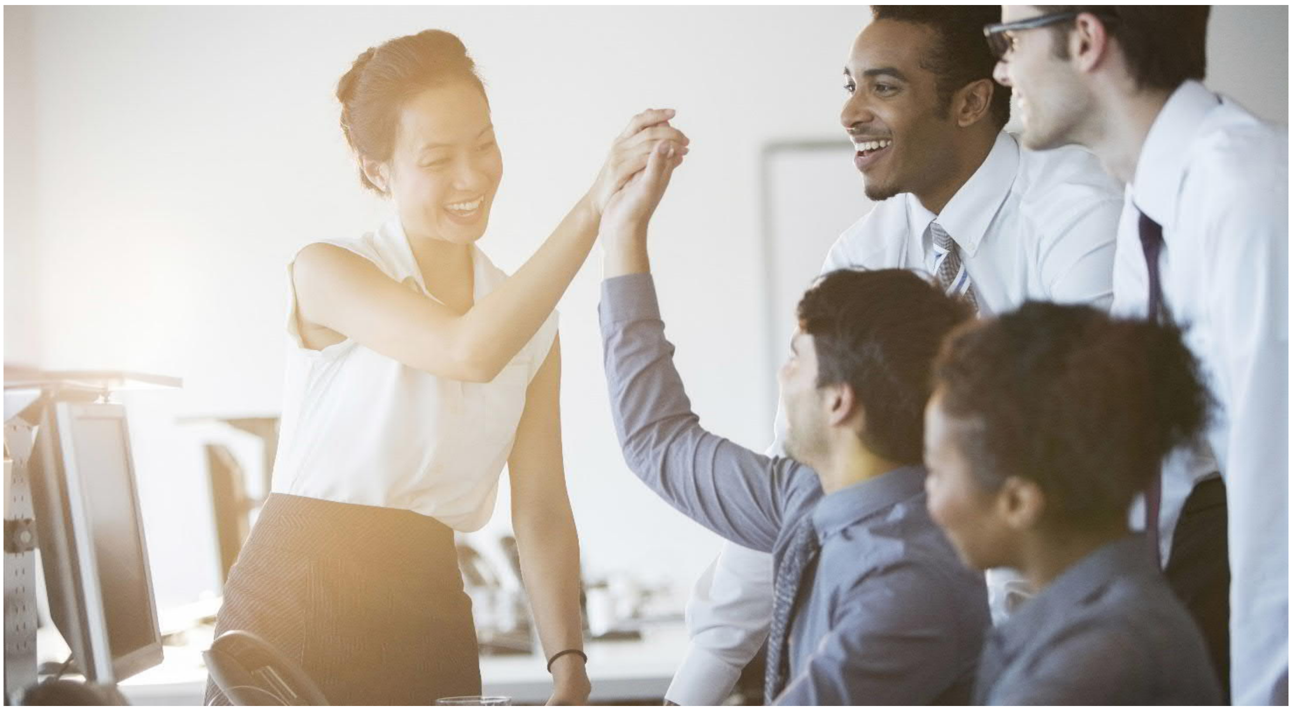


Exercise Solution: Documents Object

Data Interface API

PUBLIC

INTRODUCTION

In this exercise, you will perform the following tasks:

1. Create new buttons in your project for "Order, Invoice and Payment"
2. Create s Sales Order
3. Create a Sales Invoice based on the Sales Order
4. Create an Incoming Payment for the Sales Invoice

PREREQUISITE:

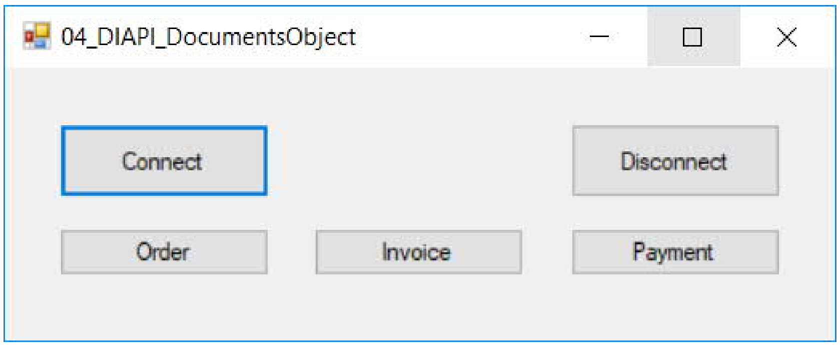
- This document is using the **C Sharp** (C#) language
- This document is using the Microsoft Visual Studio 2015
- Continue to work with the project finalized in previous exercise.
- Use the demo database for SAP Business One, version for SAP HANA or SAP Business One
- Credentials: User code: **manager**

GUIDELINES:

The screenshots provided here are for your reference only and may differ from the actual screenshots in your system.

