input type='radio' name='rateRadio' class='form-control' value='" + rates[i]["service\_type"] + "'>"; |
|  |  |
|  | for (var j = 1; j < col.length; j++) { |
|  | var tabCell = tr.insertCell(-1); |
|  | tabCell.innerHTML = rates[i][col[j]]; |
|  | } |
|  | } |
|  |  |
|  | // FINALLY ADD THE NEWLY CREATED TABLE WITH JSON DATA TO A CONTAINER. |
|  | var divContainer = document.getElementById("tableContainer"); |
|  | divContainer.innerHTML = ""; |
|  | divContainer.appendChild(table); |
|  |  |
|  | showLoading(false); |
|  | //SHow submit button |
|  | $("#submitShipment")[0].style.display = "block"; |
|  | } |
|  |  |
|  | function submitShipment() { |
|  |  |
|  | var shipmentId = getShipmentId(); |
|  |  |
|  | //get selected radio button id --> Service type |
|  | var selectedServiceType = $("input[name=rateRadio]:checked").val(); |
|  |  |
|  | if (selectedServiceType) { |
|  | showLoading(true); |
|  | var parameters = {}; |
|  | parameters.shipmentid = shipmentId; |
|  | parameters.servicetype = selectedServiceType; |
|  |  |
|  | var req = new XMLHttpRequest(); |
|  | req.open("POST", Xrm.Page.context.getClientUrl() + "/api/data/v8.2/zk\_submitshipment", true); |
|  | req.setRequestHeader("OData-MaxVersion", "4.0"); |
|  | req.setRequestHeader("OData-Version", "4.0"); |
|  | req.setRequestHeader("Accept", "application/json"); |
|  | req.setRequestHeader("Content-Type", "application/json; charset=utf-8"); |
|  | req.onreadystatechange = function () { |
|  | if (this.readyState === 4) { |
|  | req.onreadystatechange = null; |
|  | if (this.status === 200) { |
|  | var results = JSON.parse(this.response); |
|  | var shipmentUpdated = results.success; |
|  |  |
|  | if (shipmentUpdated) { |
|  | showLoading(false); |
|  | window.close(); |
|  | } |
|  | else { |
|  | var errorMessage = ""; |
|  | if (results.errormessage) |
|  | errorMessage = results.errormessage; |
|  |  |
|  | alert("An error occured while submitting the shipment: " + errorMessage); |
|  | showLoading(false); |
|  | } |
|  | } else { |
|  | alert("An error occured while submitting the shipment. " + this.statusText); |
|  | showLoading(false); |
|  | } |
|  | } |
|  | }; |
|  | req.send(JSON.stringify(parameters)); |
|  |  |
|  |  |
|  |  |
|  | } |
|  | else { |
|  | alert("Please select a service type before submitting..."); |
|  |  |
|  | } |
|  |  |
|  |  |
|  |  |
|  | } |
|  |  |
|  | function getShipmentId() { |
|  |  |
|  | var shipmentId = ""; |
|  | var query = GetGlobalContext().getQueryStringParameters(); |
|  | if (query != null && query.Data != null) { |
|  | var parameters = ParseData(query.Data); |
|  | shipmentId = parameters.entityId; |
|  | } |
|  |  |
|  | return shipmentId; |
|  | } |
|  |  |
|  | function ParseData(query) { |
|  | var result = {}; |
|  |  |
|  | if (typeof query == "undefined" || query == null) { |
|  | return result; |
|  | } |
|  |  |
|  | var queryparts = query.split("|"); |
|  | for (var i = 0; i < queryparts.length; i++) { |
|  | var params = queryparts[i].split("="); |
|  | result[params[0]] = params.length > 1 ? params[1] : null; |
|  | } |
|  | return result; |
|  | } |
|  |  |
|  | function cleanId(sampleId) { |
|  | if (sampleId) { |
|  | return sampleId.replace('{', '').replace('}', ''); |
|  | } |
|  | else { |
|  | alert("Can't clean empty Id"); |
|  | } |
|  | } |
|  |  |
|  | function isValueInArray(\_value, \_array) { |
|  | exists = false; |
|  |  |
|  | if (\_array.indexOf(\_value) != -1) |
|  | exists = true; |
|  |  |
|  | return exists; |
|  | } |
|  |  |
|  | function getHeaderDisplayName(key) { |
|  | var displayName = ""; |
|  | switch (key) { |
|  | case "service\_name": |
|  | displayName = "Service"; |
|  | break; |
|  | case "carrier": |
|  | displayName = "Carrier"; |
|  | break; |
|  | case "currency": |
|  | displayName = "Currency"; |
|  | break; |
|  | case "total\_cost": |
|  | displayName = "Total Cost"; |
|  | break; |
|  |  |
|  | } |
|  |  |
|  | return displayName; |
|  | } |
|  |  |
|  | function showLoading(show) { |
|  | if (show) { |
|  | // $('#loadingModal').modal('show'); |
|  | $("#loadingModal").modal({ backdrop: "static" }); |
|  | $('#submitShipment').prop('disabled', true); |
|  |  |
|  | } |
|  | else { |
|  | $('#loadingModal').modal('hide'); |
|  | $('#submitShipment').prop('disabled', false); |
|  |  |
|  | } |
|  | } |

