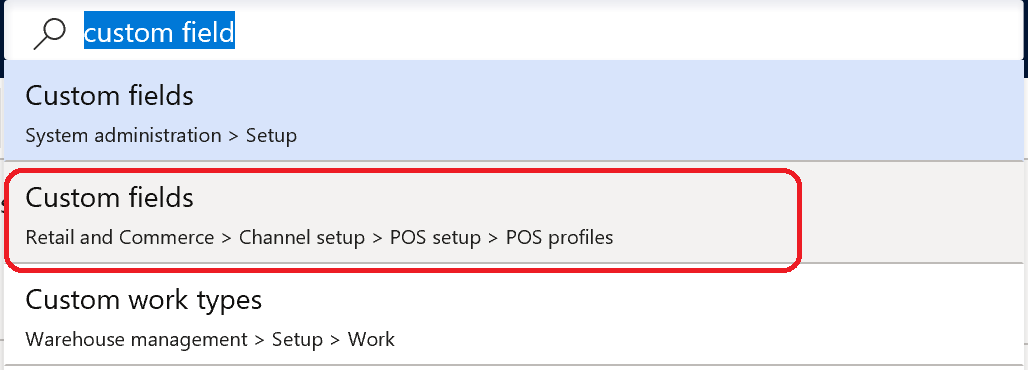
Goal:

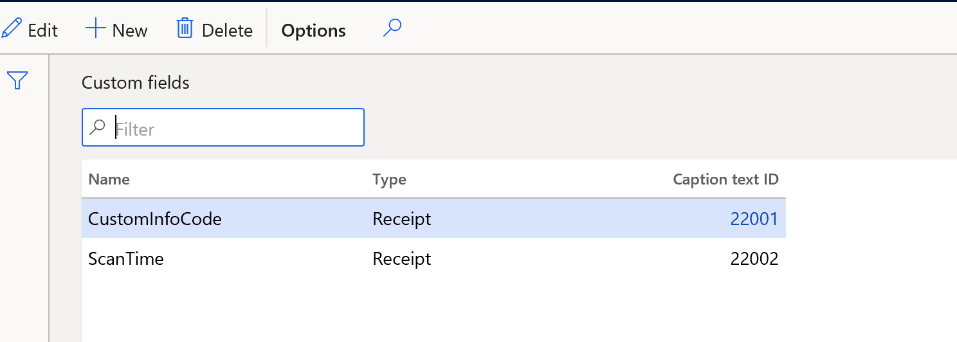
How to add a custom field to the POS receipt?

This document will take “Customer account deposit” receipt as an example.

Steps:

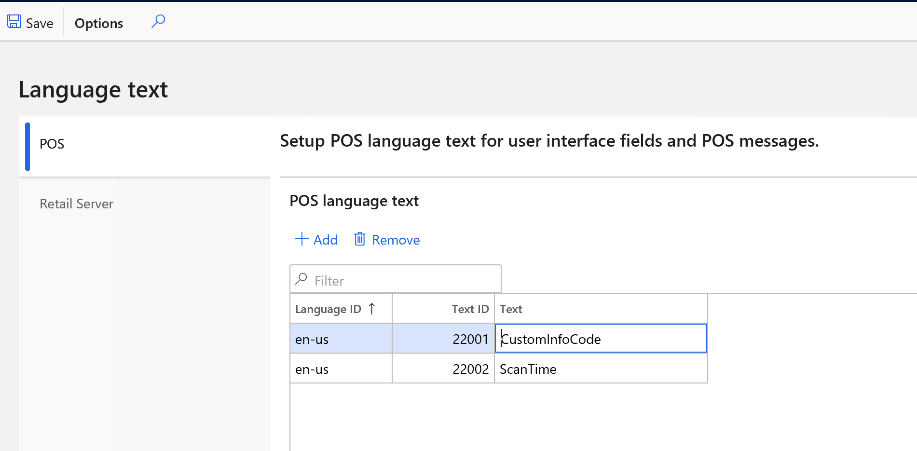
1. Create custom fields:

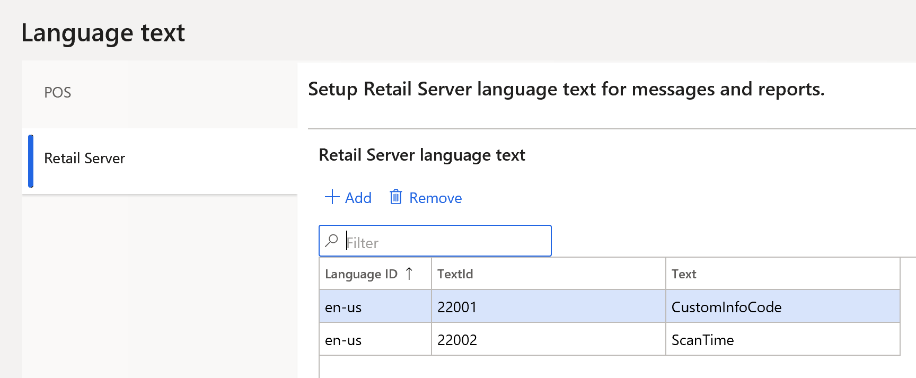




1. Before creating custom fields, maybe you need create Caption text ID firstly

Please go to this menu: https://usnconeboxax1aos.cloud.onebox.dynamics.com/? cmp=usrt&mi=RetailLanguageText

