Microsoft Dynamics 365 for Finance and Operations

Shipment management

overview document

Contents

[Introduction 2](#_Toc4405441)

[Integration DLL 2](#_Toc4405442)

[Models 2](#_Toc4405443)

[General mapping 2](#_Toc4405444)

[List of object models 3](#_Toc4405445)

[Messages 3](#_Toc4405446)

[Mapping 3](#_Toc4405447)

[API client 4](#_Toc4405448)

[Methods signature 4](#_Toc4405449)

[Shipping Management model 4](#_Toc4405450)

[Importing model 4](#_Toc4405451)

[New tables 4](#_Toc4405452)

[Extended tables 4](#_Toc4405453)

[Assets attached 5](#_Toc4405454)

# Introduction

The object of this project is to create an integration between Zenkraft APIs and Microsoft Dynamics 365 for Operations. The main modules integrated with Zenkraft integration are: Warehouse management and inventory management. For that, the transportation modules is also customized in order to adapte the API data with Microsoft Dynamics 365 for Operations.

# Integration DLL

## Models

### General mapping

Each ZK object is mapped to c# object in Zenkraft.FO.Services. The DataContract and DataMember attributes are mandatory for object conversion. Below example shown identical fields between both objects.

**Input c# example**

[DataContract(Name = "shipment")]

public class Shipment : Testable

{

[DataMember(Name = "id", EmitDefaultValue = false)]

public string Id { get; set; }

[DataMember(Name = "carrier", EmitDefaultValue = false)]

public string CarrierCode { get; set; }

[DataMember(Name = "service", EmitDefaultValue = false)]

public string Service { get; set; }

[DataMember(Name = "sender", EmitDefaultValue = false)]

public Address Sender { get; set; }

[DataMember(Name = "references", EmitDefaultValue = false)]

public Reference[] References { get; set; }

}

**Output JSON example**

{

"shipment": {

"id": "", "carrier": "", "service": "", "packaging": "",

"sender": {

"city": "", "company": "", "country": "",

},

"references": [

{"type": "", "value": ""}

],

}

}

### List of object models

* Account
* Address
* Authorization
* Cancel
* Carrier
* CarrierAuth
* CarrierReference
* CheckPoint
* Cost
* CustomsInfo
* CustomsItem
* Error
* Header
* LabelType
* Location
* Package
* PackagingType
* Rate
* Reference
* ServiceType
* Shipment
* ShippingAccount
* ShippingDocument
* Success
* Track
* WebError

## Messages

### Mapping

List of message objects

* AccountRequest
* AccountResponse
* CancelRequest
* CancelResponse
* CarrierResponse
* CarriersResponse
* RateRequest
* RateResponse
* ShippingAccountRequest
* ShippingAccountResponse
* ShipRequest
* ShipResponse
* TrackRequest
* TrackResponse

## API client

### Methods signature

* public static AccountResponse CreateAccount(Header header, AccountRequest request)
* public static ShippingAccountResponse PostShippingAccount(Header header, ShippingAccountRequest request)
* public static CarriersResponse GetCarriers(Header header)
* public static CarrierResponse GetCarrier(Header header, string carrierSlug)
* public static RateResponse PostRate(Header header, RateRequest request)
* public static ShipResponse PostShip(Header header, ShipRequest request)
* public static TrackResponse PostTrack(Header header, TrackRequest request)
* public static CancelResponse PostCancel(Header header, CancelRequest request)

# Shipping Management model

## Importing model

After running the VM and after visual studio configuration, the model should be imported to the VM by running model tools for *ZKShippingManagement-zenkraft.axmodel* file. Then open the *Zenkraft.FO* project in visual studio.

Use this command for model imports: **ModelUtil.exe -import -metadatastorepath="K:\AosService\PackagesLocalDirectory" -file="c:\Temp\ZKShippingManagement-zenkraft.axmodel"**

After importing model, user should build the new mode and synchronize the database.

## New tables

* ZKAPILogs
* ZKCarrierAuth
* ZKCarrierLabelTypes
* ZKCarrierPackagingTypes
* ZKCarrierReferences
* ZKLabelTypes
* ZKLabelTypeSizes
* ZKPackages
* ZKPackagesCopy
* ZKPackagesTmp
* ZKParameters
* ZKRates
* ZKShipments
* ZKShipmentsCopy
* ZKShippingAccount
* ZKShippingAccountKeys
* ZKShippingAccountKeysTmp
* ZKShippingDocuments
* ZKTableFieldLookupTmp
* ZKTracksCheckPoints

## Extended tables

* TMSCarrier
* TMSCarrierService
* WHSShipmentTable
* WMSShipment

# Assets attached

* **AXDeployablePackage\_20190325\_09\_06\_12.zip**: The latest deployable package.
* **ZKShippingManagement-zenkraft.axmodel:** contains main customization model for Zenkraft, and should imported in development VM.
* **ZKShippingManagment.zip:** containsall customized object, this is used if anyone need to check the object without importing the model.
* **ZKShippingManagement.axpp** and **ZKShippingManagement\_WHS.axpp** the two unified project used in the above solution.