#### Create Shipment

When a user selects a rate and clicks “Submit Shipment” in the “getRates” HTML Web resource, the shipment gets created on the carriers servers and a tracking number + cost will be returned (in our case, saved to the Shipment entity + status changes from Draft -> Active)

##### Custom Activity to Submit Shipment

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

using Microsoft.Xrm.Sdk.Workflow;

using System;

using System.Activities;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Newtonsoft.Json;

using Newtonsoft.Json.Linq;

using System.Dynamic;

using System.Net.Http;

namespace ZenkraftWorkflows

{

public class SubmitShipment : CodeActivity

{

#region Input Parameters

[Input("Shipment Id")]

public InArgument<String> ShipmentId { get; set; }

[Input("Service Type")]

public InArgument<String> ServiceType { get; set; }

#endregion

#region Output Parameters

[Output("Success")]

public OutArgument<Boolean> Success { get; set; }

[Output("Error Message")]

public OutArgument<string> ErrorMessage { get; set; }

#endregion

#region variables

static string errorMessage = "";

#endregion

protected override void Execute(CodeActivityContext executionContext)

{

try

{

//Create the tracing service

ITracingService tracingService = executionContext.GetExtension<ITracingService>();

//Create the context

IWorkflowContext context = executionContext.GetExtension<IWorkflowContext>();

IOrganizationServiceFactory serviceFactory = executionContext.GetExtension<IOrganizationServiceFactory>();

IOrganizationService service = serviceFactory.CreateOrganizationService(context.UserId);

tracingService.Trace("Starting Workflow: SubmitShipment");

//get input params

string shipmentId = ShipmentId.Get<String>(executionContext);

string serviceType = ServiceType.Get<String>(executionContext);

tracingService.Trace("Retrieved Input Param: {0}, {1}", shipmentId, serviceType);

//Checking if Configuration exists, if it does get

string apiKey = “your\_api\_key\_here”;

tracingService.Trace("Retrieved API Key: {0}", apiKey);

if (string.IsNullOrEmpty(apiKey))

{

Success.Set(executionContext, false);

ErrorMessage.Set(executionContext, "No configuration with API key was found.");

return;

}

//Getting test mode from config

bool test = true;

tracingService.Trace("Test Mode: {0}", test.ToString());

//Get details from shipment

Guid shipmentGuid = Guid.Parse(shipmentId);

Entity shipmentEntity = service.Retrieve("zk\_shipment", shipmentGuid, new ColumnSet("zk\_carrierid", "zk\_labeltypesizeid", "zk\_labeltypeid", "zk\_packagingtypeid", "zk\_shippingaccount", "zk\_returnshipment", "zk\_dimensionunits", "zk\_weightunits", "zk\_currencyid", "zk\_shipperemail", "zk\_shippercompany", "zk\_shipperphone", "zk\_shippername", "zk\_shipperstreet", "zk\_shippercity", "zk\_shipperstatecode", "zk\_shipperpostalcode", "zk\_shippercountrycode", "zk\_recipientemail", "zk\_recipientcompany", "zk\_recipientphone", "zk\_recipientname", "zk\_recipientstreet", "zk\_recipientcity", "zk\_recipientstatecode", "zk\_recipientpostalcode", "zk\_recipientcountrycode"));

tracingService.Trace("Successfully retrieved shipment entity...");

EntityReference carrierRef = (EntityReference)shipmentEntity.Attributes["zk\_carrierid"];

Entity carrierEntity = service.Retrieve("zk\_carrier", carrierRef.Id, new ColumnSet("zk\_code"));

string carrier = carrierEntity.Attributes["zk\_code"].ToString();

EntityReference labelTypeRef = (EntityReference)shipmentEntity.Attributes["zk\_labeltypeid"];

Entity labelTypeEntity = service.Retrieve("zk\_labeltype", labelTypeRef.Id, new ColumnSet("zk\_name"));

string labelType = labelTypeEntity.Attributes["zk\_name"].ToString();

string labelTypeSize = string.Empty;

if (shipmentEntity.Attributes.Contains("zk\_labeltypesizeid"))

{

var ltz = shipmentEntity.Attributes["zk\_labeltypesizeid"];

if (ltz != null)

{

EntityReference labelTypeSizeRef = (EntityReference)ltz;

Entity labelTypeSizeEntity = service.Retrieve("zk\_labeltypesize", labelTypeSizeRef.Id, new ColumnSet("zk\_name"));

labelTypeSize = labelTypeSizeEntity.Attributes["zk\_name"].ToString();

}

}

EntityReference packagingTypeRef = (EntityReference)shipmentEntity.Attributes["zk\_packagingtypeid"];

Entity packagingTypeEntity = service.Retrieve("zk\_packagingtype", packagingTypeRef.Id, new ColumnSet("zk\_slug"));

string packagingType = packagingTypeEntity.Attributes["zk\_slug"].ToString();

EntityReference shippingAccountRef = (EntityReference)shipmentEntity.Attributes["zk\_shippingaccount"];

Entity shippingAccountEntity = service.Retrieve("zk\_shippingaccount", shippingAccountRef.Id, new ColumnSet("zk\_spid"));

Int64 shipping\_account = Int64.Parse(shippingAccountEntity.Attributes["zk\_spid"].ToString());

bool returnShipment = (bool)shipmentEntity.Attributes["zk\_returnshipment"];

string type = "";

if (returnShipment)

type = "return";

else

type = "outbound";

string dim\_units = Common.getOptionSetLabel(service, ((OptionSetValue)shipmentEntity.Attributes["zk\_dimensionunits"]).Value, "zk\_dimensionunits", "zk\_shipment");

string weight\_units = Common.getOptionSetLabel(service, ((OptionSetValue)shipmentEntity.Attributes["zk\_weightunits"]).Value, "zk\_weightunits", "zk\_shipment");

EntityReference currencyRef = (EntityReference)shipmentEntity.Attributes["zk\_currencyid"];

Entity currencyEntity = service.Retrieve(currencyRef.LogicalName, currencyRef.Id, new ColumnSet("isocurrencycode"));

string currency = currencyEntity.Attributes["isocurrencycode"].ToString();

//Sender details

string sender\_email = shipmentEntity.Attributes["zk\_shipperemail"].ToString();

string sender\_company = shipmentEntity.Attributes["zk\_shippercompany"].ToString();

string sender\_phone = shipmentEntity.Attributes["zk\_shipperphone"].ToString();

string sender\_name = shipmentEntity.Attributes["zk\_shippername"].ToString();

string sender\_street1 = shipmentEntity.Attributes["zk\_shipperstreet"].ToString();

string sender\_city = shipmentEntity.Attributes["zk\_shippercity"].ToString();

string sender\_state = shipmentEntity.Attributes["zk\_shipperstatecode"].ToString();

string sender\_postalcode = shipmentEntity.Attributes["zk\_shipperpostalcode"].ToString();

string sender\_country = shipmentEntity.Attributes["zk\_shippercountrycode"].ToString();

//recipient details

string recipient\_email = shipmentEntity.Attributes["zk\_recipientemail"].ToString();

string recipient\_company = shipmentEntity.Attributes["zk\_recipientcompany"].ToString();

string recipient\_phone = shipmentEntity.Attributes["zk\_recipientphone"].ToString();

string recipient\_name = shipmentEntity.Attributes["zk\_recipientname"].ToString();

string recipient\_street1 = shipmentEntity.Attributes["zk\_recipientstreet"].ToString();

string recipient\_city = shipmentEntity.Attributes["zk\_recipientcity"].ToString();

string recipient\_state = shipmentEntity.Attributes["zk\_recipientstatecode"].ToString();

string recipient\_postalcode = shipmentEntity.Attributes["zk\_recipientpostalcode"].ToString();

string recipient\_country = shipmentEntity.Attributes["zk\_recipientcountrycode"].ToString();