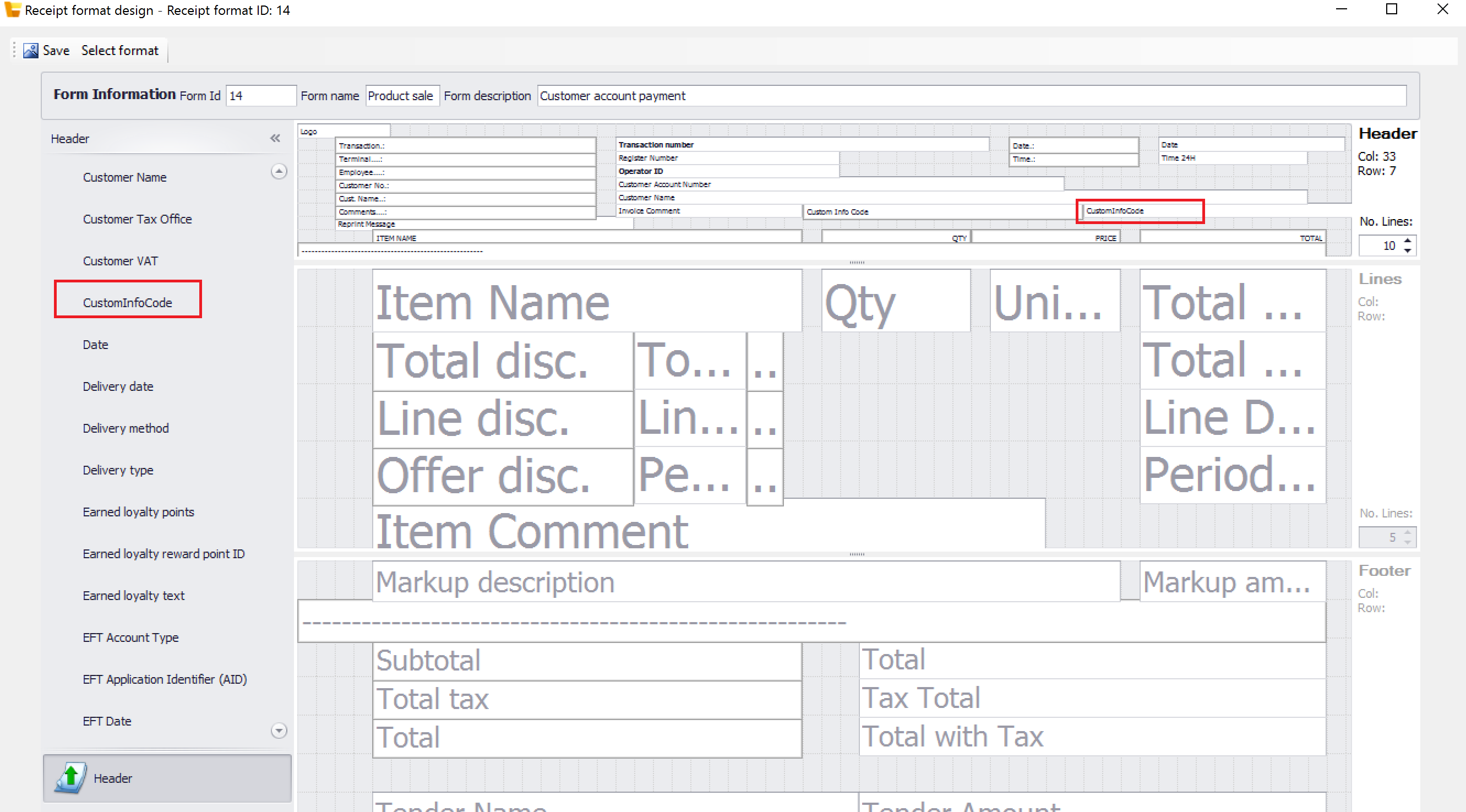




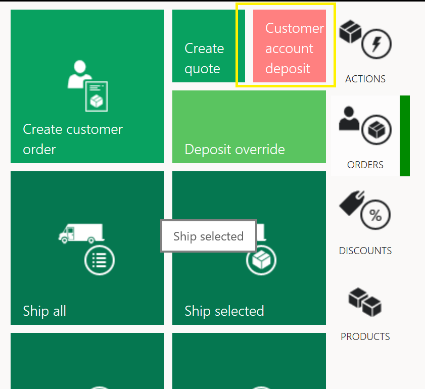
I did not test POS/Retail Server , which is mandatory, so just create both for them.

1. As per the document, the custom fields are available for header/body/footer for receipt designer.

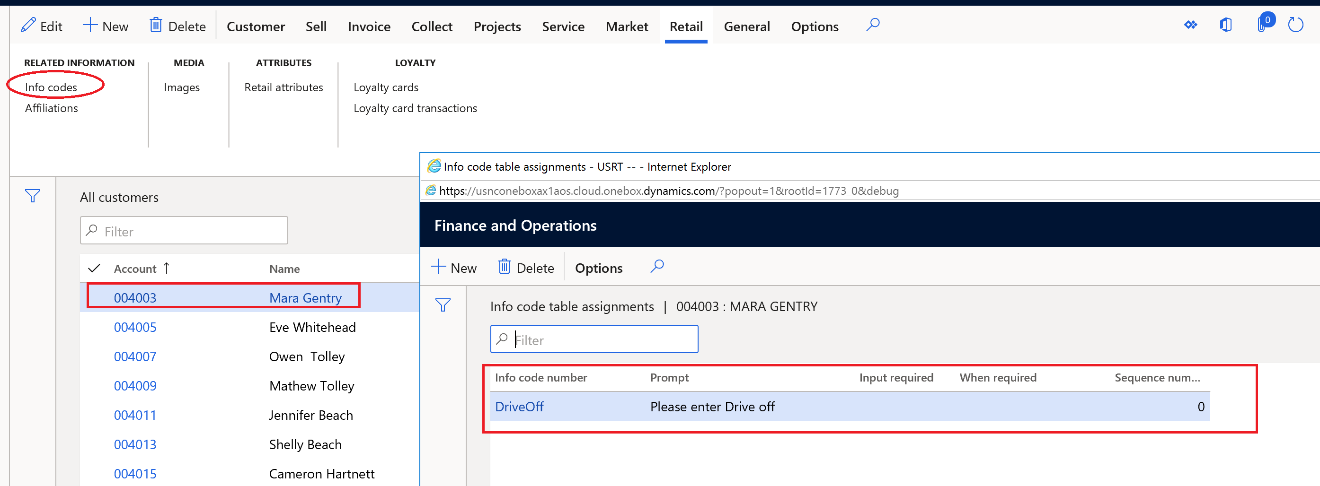
So go to receipt designer and add the custom fields to header:



1. Run 1090 job to push the change to channel database.
2. Create customer account deposit transaction on MPOS, you need add the button



And set customer info code:



1. After that, you need implement :

GetSalesTransactionCustomReceiptFieldServiceRequest

GetCustomReceiptFieldServiceResponse

To fetch the value of custom field and put it on the receipt:

|  |
| --- |
| public class GetCustomReceiptFieldsService : IRequestHandler  {  /// <summary>  /// Gets the supported request types.  /// </summary>  public IEnumerable<Type> SupportedRequestTypes  {  get  {  return new[]  {  typeof(GetSalesTransactionCustomReceiptFieldServiceRequest),  };  }  }  /// <summary>  /// Executes the requests.  /// </summary>  /// <param name="request">The request parameter.</param>  /// <returns>The GetReceiptServiceResponse that contains the formatted receipts.</returns>  public Response Execute(Request request)  {  if (request == null)  {  throw new ArgumentNullException("request");  }  Type requestedType = request.GetType();  if (requestedType == typeof(GetSalesTransactionCustomReceiptFieldServiceRequest))  {  return this.GetCustomReceiptFieldForSalesTransactionReceipts((GetSalesTransactionCustomReceiptFieldServiceRequest)request);  }  throw new NotSupportedException(string.Format("Request '{0}' is not supported.", request.GetType()));  }  /// <summary>  /// Gets the custom receipt filed for all transaction-based receipts, such as SalesReceipt, CustomerOrderReceipt, PickupReceipt, CreditCardReceipt, and so on.  /// </summary>  /// <param name="request">The service request to get custom receipt filed.</param>  /// <returns>The value of custom receipt filed.</returns>  private GetCustomReceiptFieldServiceResponse GetCustomReceiptFieldForSalesTransactionReceipts(GetSalesTransactionCustomReceiptFieldServiceRequest request)  {  string receiptFieldName = request.CustomReceiptField;  SalesOrder salesOrder = request.SalesOrder;  SalesLine salesLine = request.SalesLine;  TenderLine tenderLine = request.TenderLine;  // Get the store currency.  string currency = request.RequestContext.GetOrgUnit().Currency;  string reasonCodeStr = string.Empty;  /\* if (salesOrder.ExtensibleSalesTransactionType != ExtensibleSalesTransactionType.IncomeExpense || salesOrder.ExtensibleSalesTransactionType != ExtensibleSalesTransactionType.CustomerAccountDeposit)  {  reasonCodeStr = GetReasonCodesReceiptPrintText(salesOrder.ReasonCodeLines, request.RequestContext);  }\*/  string returnValue = string.Empty;  switch (receiptFieldName)  {  case "TIPAMOUNT":  {  // FORMAT THE VALUE  decimal tipAmount = salesOrder == null ? 0 : (salesOrder.TotalAmount \* 0.18m);  returnValue = this.FormatCurrency(tipAmount, currency, request.RequestContext);  }  break;  case "ITEMNUMBER":  {  returnValue = salesLine == null ? string.Empty : "Custom\_" + salesLine.ItemId;  }  break;  case "TENDERID":  {  returnValue = tenderLine == null ? string.Empty : "Custom\_" + tenderLine.TenderTypeId;  }  break;  case "CUSTOMINFOCODE":  {  reasonCodeStr = GetReasonCodesReceiptPrintText(salesOrder.ReasonCodeLines, request.RequestContext);  returnValue = reasonCodeStr;  }  break;  }  return new GetCustomReceiptFieldServiceResponse(returnValue);  } |

1. Finally the custom field will be printed on the receipt:

