

# **LS Omni 3.x Ecommerce Interface**

## **Developers Guide**

March 2019

# Contents

<b>1</b>	<b>Introduction.....</b>	<b>3</b>
<b>2</b>	<b>Create new Project.....</b>	<b>4</b>
<b>3</b>	<b>Replication .....</b>	<b>7</b>
<b>4</b>	<b>Member Contact.....</b>	<b>10</b>
4.1	Create Member Contact.....	11
4.2	Profiles .....	13
4.3	Login.....	13
<b>5</b>	<b>Product Setup .....</b>	<b>14</b>
5.1	Hierarchy.....	14
5.2	Price and discount.....	15
5.3	Stock Status.....	18
<b>6</b>	<b>Order, Basket &amp; Wishlist .....</b>	<b>19</b>
6.1	Basket.....	20
6.2	Wish List.....	21
6.3	Calculate .....	22
6.4	Order Create .....	23
6.5	Order Lines.....	25
6.6	Payments.....	26
6.7	Order History.....	27
6.8	Orders in LS Nav/Central .....	28
6.8.1	Sales Order	28
6.8.2	Click And Collect Order	29
<b>7</b>	<b>Other Code Samples .....</b>	<b>31</b>
7.1	Get Image for Item.....	31
7.2	Get Image Stream for Item.....	31

# 1 Introduction

The purpose of this document is to give an overview of how to start implementing Omni against eCommerce solution. The document describes how to create new project and how to connect it to the LS Omni Interface and shows sample of some common operations. The document is written using Visual Studio 2017. It uses Web Form sample and it only focus on the code behind to communicate with Omni, no entry form or result forms are generated, just simple buttons with code behind.

**NOTE: LS Central is still in development against the eCommerce so changes will happen, and this document will be updated when new features or different methods of doing things will be available.**

Install LS Omni Server and configure it to connect to LS Central (13.xx). For installation of the LS Omni server, see “LS Omni Server install and configuration guide”

After Installing the LS Omni server some data needs to be prepared in LS Central for LS Omni, see “LS Central Configuration for LS Omni 3.x” for more information.

To test connection to LS Omni and check if everything is working and LS Omni can talk to LS Central, open browser and type in this URL (Pointing to the host where LS Omni is running on)

<http://localhost/lsonnIService/appjson.svc/ping>

if all is ok, you should get reply like this:

```
"PONG OK. Successfully connected to [LSOmni Db] & [NAV Db] & [NAV web service] NAV:
13.04.00.819 [27233] OMNI: 3.6.0.0"
```

Open up **AppSettings.config** file where LS Omni is installed (C:\LS Retail\LSOmni\LSOmniService) and change the value **Security.Validatetoken** to **false**.

```
<add key="Security.Validatetoken" value="false"/>
```

The purpose of this value is to validate requests coming from Mobile Loyalty Apps.

To secure eCommerce communication, use the Basic Authentication settings in IIS Server and send the Login Authentication with every request.

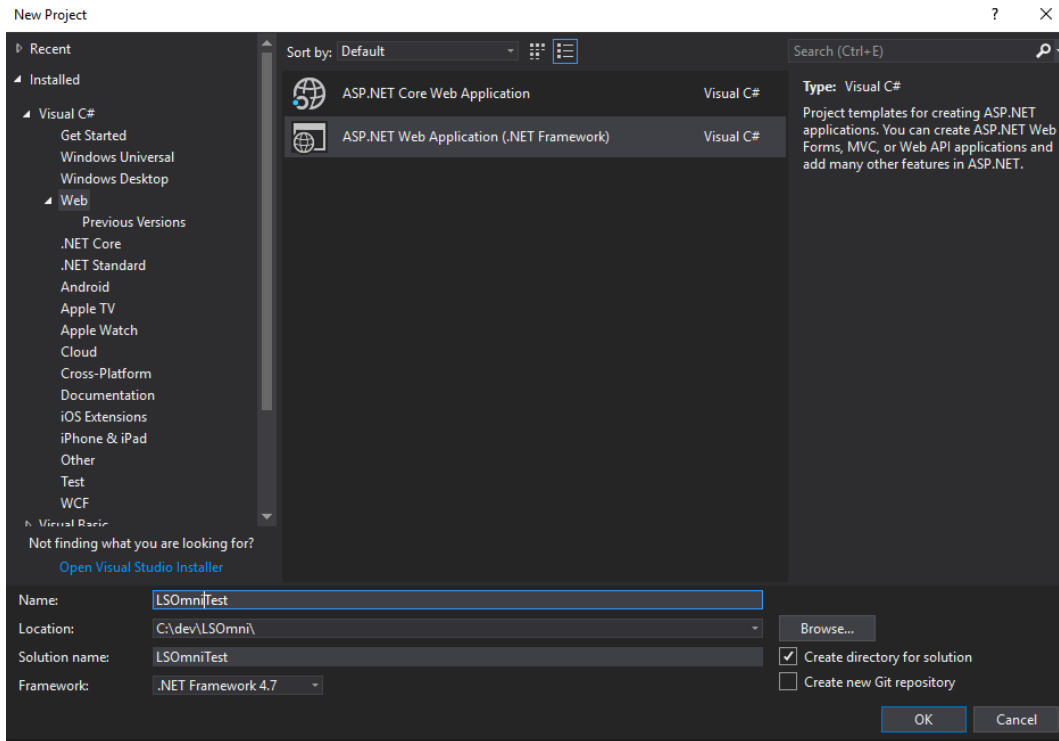
Online help for the LS Omni eCommerce Interface can be found here:

[http://mobiledemo.lsretail.com/lsonnihelp/html/T\\_LSOmni\\_Service\\_IeCommerceService.htm](http://mobiledemo.lsretail.com/lsonnihelp/html/T_LSOmni_Service_IeCommerceService.htm)

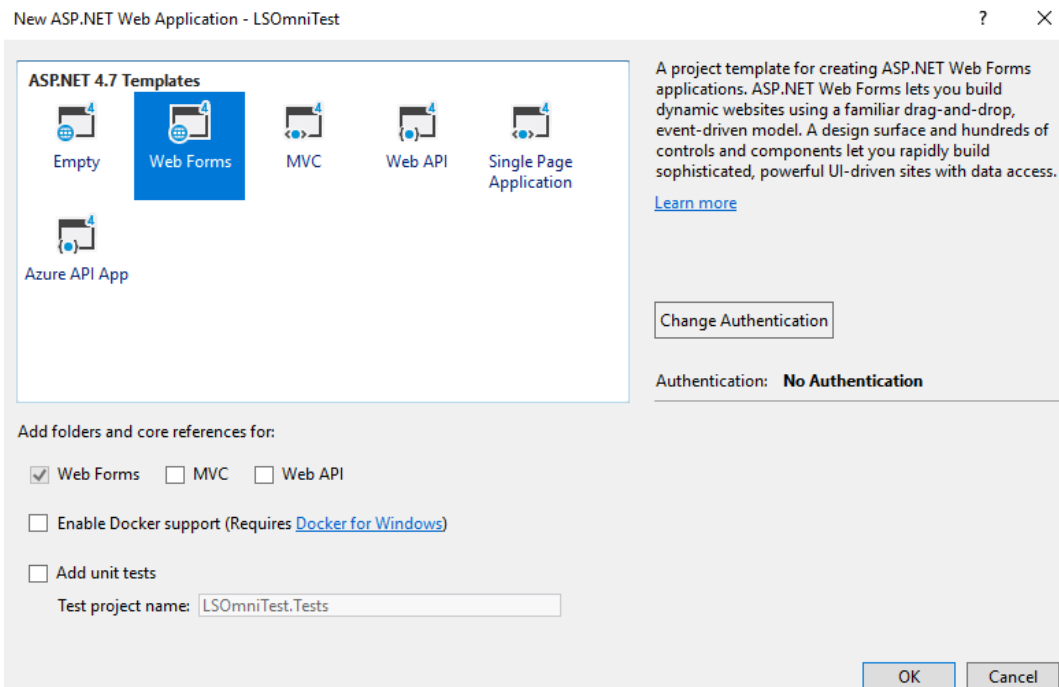
There you can see some samples on SOAP requests, LS Nav/Central WS request data and how LS Omni object maps to the LS Nav/Central WS XML.

## 2 Create new Project

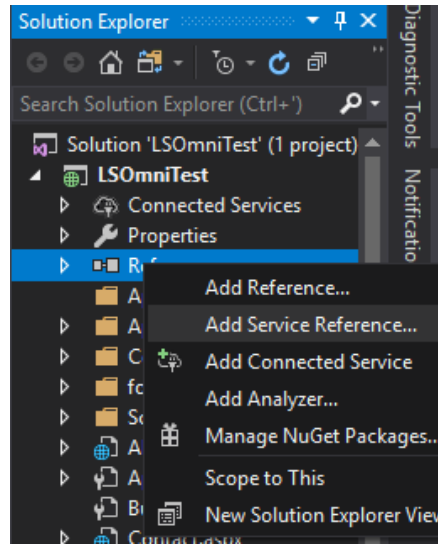
In Visual Studio, create new ASP.NET Web-Application project



### Select Web Forms



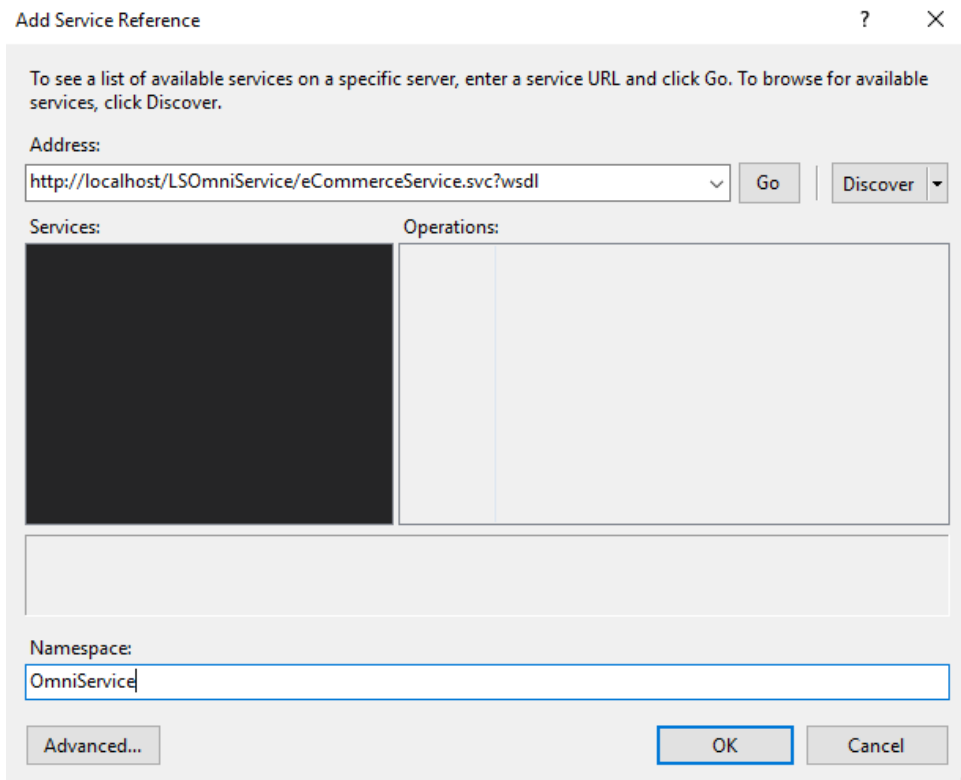
Add Service Reference by right click on Reference section



Put in the URL Address for LS Omni service pointing to the host where LS Omni is running on

<http://localhost/LSOmniService/eCommerceService.svc?wsdl>

Press Go, then OK.



Open **web.config** file and change the **<bindings>** section and add max size values to it.

```
<system.serviceModel>
  <bindings>
    <basicHttpBinding>
      <binding name="BasicHttpBinding_IeCommerceService"
        bufferSize="2147483647" maxReceivedMessageSize="2147483647" />
    </basicHttpBinding>
  </bindings>
  <client>
    <endpoint address=http://bunga/LSOmniService/eCommerceService.svc
      binding="basicHttpBinding"
      bindingConfiguration="BasicHttpBinding_IeCommerceService"
      contract="OmniService.IeCommerceService"
      name="BasicHttpBinding_IeCommerceService" />
  </client>
</system.serviceModel>
```

### 3 Replication

LS Omni has interface to replicate data from LS Central via Omni over to eCommerce.

All functions take in a **ReplRequest** Object that tells what data to get and return a Response object with list of records and other information about the status of the replication.

A Full Replication takes all the data from the source table. Full Replication should be done when new store has been set up or data for a store needs to be recovered or restored again. When doing Update Replication, then LS Central PreActions are used to get all affected records since last Replication and **lastKey** will point to last PreAction counter which should be sent in every time an Update Replication is performed. If **lastKey** is set to 0, then all affected records with PreAction in LS Central will be replicated.

For deleted data, Update Replication will include that record with key data and property **IsDeleted** set to **true**.

Some of the functions support Store distribution. Set the **StoreId** in replication request to replicate only data relevant to that store.

More detailed information on what data it replicates can be found online for each function. The functions are:

#### Product Setup

ReplEcommItems  
 ReplEcommUnitOfMeasures  
 ReplEcommItemUnitOfMeasures  
 ReplEcommItemVariantRegistrations  
 ReplEcommExtendedVariants  
 ReplEcommAttribute  
 ReplEcommAttributeOptionValue  
 ReplEcommAttributeValue  
 ReplEcommBarcodes  
 ReplEcommBasePrices  
 ReplEcommPrices  
 ReplEcommDiscounts  
 ReplEcommMixAndMatch  
 ReplEcommFullItem  
 ReplEcommHierarchy

ReplEcommImageLinks  
 ReplEcommImages  
 ReplEcommItemCategories  
 ReplEcommProductGroups

#### Basic Data

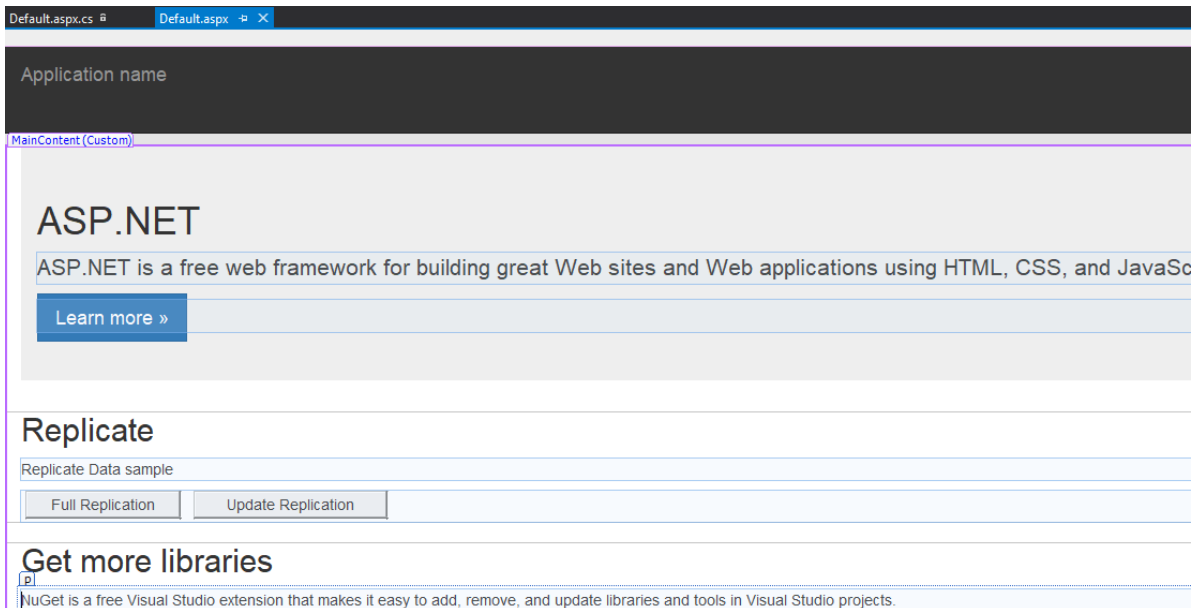
ReplEcommCountryCode  
 ReplEcommCurrency  
 ReplEcommCurrencyRate  
 ReplEcommDataTranslation  
 ReplEcommMember  
 ReplEcommShippingAgent  
 ReplEcommStores  
 ReplEcommStoreTenderTypes  
 ReplEcommVendor  
 ReplEcommVendorItemMapping

FullItem function returns Item with list of Variants, Unit of Measures, Attributes and Prices.

Open **Default.aspx** page and go to Design mode.

It has lot of default information, we will just replace some of it to get things started.

Replace the first section with new text like as in the image here above and remove the Learn More item and add 2 Buttons instead, one for Full Replication and one for Update Replication.