EntityCollection packagesCollection = getPackagesForShipment(shipmentGuid, service, tracingService);

List<Package> packagesArray = new List<Package>();

if (packagesCollection != null)

{

if (packagesCollection.Entities.Count == 0)

{

Success.Set(executionContext, false);

ErrorMessage.Set(executionContext, "Couldn't find packages for that shipment.");

return;

}

else

{

foreach (Entity packageEntity in packagesCollection.Entities)

{

Package package = new Package();

package.value = packageEntity.Attributes["zk\_value"].ToString();

package.length = packageEntity.Attributes["zk\_length"].ToString();

package.width = packageEntity.Attributes["zk\_width"].ToString();

package.height = packageEntity.Attributes["zk\_height"].ToString();

package.weight = packageEntity.Attributes["zk\_weightvalue"].ToString();

packagesArray.Add(package);

}

}

}

else

{

Success.Set(executionContext, false);

ErrorMessage.Set(executionContext, "Couldn't retrieve packages for shipment.");

return;

}

tracingService.Trace("Successfully retrieved all attributes from shipment and related packages...");

//Shipment details

Shipment shipmentObj = new Shipment();

shipmentObj.service = serviceType;

shipmentObj.shipping\_account = shipping\_account;

shipmentObj.ship\_date = DateTime.Now.ToString("yyyy-MM-dd");

shipmentObj.carrier = carrier;

shipmentObj.packaging = packagingType;

shipmentObj.weight\_units = weight\_units;

shipmentObj.currency = currency;

shipmentObj.type = type;

shipmentObj.label\_type = labelType;

if (!string.IsNullOrEmpty(labelTypeSize))

shipmentObj.label\_size = labelTypeSize;

shipmentObj.include\_base64\_label = true;

//sender

Address senderObj = new Address();

senderObj.email = sender\_email;

senderObj.street1 = sender\_street1;

senderObj.country = sender\_country;

senderObj.company = sender\_company;

senderObj.city = sender\_city;

senderObj.postal\_code = sender\_postalcode;

senderObj.phone = sender\_phone;

senderObj.state = sender\_state;

senderObj.name = sender\_name;

shipmentObj.sender = senderObj;

shipmentObj.dim\_units = dim\_units;

shipmentObj.debug = true;

shipmentObj.test = test;

//recipient

Address recipientObj = new Address();

recipientObj.email = recipient\_email;

recipientObj.street1 = recipient\_street1;

recipientObj.country = recipient\_country;

recipientObj.company = recipient\_company;

recipientObj.city = recipient\_city;

recipientObj.postal\_code = recipient\_postalcode;

recipientObj.phone = recipient\_phone;

recipientObj.state = recipient\_state;

recipientObj.name = recipient\_name;

shipmentObj.recipient = recipientObj;

shipmentObj.packages = packagesArray;

shipmentObj.references = new List<ZkField>();

ZkService zkService = new ZkService(apiKey);

zkService.Test = test;

Shipment shipmentFromAPI = zkService.Ship(shipmentObj);

//Update shipment and packages with returned values (cost, tracking numbers, label types)

updateShipment(shipmentGuid, shipmentFromAPI, packagesCollection, service, tracingService);

if (errorMessage == "")

{

Success.Set(executionContext, true);

}

else

{

Success.Set(executionContext, false);

ErrorMessage.Set(executionContext, errorMessage);

}

}

catch (Exception ex)

{

Success.Set(executionContext, false);

string errorMsg = "Failed to submit shipment.";

if (ex != null)

errorMsg = ex.ToString();

ErrorMessage.Set(executionContext, errorMsg);

throw new InvalidWorkflowException(errorMsg);

}

}

private void updateShipment(Guid shipmentGuid, Shipment shipmentObj, EntityCollection packagesCollection, IOrganizationService \_service, ITracingService \_tracingService)

{

\_tracingService.Trace("Entered Method 'updateShipment'");

\_tracingService.Trace("Shipment ID: {0}", shipmentGuid.ToString());

