Solutions

Unit: Data Interface API

Topic: Establish a Connection to SAP Business One

1-1 Log on to SAP Business One.

1-1-1 Note the name of one database you want to log on to E.g. SBODemo_UK

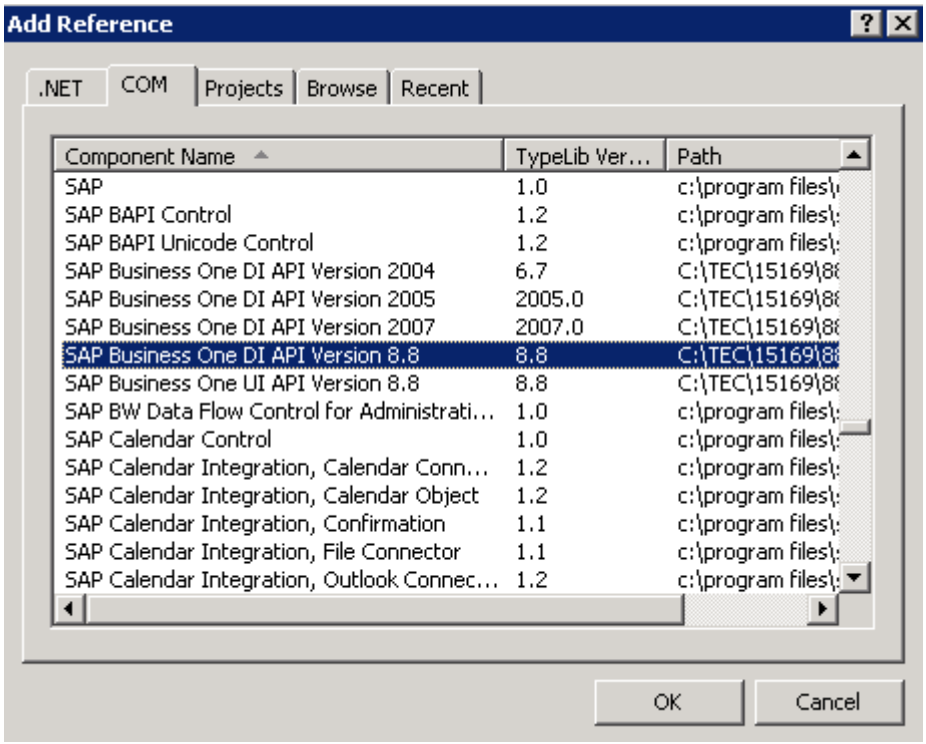
1-1-2 Note one user in that database and the user's password E.g. Manager, Manager

1-2 Create a new Visual Studio project.

1-2-1 Within this project, create a form with two buttons on it. One of the buttons should be used to connect to the SAP Business One database, the other to disconnect from it.

1-2-2 Add a reference to the SAP Business One DI API COM library…

Click Project -> Add Reference and click on the COM tab



1-3 Code the connection to the SAP Business One database.

1-3-1 Define a variable for the Company object – ensure it is defined as a member of the add-on application class or globally. Suggestion: Create a new module and put the Company variable there. Since this is a separate module you need to either specify the module in each call or add a declaration so the other form/modules can see this.

```
Public oCompany As SAPbobsCOM.Company
```

1-3-2 Create a new Company object.

```
oCompany = New SAPbobsCOM.Company
```

1-3-3 Set the properties needed to connect to the SAP Business One database.

```
oCompany.Server = "Your server name"

oCompany.CompanyDB = "SBODemo_UK"

oCompany.UserName = "manager"

oCompany.Password = "B1Admin"

oCompany.DbServerType =
SAPbobsCOM.BoDataServerTypes.dst_MSSQL2005

oCompany.DbUserName = "sa"

oCompany.DbPassword = "sapass"

oCompany.LicenseServer = "Your license server name"
```

Note DBUserName and DBPassword are not required in Version 8.8.

1-3-4 Call connect on the Company object

```
retVal = oCompany.Connect
```

1-4 Implement error handling and success handling.

1-4-1 If the connection succeeds, display a message box displaying a corresponding message.