Full Replication replicates all the data from LS Central and will store the last PreAction counter when done. Update Replication will look for any changes from **LastKey** since last Full or Update Replication was made. **LastKey** for each Replication function should be stored separately.

In this sample, you will get data back when doing Full Replication but after Full Replication, Update Replication will be empty. Either do some changes to the data in LS Central and then run Update Replication again or run Update Replication first after starting up the web sample. Adding this line of code before the while loop in Button2 function will reset the **LastKey** value so Update Replication will always return data.

```
Session["LastItemKey"] = "0";
```

Replicating Images will return Image in base64 string format. This data can be used directly on a web page by adding the string to WebControl Image control

```
Image1.ImageUrl = "data:image/jpeg;base64," + replImage.Image64;
```



## Double click on both Replication Buttons and then add following code to **Default.aspx.cs**

```
public partial class _Default : Page
{
    private OmniService.ReplRequest request; // Replication Request object
    private OmniService.IeCommerceServiceClient serviceClient;

    protected void Page_Load(object sender, EventArgs e)
    {
        serviceClient = new OmniService.IeCommerceServiceClient();
        request = new OmniService.ReplRequest();
        request.BatchSize = 100;           // Number of records to get in each Replication call
        request.LastKey = "0";             // Current record point during replication
        request.MaxKey = "0";              // Highest record point expected after replication
        request.StoreId = "S0001";        // Store to replicate data for
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        // Full replication is needed first time Store is activated
        request.FullReplication = true;
        while (true)
        {
            request.LastKey = Session["LastItemKey"] as string; // Set LastKey from previous call
            OmniService.ReplItemResponse result = serviceClient.ReplEcommItems(request);
            Session["LastItemKey"] = result.LastKey; // Store LastKey
            if (result.RecordsRemaining <= 0)
                break; // Replication is done, no more data

            foreach (OmniService.ReplItem item in result.Items)
            {
                // iterate through the list of returned items
            }
        }
    }

    protected void Button2_Click(object sender, EventArgs e)
    {
        // After Full replication, only Update replication is needed from LastKey record point
        request.FullReplication = false;
        while (true)
        {
            request.LastKey = Session["LastItemKey"] as string; // Set LastKey from previous call
            OmniService.ReplItemResponse result = serviceClient.ReplEcommItems(request);
            Session["LastItemKey"] = result.LastKey; // Store LastKey
            if (result.RecordsRemaining <= 0)
                break; // Replication is done, no more data

            foreach (OmniService.ReplItem item in result.Items)
            {
                // iterate through the list of returned items
            }
        }
    }
}
```

All **OmniService.ReplXxxxResponse** Objects include a list Object with the data record that is being replicated. **LastKey** has the last point in time or last PreAction Id during and after replication. **MaxKey** is not really used but is show the highest PreAction Id that is expected after replication is done. **RecordsRemaining** show how many records are left to pull and when it reaches 0 then all records have been delivered.

**NOTE:** In some cases, replication will always return all records, as these are tables that don't have any PreAction setup and are usually include small amount of data that does not get updated frequently.

## 4 Member Contact

Member Contact is a customer that can log into the eCommerce site and look at his shopping history, check offers and do some shopping.

Each Member Contact is assigned to a Member Account. Account can have more than one member if it's not set as Private. Member Account can be in Member Club and Club has Scheme levels.

Member Contact gets a Member Card and can have many Member Cards. Many functions in LS Central / Omni work on Member Card as it's a unique identifier for the Contact. When sending in an Order, only Member Card Id is needed and from there Contact and Account can be looked up.

Member Contact Functions are:

ContactCreate	Login
ContactUpdate	LoginWeb
ContactGetByAlternateId	Logout
ContactGetById	
ContactGetPointBalance	NotificationsGetByContactId
ContactSearch	NotificationsUpdateStatus
ChangePassword	AccountGetById
ForgotPassword	ProfilesGetAll
ResetPassword	ProfilesGetByContactId
	PublishedOffersGetByCardId
	SchemesGetAll

Open Design mode on **Default.aspx** page and replace next section with Member Contact and add Create and Login button.

---

## Replicate

Replicate Data sample

---

## Member Contact

Member contact functions

---

[liv.col-md-4](#)

## 4.1 Create Member Contact

Double click on the Create button and add following code

```
protected void Button3_Click(object sender, EventArgs e)
{
    OmniService.MemberContact contact = new OmniService.MemberContact();

    contact.FirstName = "John1";
    contact.LastName = "Doh";
    contact.Gender = OmniService.Gender.Male;
    contact.Email = "john1@company.com";
    contact.UserName = "john1@company.com";
    contact.Password = "myPwd";
    contact.Addresses = new OmniService.Address[1];
    contact.Addresses[0] = new OmniService.Address()
    {
        Address1 = "MyStreet 1",
        City = "MyCity",
        PostCode = "1234",
        Type = OmniService.AddressType.Residential
    };

    OmniService.MemberContact result = serviceClient.ContactCreate(contact);
}
```

**contactResponse** object will include all information assigned to the new Member Contact in LS Central. It will include Member Contact Id, new Member Account, new Member Card. The Account will be assigned to default Member Scheme Clubs.

To Assign the new Member Account to certain Member Club, add

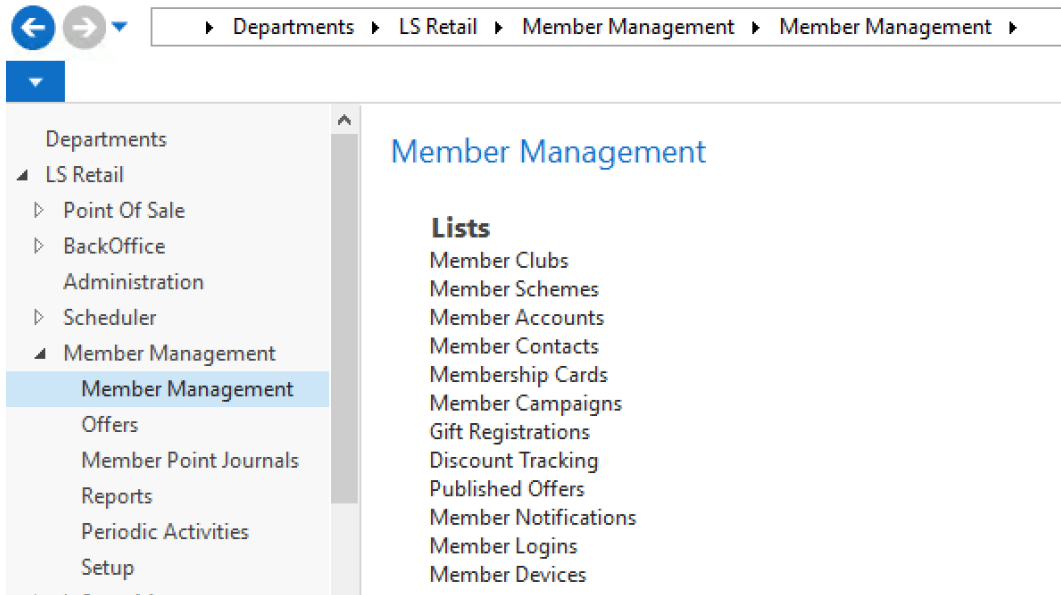
```
contact.Account = new OmniService.Account();
contact.Account.Scheme = new OmniService.Scheme();
contact.Account.Scheme.Club = new OmniService.Club();
contact.Account.Scheme.Club.Id = "CRONUSHOSP";
```

To add Contact to existing Member Account, add

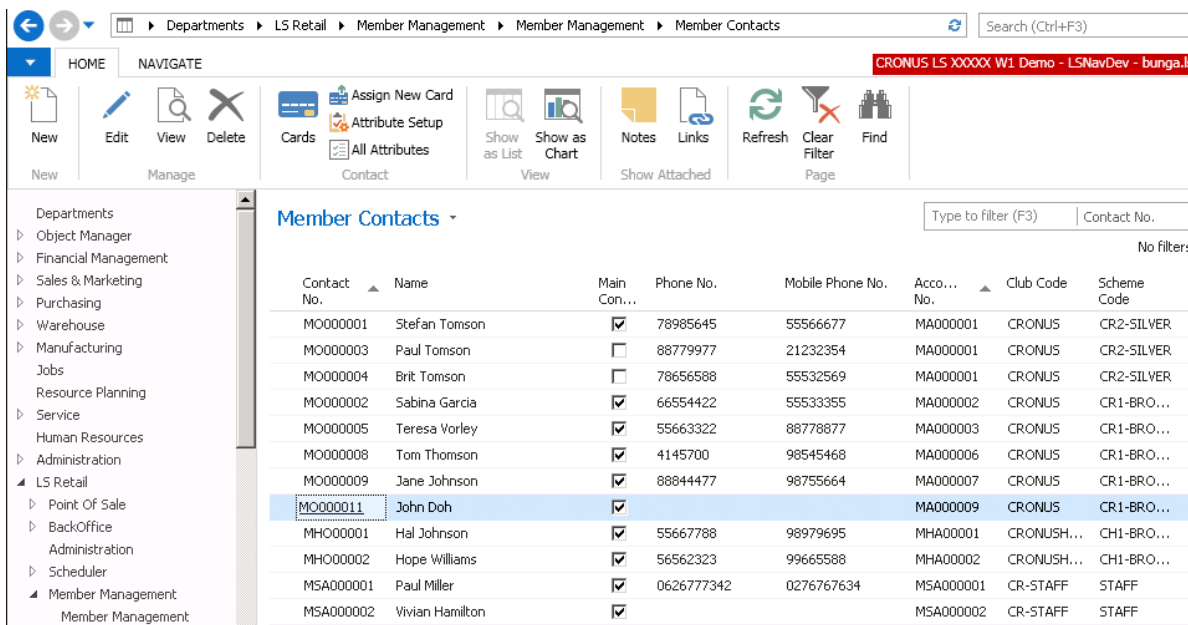
```
contact.Account = new OmniService.Account();
contact.Account.Id = "MA000003";
```

Member Management in LS Central can be found under

->Departments/LS Retail/Member Management/Member Management



Your new contact can be found under Member Contacts



## 4.2 Profiles

Profile is a Member Attribute in LS Central that can be assigned to the Member Contact.

ProfilesGetAll – Gets all Attributes that are assigned to the Default Club Set in Member Management Setup, and are Type Boolean and have Lookup Type set as None.

ProfilesGetByContactId – Gets all attributes that have been assigned to the Contact. The Attribute Value has to be 'Yes' or 'No' as it used to determine if the Attribute is selected or not.

## 4.3 Login

Login function will check if the Member Contact exists in LS Central and gets all the details about the Member Contact and gets all offers, coupon and profiles that are available for the Member Contact.

Double click on the Login button that was added in previous chapter and add following code

```
protected void Button4_Click(object sender, EventArgs e)
{
    OmniService.MemberContact result = serviceClient.Login("john1@company.com", "myPwd",
string.Empty);
}
```

**contactResponse** object will include all information assigned to the Member Contact in LS Central.

Difference between **Login** and **LoginWeb** is Login provides more detailed information about the Contact, and includes Profiles, Publish Offers and Notifications, while **LoginWeb** has just basic Information.

Logout only removes the active device record, other than that it does not do anything else.

## 5 Product Setup

### 5.1 Hierarchy

Products can be organized in two different ways for display on Web Site. The usual method is to use Item Categories and Product Groups. This only gives two levels to group products together.

The other method is to use Hierarchy, which was added into later versions of LS Nav 2017.

In the LS Central Demo data, hierarchy FASHION is set to Store S0013

Open the Store Card and at the bottom there is section to select hierarchy and it has a starting date when it becomes active so there could be more than one hierarchy set to a store.

#### S0013 - Cronus Web Store

web store:		web store shipping cost item:	100010
Loyalty:	<input type="checkbox"/>	Last Action Entry No.:	0
Web Store POS Terminal:	P0040	Mobile:	<input type="checkbox"/>
Web Store Staff ID:	1301	PLU Menu Profile:	
Web Store Customer No.:	44090	PLU Menu ID:	

**Attributes** ^

---

**Retail Calendar** ⚙️ ^

📅 Card
🔗 Group Linking
📅 Retail Calendar List
🔍 Find

Cal...	Group	ID	Description	Used as
Type	Type			Type

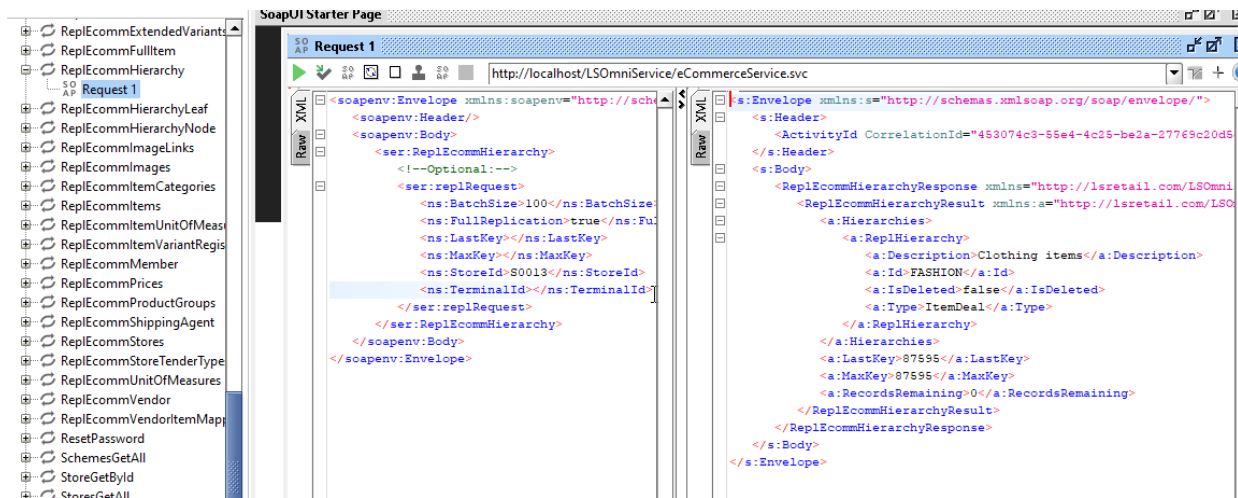
---

**Hierarchy Dates** ^

✕ Delete
🔍 Find
🔍 Filter
🗑️ Clear Filter

Hierarchy Code	Start Date
<b>FASHION</b>	<b>1.3.2018 00:00</b>

To get the hierarchy setup, there are two ways of doing it. One is to replicate the whole setup in three separate calls `ReplEcommHierarchyXXX` and pass in the store Id (S0013). This will give you the Hierarchy setup as Root, Node and Leaves.



The other way is to get the hierarchy set up in one Hierarchy object, by calling **HierarchyGet** with store ID.

## 5.2 Price and discount

Before prices and discount can be replicated, OMNI\_xxxx Scheduler Job in LS Central has to be executed.

**OMNI\_INIT** generates Best Price and gets Discounts and other information for current day for all Items.

**OMNI\_UPDATE** updates this data for items that have been updated since last OMNI\_UPDATE was scheduled.

OMNI\_UPDATE should be scheduled to run frequently but OMNI\_INIT is just needed for first time and if new store has been added or if whole data needs to be processed again for all stores.

**Prices** are replicated with either of these two functions:

### ReplEcommPrices

These are the “best prices” for today, based on the price engine that LS Central has in place. The data for this function comes from **WI Prices** table in LS Central.

**NOTE: Only Price with VAT is included in this data.**

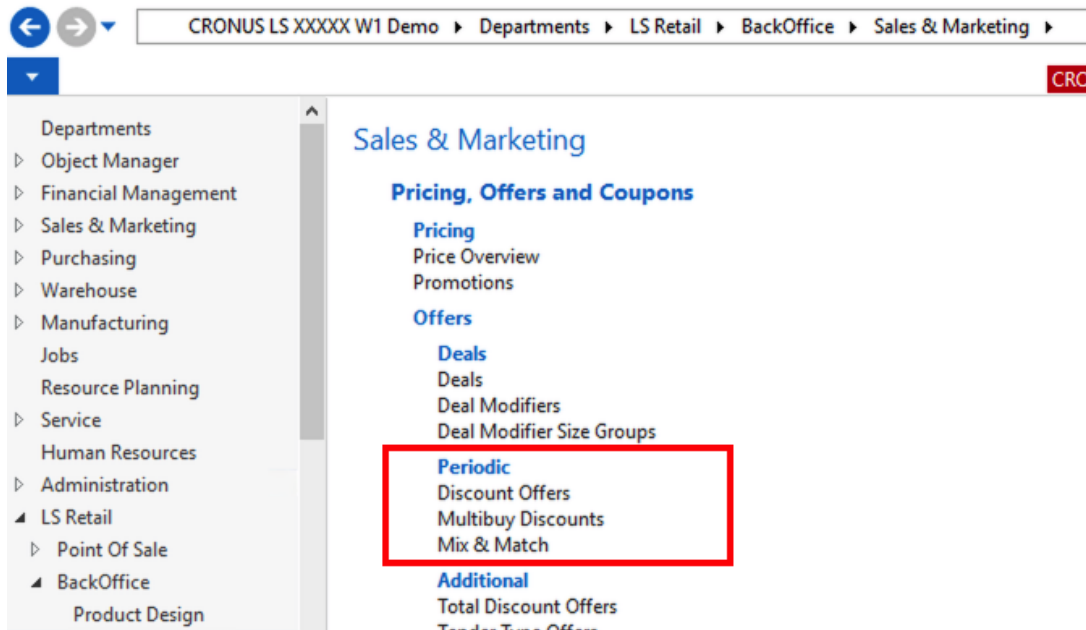
### ReplEcommBasePrices

Gets data direct from **Sales Price** Table in LS Central. These prices can be found in the Item page under the Pricing Section in LS Central.