1. TASK - CREATE NEW BUTTONS IN YOUR PROJECT FOR "ORDER, INVOICE AND PAYMENT"

1.1. On your Visual Studio project create new buttons called “Order, Invoice and Payment”



2. TASK - CREATE S SALES ORDER

2.1. Create a new *Document* object instance for the Sales Order. Then you set the properties of the Documents object and the *Documents_Lines*.

```
SAPbobsCOM.Documents oSO;

oSO = (SAPbobsCOM.Documents)oCompany.GetBusinessObject
(SAPbobsCOM.BoObjectTypes.oOrders);
oSO.CardCode = "C20000";
oSO.DocDueDate = DateTime.Today;
oSO.Comments = "this is my sales order’s comment";
oSO.Lines.ItemCode = "A00001";
oSO.Lines.Quantity = 2;
oSO.Lines.Price = 100;
oSO.Lines.Add();
oSO.Lines.ItemCode = "A00002";
oSO.Lines.Quantity = 1;
oSO.Lines.Price = 50;
```

2.2. Define a variable for the *Order* object – ensure it is defined as a member of the add-on application class or globally.

```
string MySalesOrder;
```

2.3. Add the whole document. In the case of success, you should bring up a message box telling the user the number of the newly added Sales Invoice using the method *GetNewObjectCode*. In case of any error, you should display a message box with an error message.

```
int ret = oSO.Add();
if (ret == 0)
{
      oCompany.GetNewObjectCode(out MySalesOrder);
      MessageBox.Show("Add Sales Order successful - " + MySalesOrder); }
 else
 {
      MessageBox.Show("Add Sales Order failed: " +
oCompany.GetLastErrorDescription());
 }
```

2.4. Finally, you should release the *Document* object variables.

```
System.Runtime.InteropServices.Marshal.ReleaseComObject(oSO);
oSO = null;
```

3. TASK - CREATE A SALES INVOICE BASED ON THE SALES ORDER

3.1. Create a new *Document* object instance for the Sales Invoice. Then you set the properties of the Documents object and the *Documents_Lines*.

```
SAPbobsCOM.Documents oSO;
SAPbobsCOM.Documents oInv;
```

3.2. To create a document based on a document you need to utilize the properties *BaseEntry* (DocEntry of Base document), *BaseType* (in this case Sales Order), *BaseLine* (line you wish to copy to target document)

```
oSO =
(SAPbobsCOM.Documents)oCompany.GetBusinessObject(SAPbobsCOM.BoObjectTypes.oOrders);
oInv =
(SAPbobsCOM.Documents)oCompany.GetBusinessObject(SAPbobsCOM.BoObjectTypes.oInvoices)
;

oSO.GetByKey(Int32.Parse(MySalesOrder));
oInv.CardCode = oSO.CardCode;

for (int i = 0; i <= oSO.Lines.Count - 1; i++)
{
      oInv.Lines.BaseEntry = oSO.DocEntry;
      oInv.Lines.BaseLine = i;
      oInv.Lines.BaseType = 17;
      oInv.Lines.Add();
}
```

3.3. Define a variable for the *Invoice* object – ensure it is defined as a member of the add-on application class or globally.

```
string MySalesInvoice;
```

3.4. Add the whole document. In the case of success, you should bring up a message box telling the user the number of the newly added Sales Invoice using the method *GetNewObjectCode*. In case of any error, you should display a message box with an error message

```
int ret = oInv.Add();

 if (ret == 0)
 {
      oCompany.GetNewObjectCode(out MySalesInvoice);
      MessageBox.Show("Add Invoice successfully");
 }
 else
      MessageBox.Show("Add Invoice failed: " + oCompany.GetLastErrorDescription());
```

3.5. Finally, you should release the *Document* object variables.

```
System.Runtime.InteropServices.Marshal.ReleaseComObject(oSO);
oSO = null;
System.Runtime.InteropServices.Marshal.ReleaseComObject(oInv);
oInv = null;
```

4. TASK - CREATE AN INCOMING PAYMENT FOR THE SALES INVOICE

4.1. Create a new *Payments* object instance for the Incoming Payment. Then you set the properties for the CardCode, Invoice DocEntry, and we will pay via cash, so we will use the properties *CashAccount* and *CashSum*.

```
       SAPbobsCOM.Payments oPay;
       oPay = (SAPbobsCOM.Payments)oCompany.GetBusinessObject
(SAPbobsCOM.BoObjectTypes.oIncomingPayments);
       SAPbobsCOM.Documents oInv;
       oInv = (SAPbobsCOM.Documents)oCompany.GetBusinessObject
(SAPbobsCOM.BoObjectTypes.oInvoices);

       oInv.GetByKey(Int32.Parse(MySalesInvoice));
       oPay.CardCode = oInv.CardCode;
       oPay.Invoices.DocEntry = oInv.DocEntry;
       oPay.CashAccount = "211100";
       oPay.CashSum = oInv.DocTotal;
```

4.2. Add the whole document. In the case of success, you should bring up a message box telling the user the number of the newly added Payment using the method *GetNewObjectCode*. In case of any error, you should display a message box with an error message.

```
int ret = oPay.Add();
if (ret == 0)
      MessageBox.Show("Add Payment successfully");
else

      MessageBox.Show("Add Payment failed: " + oCompany.GetLastErrorDescription());
```

4.3. Finally, you should release the *Document* object variables.

```
System.Runtime.InteropServices.Marshal.ReleaseComObject(oPay);
oPay = null;
System.Runtime.InteropServices.Marshal.ReleaseComObject(oInv);
oInv = null;
```

Another sample exercise can be found in the SDK samples (in the SDK Folder – see Appendix "SDK Installations" for more information), COM DI/5.OderAndInvoice.

www.sap.com