//\_tracingService.Trace("Shipment Returned from Endpoint: {0}", shipmentFromAPI);

if (shipmentFromAPI["shipment"] != null)

{

Entity shipmentEntity = new Entity("zk\_shipment", shipmentGuid);

\_tracingService.Trace("Setting shipment attributes...");

shipmentEntity["zk\_trackingnumber"] = shipmentObj["tracking\_number"].ToString();

shipmentEntity["zk\_shipmentapiid"] = shipmentObj["id"].ToString();

shipmentEntity["zk\_shipdate"] = DateTime.ParseExact(shipmentObj["ship\_date"].ToString(), "yyyy-MM-dd", null);

shipmentEntity["zk\_service"] = shipmentObj["service"].ToString();

\_tracingService.Trace("Setting Shipment Costs...");

JObject costsObj = JObject.Parse(shipmentObj["costs"].ToString());

shipmentEntity["zk\_basecharge"] = costsObj["base\_charge"] == null? null: new Money((decimal)costsObj["base\_charge"]);

shipmentEntity["zk\_discounts"] = costsObj["discounts"] == null ? null : new Money((decimal)costsObj["discounts"]);

shipmentEntity["zk\_netcharge"] = costsObj["net\_charge"] == null ? null : new Money((decimal)costsObj["net\_charge"]);

shipmentEntity["zk\_surcharges"] = costsObj["surcharges"] == null ? null : new Money((decimal)costsObj["surcharges"]);

shipmentEntity["zk\_taxes"] = costsObj["taxes"] == null ? null : new Money((decimal)costsObj["taxes"]);

shipmentEntity["statuscode"] = new OptionSetValue(1); //active

\_tracingService.Trace("Updating shipment...");

\_service.Update(shipmentEntity);

\_tracingService.Trace("Succesffully updated shipment entity.");

\_tracingService.Trace("Setting packages values...");

JArray packagesArray = JArray.Parse(shipmentObj["packages"].ToString());

if(packagesArray.Count != packagesCollection.Entities.Count)

{

throw new InvalidWorkflowException("The number of packages returned by the API is different than the one submitted.");

}

else

{

for(int i = 0; i< packagesArray.Count; i++)

{

JObject packageFromAPI = JObject.Parse(packagesArray.ElementAt(i).ToString());

Entity packageToUpdate = new Entity("zk\_package", packagesCollection.Entities.ElementAt(i).Id);

string labelType = shipmentObj["label\_type"].ToString();

packageToUpdate["zk\_trackingnumber"] = packageFromAPI["tracking\_number"].ToString();

packageToUpdate["zk\_imagebase64"] = labelType + ";" + packageFromAPI["label"].ToString();

packageToUpdate["statuscode"] = new OptionSetValue(1);

\_tracingService.Trace("Updating Package with Id: {0}", packagesCollection.Entities.ElementAt(i).Id);

\_service.Update(packageToUpdate);

\_tracingService.Trace("Successfully updated package.");

}

}

}

else

{

if (shipmentFromAPI["error"] != null)

{

JObject error = JObject.Parse(shipmentFromAPI["error"].ToString());

errorMessage = error["message"].ToString();

}

else

{

throw new InvalidWorkflowException("Shipment object returned is not in the right format.");

}

}

}

private string submitShipmentToAPI(string api\_key, Shipment myShipment, ITracingService \_tracingService)

{

\_tracingService.Trace("Entered Method 'submitShipmentToAPI'");

ZkService zkService = new ZkService(api\_key);

Shipment response = zkService.Ship(myShipment);

return response;

}

private EntityCollection getPackagesForShipment(Guid shipmentId, IOrganizationService \_service, ITracingService \_tracingService)

{

\_tracingService.Trace("Entered Method 'getPackagesForShipment'");

QueryExpression qe = new QueryExpression("zk\_package");

qe.ColumnSet = new ColumnSet("zk\_weightvalue", "zk\_value", "zk\_length", "zk\_width", "zk\_height");

ConditionExpression condition1 = new ConditionExpression("statecode", ConditionOperator.Equal, 0); //Active

ConditionExpression condition2 = new ConditionExpression("zk\_shipmentid", ConditionOperator.Equal, shipmentId);

qe.Criteria.AddCondition(condition1);

qe.Criteria.AddCondition(condition2);

EntityCollection ec = \_service.RetrieveMultiple(qe);

return ec;

}

}

}