Main difference between these two functions is that Best price has to be generated by running the OMNI\_xxxx Job in LS Central and the data has to be replicated daily to get the best price for today. There is no start or end date to those prices.

Sales Price is taken “as is” from LS Central so it can be set up for the item in the Retail Item card under Pricing section with start and end date. These prices do not take into account any price modification that could be in place in LS Central.

Only Periodic Offers are available for replication in LS Omni. It can be found here in LS Central.



**Discounts** are replicated with these two functions:

### ReplEcommDiscounts

Gets **Periodic Discount Offers** and **Multibuy Discounts**, including the discount Percent and minimum Quantity for Multibuy Discounts. Use this to calculate discount from the Price from Price replication functions.

### ReplEcommMixAndMatch

Gets **Mix & Match Offers** information, no setup for the Mix & Match is delivered, only what items are involved and descriptions of the Offer. Use this to display that a item is part of some offer, but to be able to get the discounted value the basket has to be sent in for calculation and LS Central will then return the price for the basket based on the Mix & Match Offer.

Both these functions return Offers with start and end date so its possible to see the validation dates for the Offers. A Priority value is also included which tells which offer would be taken prior if an item is part of more than one Offer. In both cases the Basket Calculation will always return the real price and discount based on the LS Central Price and Discount engine, and should always be displayed prior to user confirming the final order.



**DiscountsGet** can be used to send in list of items to get all available Offers for those Items, based on Loyalty Scheme code if its sent in with the request. This function can be used on Item list page, by send in all the items codes for that page or when displaying the Shopping Basket.

The data returned is the Offer Code and description of the offer which can be used to display information about what Offers are available for the items. No discount values or percents are delivered.

**ItemsGetByPublishedOfferId** is a function that returns list of all items that belong to a Published Offer.

### 5.3 Stock Status

**ItemsInStockGet** will return stock status for an item. If store Id is passed in, only status for that store will be returned otherwise status from all available stores that have been marked with Loyalty or Mobile checked in LS Central Store Setup (Omni Section) will be returned.

#### S0013 · Cronus Web Store

Numbering		▼	
Kitchen Printing		▼	
Omni			
Web Store:	<input checked="" type="checkbox"/>	Web Store Shipping Cost I...	66010 ▼
Loyalty:	<input checked="" type="checkbox"/>	Last Action Entry No.:	0
Web Store POS Terminal:	P0040 ▼	Mobile:	<input type="checkbox"/>
Web Store Staff ID:	1301 ▼	PLU Menu Profile:	▼
Web Store Customer No.:	44090 ▼	PLU Menu ID:	▼

**ItemsInStoreGet** will return stock status for list of items and item variants. If store Id is passed in, only status for that store will be returned otherwise status for all stores that have the item available will be returned. This function does not check the Loyalty and Mobile settings like **ItemsInStockGet** does.

**StoresGetbyItemInStock** will get list of Click and Collect Stores within some distance from a location point, that item is available in.

#### S0013 · Cronus Web Store

General		POS	
No.:	S0013	Interface Profile:	#FASHION ▼
Name:	Cronus Web Store	Menu Profile:	#FASHION ▼
Store Type:	Store ▼	Functionality Profile:	#MOBILE ▼
<b>Address</b>		Style Profile:	#FASHION ▼
Address:	1560 Blix street	Hardware Profile:	#DEMO ▼
Address 2:		Inventory Lookup:	<input checked="" type="checkbox"/>
Post Code/City:	CA-ON N6B 1V7 ▼	Check Z-Report:	<input type="checkbox"/>
City:	London ▼	Responsibility Center:	▼
Country Code:	GB ▼	Location Code:	S0013 ▼
Phone No.:	11 22 33 44	In-Transit Code:	▼
Location Profile:	ALL ▼	Distribution Group:	ECOMMERCE ▼
Language Code:	▼	Default Commission Group:	▼
Currency Code:	▼	Click and Collect:	<input type="checkbox"/>
Opening Hours Today:			

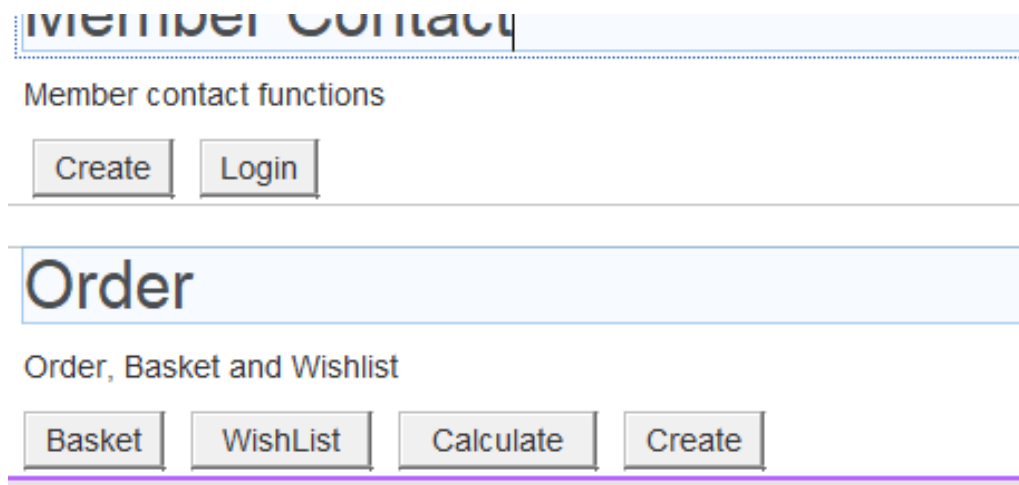
There is no way as is for now to replicate the stock status as the process of finding out the stock status is not a simple task in LS Central. Stock status can be pulled at time of Detailed Item lookup to minimize to data calls to LS Central.

## 6 Order, Basket & Wishlist

LS Omni stores user Basket and Wishlist under OneList object which is only stored in local LS Omni Database and is not sent to LS Central. The data can be found in tables named **OneListXXXX**. When user is ready to complete order, a Calculation request needs to be sent in to get all latest prices and discounts from LS Central. Order should be updated with the result from the Calculation. When user agrees and accepts the status of an Order, then **OrderCreate** Request can be sent in. Functions to handle Basket, Wishlist and Orders are:

OrderCreate	BasketCalc
OrderGetById	RecommendedActive
OrderGetByWebId	RecommendedItemsGet
OrderHistoryByContactId	ItemsInStockGet
OrderCheckAvailability	
	OneListSave
SalesEntriesGetByContactId	OneListDeleteById
SalesEntryGetById	OneListGetByContactId
TransactionGetByReceiptNo	OneListGetById
	OneListCalculate
GetPointRate	

Open Design mode on **Default.aspx** page and replace next section with Order and add 4 buttons, Basket, WishList, Calculate and Create.



## 6.1 Basket

Double click on the Basket button and add following code.

```
protected void Button5_Click(object sender, EventArgs e)
{
    OmniService.OneList list = new OmniService.OneList();
    list.CardId = "10023";
    list.ListType = OmniService.ListType.Basket;

    list.Items = new OmniService.OneListItem[3];
    list.Items[0] = new OmniService.OneListItem()
    {
        Item = new OmniService.LoyItem()
        {
            Id = "40060"
        },
        VariantReg = new OmniService.VariantRegistration()
        {
            Id = "002"
        },
        Quantity = 1
    };
    list.Items[1] = new OmniService.OneListItem()
    {
        Item = new OmniService.LoyItem()
        {
            Id = "40230"
        },
        Quantity = 1
    };
    list.Items[2] = new OmniService.OneListItem()
    {
        Item = new OmniService.LoyItem()
        {
            Id = "43120"
        },
        VariantReg = new OmniService.VariantRegistration()
        {
            Id = "002"
        },
        Quantity = 1
    };

    OmniService.OneList result = serviceClient.OneListSave(list, false);
}
```

Set the **CardId** that you get back after Create our Contact in previous Chapter.

When storing Basket, a Calculate Request can be added. If Calculate is set to true, then **StoreId** needs to be added to the **OneList** object so LS Central knows which store it should apply prices and discount from.

Omni will send Calculate Request to LS Central and update the Basket with correct Prices and Discounts. Updated Basket (OneList) will then be returned after the call is completed.

## 6.2 Wish List

Double click on the WishList button and add following code.

```
protected void Button6_Click(object sender, EventArgs e)
{
    OmniService.OneList list = new OmniService.OneList();
    list.CardId = "10023";
    list.ListType = OmniService.ListType.Wish;

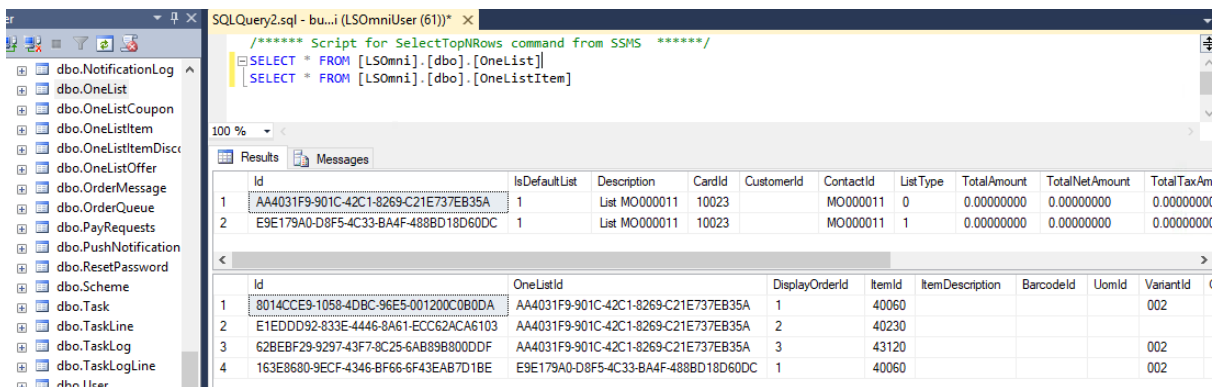
    list.Items = new OmniService.OneListItem[1];
    list.Items[0] = new OmniService.OneListItem()
    {
        Item = new OmniService.LoyItem()
        {
            Id = "40060"
        },
        VariantReg = new OmniService.VariantRegistration()
        {
            Id = "002"
        },
        Quantity = 1
    };

    OmniService.OneList result = serviceClient.OneListSave(list, false);
}
```

Wish List works in similar way as Basket, just put **ListType** to **Wish**.

Calculate Request will not be performed, even if Calculate is set to True. Only Items will be stored with the **OneList** object, no discounts will be saved.

After creating a Basket and/or Wishlist you can find the data for it in LS Omni Database



Id	IsDefaultList	Description	CardId	CustomerId	ContactId	List Type	TotalAmount	TotalNetAmount	TotalTaxAm
AA4031F9-901C-42C1-8269-C21E737EB35A	1	List MO000011	10023		MO000011	0	0.00000000	0.00000000	0.00000000
E9E179A0-D8F5-4C33-BA4F-488BD18D60DC	1	List MO000011	10023		MO000011	1	0.00000000	0.00000000	0.00000000

Id	OneListId	DisplayOrderId	ItemId	ItemDescription	BarcodeId	UomId	VariantId
8014CCE9-1058-4DBC-96E5-001200C0B0DA	AA4031F9-901C-42C1-8269-C21E737EB35A	1	40060				002
E1EDDD92-833E-4446-8A61-ECC62ACA6103	AA4031F9-901C-42C1-8269-C21E737EB35A	2	40230				
628EBF29-9297-43F7-8C25-6AB89B800DDF	AA4031F9-901C-42C1-8269-C21E737EB35A	3	43120				002
163E8680-9ECF-4346-BF66-6F43EAB7D18E	E9E179A0-D8F5-4C33-BA4F-488BD18D60DC	1	40060				002

## 6.3 Calculate

Before an Order can be created, the basket must be Calculated so all prices and discounts will be applied according to LS Central calculation engine.

Double click on the Calculate button and add following code.

```
protected void Button7_Click(object sender, EventArgs e)
{
    OmniService.OneList list = new OmniService.OneList();
    list.CardId = "10023";
    list.ListType = OmniService.ListType.Basket;
    list.Items = new OmniService.OneListItem[1];

    list.Items[0] = new OmniService.OneListItem()
    {
        Item = new OmniService.LoyItem()
        {
            Id = "40020"
        },
        VariantReg = new OmniService.VariantRegistration()
        {
            Id = "002"
        },
        Quantity = 2.0M
    };

    //Optional: This adds a coupon code to the order
    list.PublishedOffers = new List<OneListPublishedOffer>();
    OneListPublishedOffer offer = new OneListPublishedOffer("XMAS2018")
    Offer.Type = OfferDiscountType.Coupon;
    list.PublishedOffers.Add(offer);

    OmniService.Order result = serviceClient.OneListCalculate(list);
}
```

**OneListCalculate** will return Order Lines with prices and calculated amount and discount information if any. This Order object can then be used to finalize the order. Add Contact information, Address for shipping and payment information and then it is ready to be sent in "as is" for Order Create and posting.

## 6.4 Order Create

After Order has been calculated and the extra info like Contact data, Shipping and Payment have been added, it can be sent in for Creation.

Double click on the Create button and add following code to create simple shipping order.

```
protected void Button8_Click(object sender, EventArgs e)
{
    OmniService.Order order = new OmniService.Order ()
    {
        AnonymousOrder = false,
        ClickAndCollectOrder = false,
        StoreId = "S0013",
        CardId = "10023",
        PaymentStatus = OmniService.PaymentStatus.PreApproved,
        ShippingStatus = OmniService.ShippingStatus.NotYetShipped,
        ShipToAddress = new OmniService.Address ()
        {
            Address1 = "Street 11",
            City = "Ghost Town",
            Country = "US" // Country Code
        },
        OrderLines = new OmniService.OrderLine[1],
        OrderPayments = new OmniService.OrderPayment[1]
    };
    order.OrderLines[0] = new OmniService.OrderLine ()
    {
        ItemId = "40020",
        UomId = string.Empty,
        VariantId = "002",
        LineNumber = 1,
        LineType = OmniService.LineType.Item,
        Quantity = 2.0M,
        QuantityToInvoice = 2.0M,
        Price = 80.0M,
        NetPrice = 64.0M,
        Amount = 160.0M,
        NetAmount = 128.0M,
        TaxAmount = 32.0M
    };
    order.OrderPayments[0] = new OmniService.OrderPayment ()
    {
        LineNumber = 1,
        AuthorisationCode = "123456",
        CardType = "VISA",
        CardNumber = "10xx xxxx xxxx 1475",
        TenderType = "1",
        CurrencyCode = "GBP",
        CurrencyFactor = 1,
        FinalizedAmount = 0,
        PreApprovedAmount = 160.0M
    };
    OmniService.Order result = serviceClient.OrderCreate(order);
}
```

For Anonymous Orders set **AnonymousOrder** to **true**. No CardId is set as it's unknown, instead add Contact Name and other details about the Contact.

```
OmniService.Order order = new OmniService.Order()
{
    AnonymousOrder = true,
    ClickAndCollectOrder = false,
    StoreId = "S0013",
    ContactName = "John Doh",
    ContactAddress = new OmniService.Address()
    {
        Address1 = "Street 11",
        City = "Ghost Town",
        Country = "US" // Country Code
    },
    PaymentStatus = OmniService.PaymentStatus.PreApproved,
    ShippingStatus = OmniService.ShippingStatus.NotYetShipped,
    ShipToAddress = new OmniService.Address()
    {
        Address1 = "Street 11",
        City = "Ghost Town",
        Country = "US" // Country Code
    },
    OrderLines = new OmniService.OrderLine[1],
    OrderPayments = new OmniService.OrderPayment[1]
};
```

If Click and Collect set **ClickAndCollectOrder** to **true** and set **CollectLocation** to the store where order is gone be collected at.

No ShipTo info is needed unless the Click and Collect orders is to be shipped, then add ShipTo info and set **ShipClickAndCollect** to **true**.

```
OmniService.Order order = new OmniService.Order()
{
    AnonymousOrder = false,
    ClickAndCollectOrder = true,
    CollectLocation = "S0001",
    StoreId = "S0013",
    CardId = "10023",
    ShipClickAndCollect = false,
    PaymentStatus = OmniService.PaymentStatus.PreApproved,
    ShippingStatus = OmniService.ShippingStatus.NotYetShipped,
    OrderLines = new OmniService.OrderLine[1],
};
```



## 6.5 Order Lines

A valid Order line needs to have ItemId, Quantity, Price and Amounts.

Order Line without discount:

```
order.OrderLines[0] = new OmniService.OrderLine ()
{
    ItemId = "40020",
    UomId = string.Empty,
    VariantId = "002",
    LineNumber = 1,
    LineType = OmniService.LineType.Item,
    Quantity = 2.0M,
    QuantityToInvoice = 2.0M,
    Price = 80.0M,
    NetPrice = 64.0M,
    Amount = 160.0M,
    NetAmount = 128.0M,
    TaxAmount = 32.0M
};
```

When adding a manual discount, a Discount Line needs to be added to the order. Both discount % and amount should be set. If the discount is applied from LS Central when Basket was sent in for calculation, then discount lines are already available in the return object from Calculation function.

Order Line with 10% discount:

```
order.OrderLines[0] = new OmniService.OrderLine ()
{
    ItemId = "40020",
    UomId = string.Empty,
    VariantId = "002",
    LineNumber = 1,
    LineType = OmniService.LineType.Item,
    Quantity = 2.0M,
    QuantityToInvoice = 2.0M,
    Price = 80.0M,
    NetPrice = 64.0M,
    DiscountAmount = 16.0M,
    DiscountPercent = 10.0M,
    Amount = 144.0M,
    NetAmount = 115.2M,
    TaxAmount = 28.8M
};
order.OrderDiscountLines[0] = new OmniService.OrderDiscountLine ()
{
    No = "10000",
    LineNumber = 1,
    DiscountPercent = 10.0M,
    DiscountAmount = 16.0M,
    DiscountType = OmniService.DiscountType.LineDiscount
};
```

## 6.6 Payments

Omni has its own Tender Type codes.

Default are: 0=Cash, 1=Card, 2=Coupon, 3=Loyalty Points, 4=Gift Card

Omni Tender Types are then mapped over to LS Nav/Central Tender Type code, and the mapping is configured in LS Omni Database – **Appsettings** table – Record with Key: **TenderType\_Mapping**.

**Do not set the FinalizedAmount parameter as that will cause LS Nav/Central to handle the order as it's been paid in full and Order cannot be posted. Payment process is still in development in LS Central, so it will change later.**

**Credit Card:** Send in **OrderPayment** line of type Card (1) and set the **PreApprovedAmount** to the value that should be authorized for the Order and **AuthorisationCode** for the token for the payment that will be used to request the final payment.

```
order.OrderPayments[0] = new OmniService.OrderPayment()
{
    LineNumber = 1,
    AuthorisationCode = "123456",
    CardType = "VISA",
    CardNumber = "10xx xxxx xxxx 1475",
    TenderType = "1",
    CurrencyCode = "GBP",
    CurrencyFactor = 1,
    FinalizedAmount = 0,
    PreApprovedAmount = 160.0M
    PreApprovedValidDate = new DateTime(2019,01,01),
};
```

**Loyalty Points:** Send in **OrderPayment** line of type Cash (0) and Currency Code **“LOY”**. LS Nav/Central treats Loyalty points as Currency and has special Currency record with Currency Rate to handle Point payment. The Rate tells how much value each point has in cash value and can be fetched by calling **GetPointRate**.

```
order.OrderPayments[0] = new OmniService.OrderPayment()
{
    PreApprovedAmount = 100,
    TenderType = "3",
    CardNumber = "10023",
    CurrencyCode = "LOY",
    CurrencyFactor = 0.1M
};
```

**Gift Card:** Send in **OrderPayment** line of type Gift Card (4) and Gift Card number in **CardNumber** and Gift card amount to be used in **PreApprovedAmount**.

```
order.OrderPayments[0] = new OmniService.OrderPayment()
{
    PreApprovedAmount = 100,
    TenderType = "4",
    CardNumber = "123456",
    CurrencyFactor = 1
};
```

## 6.7 Order History

To get history overview of all Orders and Sales Transactions for a Member Contact, use the function **OrderHistoryByContactId**. Parameters **includeLines** and **includeTransactions** can be used to filter the history result.

If **includeLines** is set to **true**, the result includes **item**, **payment** and **discount** lines.

If **includeTransactions** is set to **true**, the result includes all other sales transactions but merged into **Order** object.

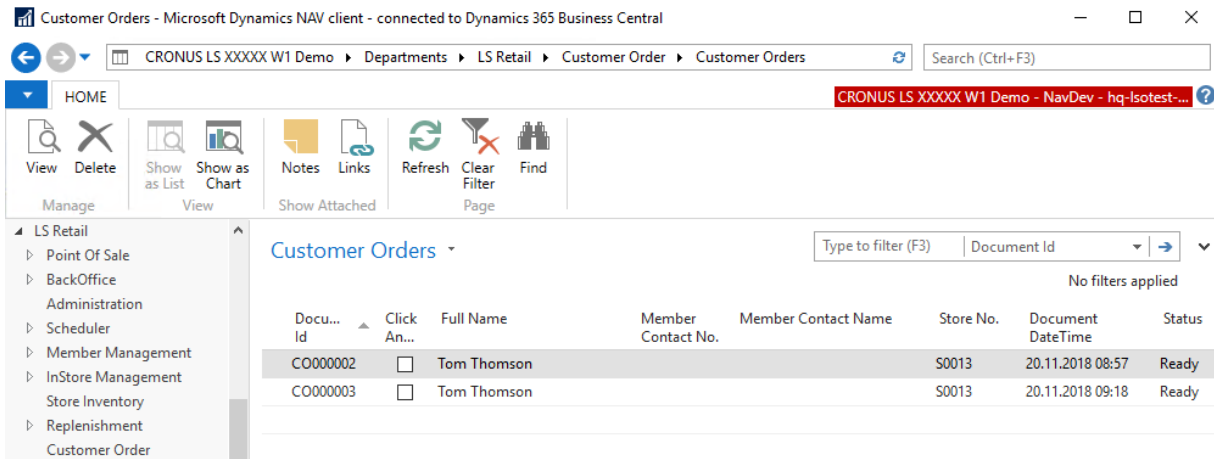
To get detailed data for the order use **OrderGetById** with **Order.DocumentId**. To get detailed data for the other sales transactions use **TransactionGetByReceiptNo** with **Order.ReceiptNo**. What separates the regular orders from the sales transactions is the **ReceiptNo** property, it's empty in the regular orders.

For anonymous users that don't have account, they must provide the Receipt Number for the Sales transaction to be used with **TransactionGetByReceiptNo** or Web Order Id (**Order.Id**) to be used with **OrderGetByWebId**.

## 6.8 Orders in LS Nav/Central

Orders can be found in LS Nav/Central under ->Departments/LS Retail/Customer Order/Customer Order

For LS Nav 12 and older go to ->Departments/LS Retail/Web Integration/Click and Collect/Orders

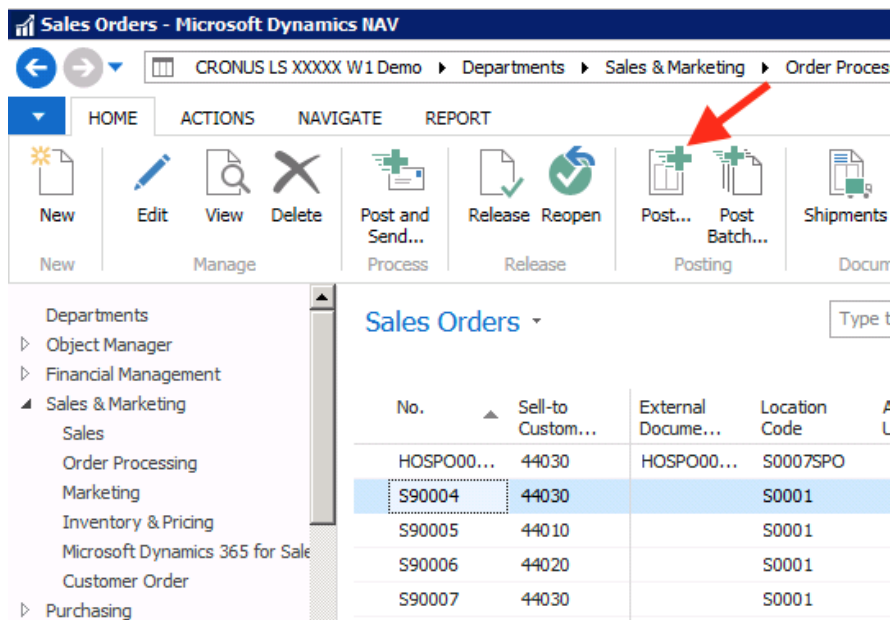


If order is to be shipped it should have been assigned automatically a Sales Order No.

### 6.8.1 Sales Order

When an order has been created, it can be found under ->Departments/Sales & Marketing/Order Processing/Sales Orders

There it can be processed for invoicing and shipping. To do that, select the order in the Sales Orders list and press the “Post” button.



To complete the process, select an option and press “Ok”.

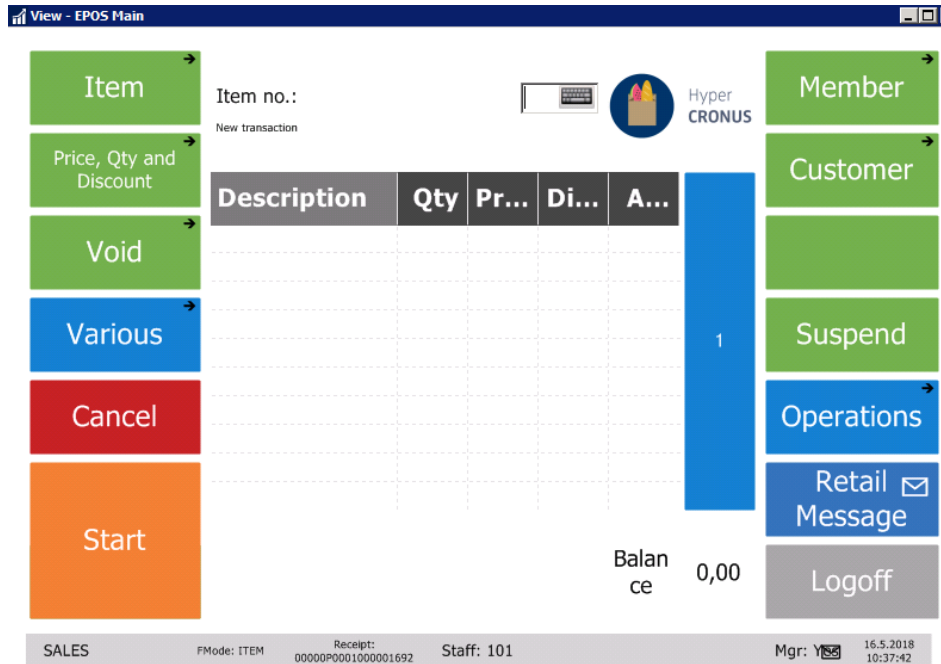
## 6.8.2 Click And Collect Order

To process Click and Collect orders, Open the Retail POS in LS Nav/Central

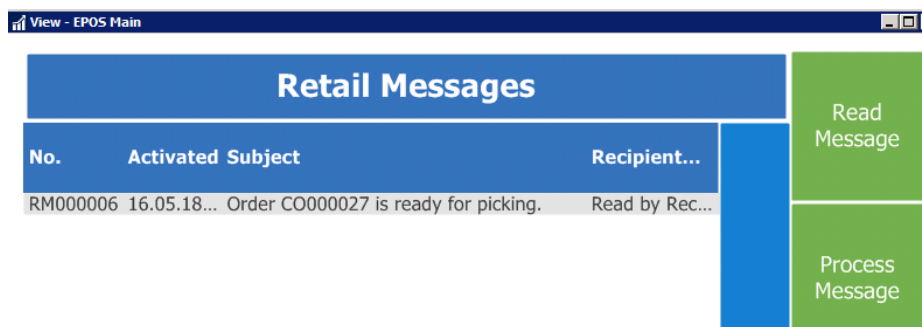
->Departments/LS Retail/Point Of Sale/POS ->Run Client

Click Logon Button, and Click 101 Flynn button to log in.

If there is a CAC order ready, a message will be waiting in Retail Message



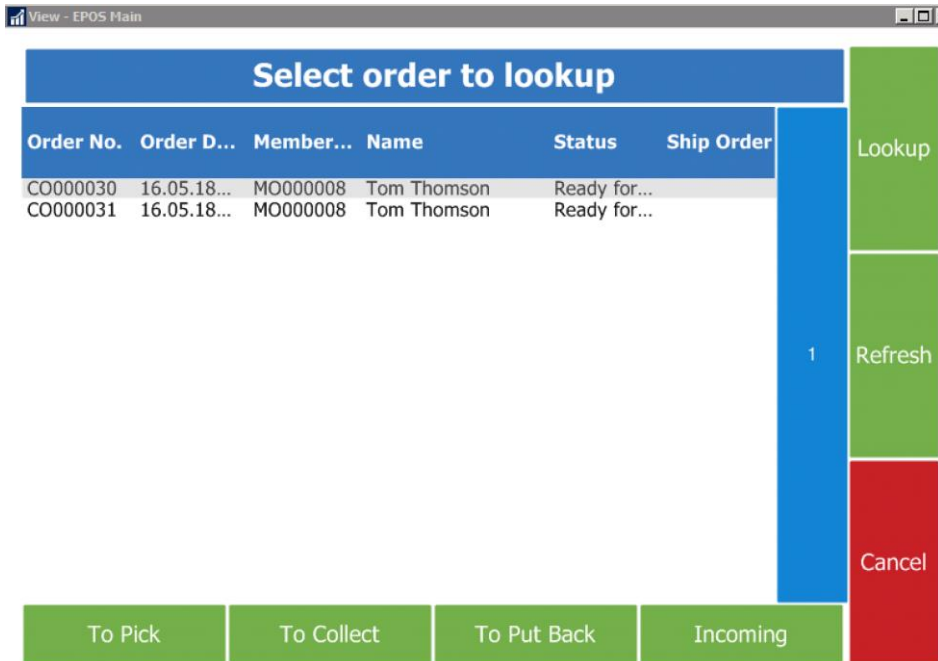
Click on Retail Message Button.



Select the order and Click Process Message Button.

This will show the order and there you can mark item that are being collected or do other work on the order, and then Confirm it as being ready for pickup.

When customer comes to pick up the order, click on Operations button in the main POS Screen, and then click Customer Orders.



View - EPOS Main

### Select order to lookup

Order No.	Order D...	Member...	Name	Status	Ship Order
CO000030	16.05.18...	MO000008	Tom Thomson	Ready for...	
CO000031	16.05.18...	MO000008	Tom Thomson	Ready for...	

1

Lookup

Refresh

Cancel

To Pick To Collect To Put Back Incoming

This shows all orders ready to be picked up. Pick an order and press To Collect and it will then show up in the POS as normal sale, process the sale to complete the process.

## 7 Other Code Samples

### 7.1 Get Image for Item

```
OmniService.ImageView img = serviceClient.ImageGetById("10020",
    new OmniService.ImageSize() { Width = 4096, Height = 4096 });

Image1.ImageUrl = "data:image/jpeg;base64," + img.Image;

System.Drawing.Image im = System.Drawing.Image.FromStream(
    new MemoryStream(Convert.FromBase64String(img.Image)));
im.Save(@"c:\temp\img999.bmp");
```

### 7.2 Get Image Stream for Item

```
Stream str = serviceClient.ImageStreamGetById("10020", 4096, 4096);

int data;
List<byte> imgbyte = new List<byte>(4096 * 4096);
while ((data = str.ReadByte()) != -1)
{
    imgbyte.Add((byte) data);
}
Byte[] bitmapData = imgbyte.ToArray();
string imgstr = Convert.ToBase64String(bitmapData, 0, bitmapData.Length);

Image1.ImageUrl = "data:image/jpeg;base64," + imgstr;

System.Drawing.Image im = System.Drawing.Image.FromStream(new MemoryStream(bitmapData));
im.Save(@"c:\temp\img998.bmp");
```