1-4-2 If the connection failed, display the error message provided by the Company object.

```
If retVal <> 0 Then

   oCompany.GetLastError(retVal, retStr)

   MsgBox("Error " & retVal & " " & retStr)

Else

   MsgBox("Connected to " & oCompany.CompanyName)

End If
```

1-5 Code the disconnection from the SAP Business One database.

```
If oCompany.Connected = True Then

   oCompany.Disconnect()

End If
```

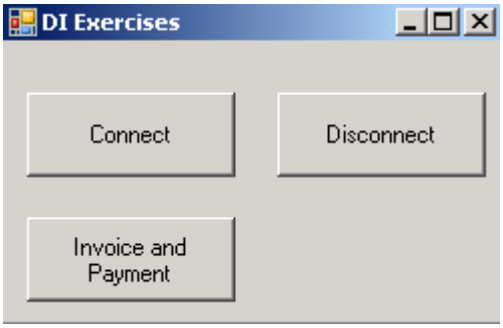
A further sample can be found in the SDK DI samples (in the SDK Folder – see Appendix "SDK Installations" for more information), COM DI/1.BasicOperations.

Solutions

Unit: Data Interface API

Topic: Documents Object

2-1 On your Visual Studio project create a new button called “Invoice and Payment”



2-2 In Business One create an Order for a particular customer and a particular item.

2-2-1 First you must create a new Document object instance for the Invoice. Then you set the properties of the Documents object and the Documents_Lines ensuring the BaseEntry, BaseLine and BaseType are set.

```
Dim oInvoice As SAPbobsCOM.Documents

oInvoice = oCompany.GetBusinessObject(SAPbobsCOM.BoObjectTypes.oInvoices)

oInvoice.CardCode = "C2000"

oInvoice.Lines.BaseEntry = 8 'DocEntry of Sales Order

oInvoice.Lines.BaseLine = 0 'Copy first line

oInvoice.Lines.BaseType = 17 'Sales Order base document
```

2-2-2 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 Sales Invoice using the method GetNewObjectCode. In case of any error, you should display a message box with an error message.

```
retVal = oInvoice.Add

If retVal <> 0 Then

    oCompany.GetLastError(retVal, retStr)

    MsgBox("Error " & retVal & " " & retStr)

Else

    MsgBox("Invoice number " &
oCompany.GetNewObjectKey & " created")

    InvNum = oCompany.GetNewObjectKey

End If
```

2-2-3 Finally you should release the document object variables.

```
oInvoice = Nothing

retVal = ""

retStr = ""
```

2-3 Create the Incoming Payment for this Invoice

2-3-1 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.

```
Dim oIncomingPymt As SAPbobsCOM.Payments

        oIncomingPymt =
oCompany.GetBusinessObject(SAPbobsCOM.BoObjectTypes.oInco
mingPayments)

        oIncomingPymt.CardCode = "C2000"

        oIncomingPymt.Invoices.DocEntry = InvNum

        oIncomingPymt.CashAccount = "_SYS00000000076"

        oIncomingPymt.CashSum = "14.10"
```

Note: CashAccount "_SYS…" uses an internal account number in a database where account segmentation is used. If Account segmentation is **not** used – just use the visible account numbers.

2-3-2 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.

```
       retVal = oIncomingPymt.Add

        If retVal <> 0 Then

            oCompany.GetLastError(retVal, retStr)

            MsgBox("Error " & retVal & " " & retStr)

        Else

            MsgBox("Incoming Payment number " &
oCompany.GetNewObjectKey & " added")

        End If
```

2-3-3 Finally you should release the document object variables.

```
oIncomingPymt = Nothing

retVal = ""

retStr = ""
```

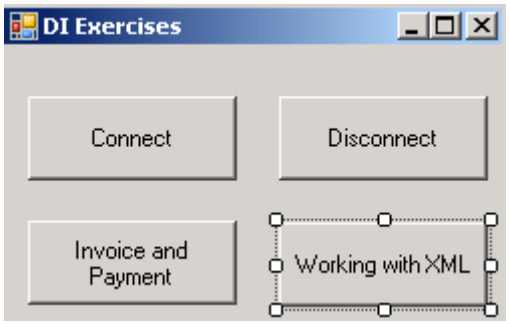
Another sample exercise can be found in the SDK samples (in the SDK Folder – see Appendix "SDK Installations" for more information), COM DI/5.OderAndInvoice.

Solutions

Unit: Data Interface API

Topic: XML

3-1 On your Visual Studio project create a new button called “Working with XML”



3-2 Save the Invoice created in the Documents exercise as XML.

3-2-1 Try all settings for XmlExportType property on the Company object and find the differences.

The 4 property types are

XML ExportType	**Definition**
oCompany.XmlExportType = SAPbobsCOM.BoXmlExportTypes.xet_AllNodes	Export to XML all fields (both read only and read/write fields) from the database.
oCompany.XmlExportType = SAPbobsCOM.BoXmlExportTypes.xet_ExportImportMode	Export to XML only valid fields that support XML import (read/write fields only) from the database.
oCompany.XmlExportType = SAPbobsCOM.BoXmlExportTypes.xet_NodesAsProperties	Export to XML all fields as properties from the database.
SAPbobsCOM.BoXmlExportTypes.xet_ValidNodesOnly	Export to XML only valid fields that support XML import and export (read/write fields only that do not contain null values) from the database.

3-2-2 Save the Invoice document created in the previous exercise in Xml format

```
        oCompany.XmlExportType =
SAPbobsCOM.BoXmlExportTypes.xet_ExportImportMode

        Dim oInvoice As SAPbobsCOM.Documents

        oInvoice =
oCompany.GetBusinessObject(SAPbobsCOM.BoObjectTypes.oInvo
ices)

        If oInvoice.GetByKey(5) = False Then

            oCompany.GetLastError(retVal, retStr)

            MsgBox("Failed to Retrieve Invoice" & retVal
& " " & retStr)

            Exit Sub

        End If

        'Save the object as an xml file

        oInvoice.SaveXML("C:\Program Files\SAP\SAP
Business One SDK\Samples\CourseXML\Invoice.xml")
```

3-2-3 Test also the method GetBusinessObjectXmlSchema of the Company object. What kind of information does it save?

```
        Dim schema As String

        schema =
oCompany.GetBusinessObjectXmlSchema(SAPbobsCOM.BoObjectTy
pes.oInvoices)

        MsgBox(schema)
```

This method retrieves the XML schema used to define the structure and content of the object.

3-3 Modify the XML data obtained before and add it to the SAP Business One database.

```
        oInvoice = oCompany.GetBusinessObjectFromXML("C:\Program
Files\SAP\SAP Business One SDK\Samples\CourseXML\Invoice.xml",
0)

        retVal = oInvoice.Add

        If retVal <> 0 Then

            oCompany.GetLastError(retVal, retStr)

            MsgBox("Error " & retVal & " " & retStr)

        Else

            MsgBox("Invoice number " & oCompany.GetNewObjectKey
& " created")

        End If
```

3-3-1 Try all files generated above and check the errors (exceptions) for details.

Similar exercises can be found in the SDK samples (in the SDK Folder – see Appendix "SDK Installations" for more information), COM DI/7.SaveXML and COM DI/8.LoadFromXML

Solutions

Unit: Data Interface API

Topic: Transactions

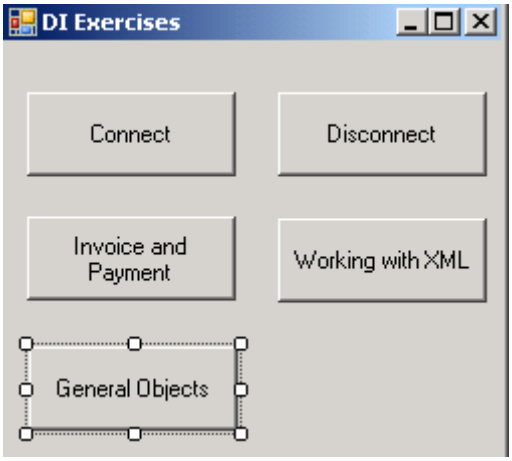
There is no additional “Solution” to this exercise.

Solutions

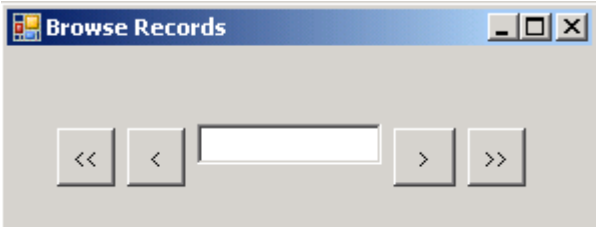
Unit: Data Interface API

Topic: Using General Objects

5-1 On your Visual Studio project create a new button called “General Objects”



5-2 Create a new form in your Visual Studio application containing a text box where you will show the Business Partners Card Code and four buttons: first, previous, next and last.



5-3 Create a Recordset object and set the Browser property of the BusinessPartners object to this Recordset. Be sure that your application only includes customers, not Leads or Vendors (Suppliers).

```
oRecordSet = oCompany.GetBusinessObject(SAPbobsCOM.BoObjectTypes.BoRecordset)

oBusinessPartner = oCompany.GetBusinessObject(SAPbobsCOM.BoObjectTypes.oBusinessPartners)

oRecordSet.DoQuery("Select CardCode from OCRD where CardType = 'C'")

oBusinessPartner.Browser.Recordset = oRecordSet
```

5-3-1 Add the code to all four of the buttons so that the user can navigate backwards and forward through the customers.

For example to move First use the following code. It needs to be changed slightly for the other 3 actions.

```
If oBusinessPartner.Browser.BoF = False Then

    oBusinessPartner.Browser.MoveFirst()

    FillField()

 End If
```

5-4 Test your changes. Be sure to include the following scenarios:

5-4-1 Click the “First Record” button (), then click it again. Try the same thing with the “Last Record” button ().

5-4-2 Click the “First Record” button (), then click the “Previous Record” button ().

5-4-3 Click the “Last Record” button (), then click the “Next Record” button ().

5-4-4 If any of these scenarios raises an error, add code that will fix the error. Then test the application again.

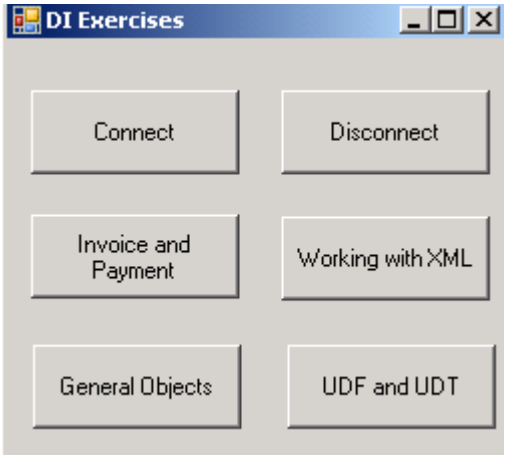
A similar exercise can be found in the SDK samples (in the SDK Folder – see Appendix “SDK Installations” for more information), COM DI/1.BasicOperations

Solutions

Unit: Data Interface API

Topic: Meta Data

6-1 As a first small exercise add a User-Defined Field to the item table (OITM) through DI API. On your Visual Studio project create a new button called “UDF and UDT”



6-1-1 Use namespace “TB1_” as a prefix…

```
    Dim oUDF As SAPbobsCOM.UserFieldsMD
        oUDF =
oCompany.GetBusinessObject(SAPbobsCOM.BoObjectTypes.oUser
Fields)
        oUDF.TableName = "OITM"
        oUDF.Name = "TB1_Course"
        oUDF.Description = "Course UDF"
        oUDF.Type = SAPbobsCOM.BoFieldTypes.db_Alpha
        oUDF.EditSize = 20
        retVal = oUDF.Add
        If retVal <> 0 Then
            oCompany.GetLastError(retVal, retStr)
            MsgBox("Error " & retVal & " " & retStr)
        Else
            MsgBox("UDF Added")
        End If
        oUDF = Nothing
```

6-2 Add a User-Defined Table (use namespace “TB1_” as a prefix…), but do not add any fields to the table yet.

```
Dim oUsrTble As SAPbobsCOM.UserTablesMD

oUsrTble =
oCompany.GetBusinessObject(SAPbobsCOM.BoObjectTypes.oUserTables)

oUsrTble.TableName = "TB1_DVD"

oUsrTble.TableDescription = "DVD Management"

retVal = oUsrTble.Add

If retVal <> 0 Then

    oCompany.GetLastError(retVal, retStr)

    MsgBox("Error " & retVal & " " & retStr)

Else

    MsgBox("UDT Added")

End If

oUsrTble = Nothing
```

6-3 Test your application by opening the “Manage User Fields” window in SAP Business One. Check to see that the table was added.

6-4 Remove the User-Defined Table (in the SAP Business One application) you just created before. Enhance your application with the capability to remove the User-Defined Table through DI API – and then test your application to see that you can also add and delete the User-Defined Table in SAP Business One.

```
If oUsrTble.GetByKey("TB1_DVD") = True Then

    retVal = oUsrTble.Remove

End If

If retVal <> 0 Then

    oCompany.GetLastError(retVal, retStr)

    MsgBox("Error " & retVal & " " & retStr)

Else

    MsgBox("UDT Removed")

End If
```

6-5 Add the following User-Defined Fields to your new User-Defined Table:

Aisle Number – Indicates in which aisle the movie is stored.

- Field Name: AISLE
- Field Description: Aisle Number
- Field Type: db_Numeric
- Field EditSize: 2

Section – Indicates the section the movie is store in the aisle.

- Field Name: SECTION
- Field Description: Section Number
- Field Type: db_Alpha
- Field EditSize: 20

Rented – Indicates weather the movie is rented or not.
Holds 2 “valid values”: Y/N.

- Field Name: RENTED
- Field Description: Rented/Available
- Field Type: db_Alpha
- Field EditSize: 1

CardCode – In case the movie is “Rented”
This field will hold the CardCode of the customer who rented it otherwise it will be empty.

- Field Name: CARDCODE
- Field Description: Card Code
- Field Type: db_Alpha
- Field EditSize: 20

Same process as adding a user defined field to a System table except we use the correct notation for a User Defined Table i.e. using @

```
oUDF.TableName = "@TB1_DVD"
```

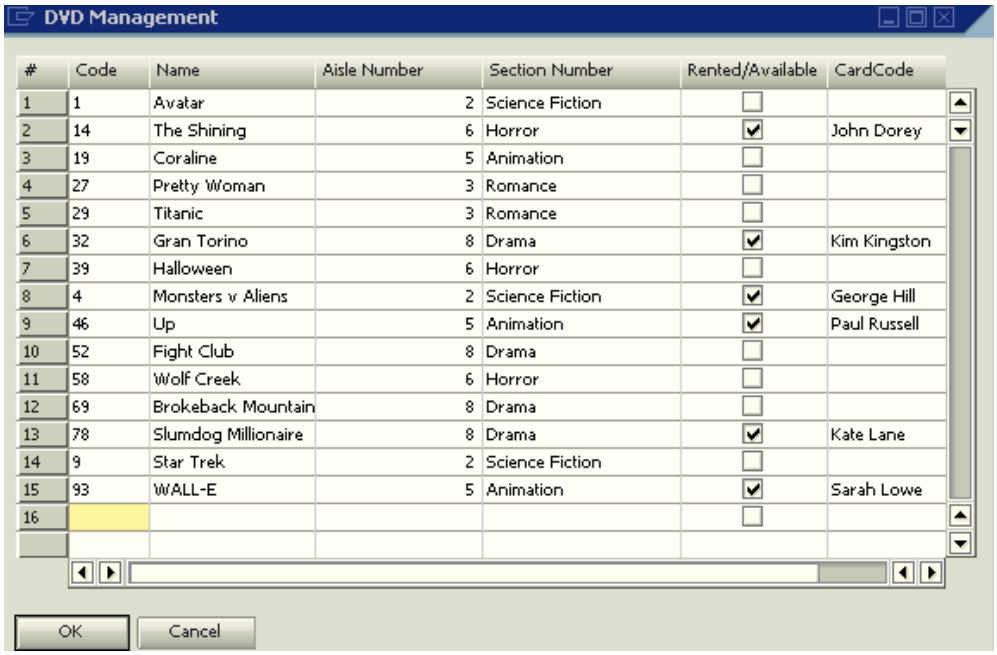
6-6 Test your application and make sure all your fields were added successfully.

6-7 Write data into the User-Defined Table.

6-7-1 Add about 15 records to your new User-Defined Table.

```
        Dim oUserTable As SAPbobsCOM.UserTable
        oUserTable = oCompany.UserTables.Item("TB1_DVD")
        oUserTable.Code = "1"
        oUserTable.Name = "Avatar"

oUserTable.UserFields.Fields.Item("U_AISLE").Value = "2"

oUserTable.UserFields.Fields.Item("U_SECTION").Value =
"Science Fiction"

oUserTable.UserFields.Fields.Item("U_RENTED").Value = "N"


        retVal = oUserTable.Add
        If retVal <> 0 Then
            oCompany.GetLastError(retVal, retStr)
            MsgBox("Error " & retVal & " " & retStr)
        Else
            MsgBox("Record Added")
        End If

        oUserTable = Nothing
```

6-7-2 Your User-Defined Table could look like this:



#	Code	Name	Aisle Number	Section Number	Rented/Available	CardCode
1	1	Avatar	2	Science Fiction	☐	
2	14	The Shining	6	Horror	☑	John Dorey
3	19	Coraline	5	Animation	☐	
4	27	Pretty Woman	3	Romance	☐	
5	29	Titanic	3	Romance	☐	
6	32	Gran Torino	8	Drama	☑	Kim Kingston
7	39	Halloween	6	Horror	☐	
8	4	Monsters v Aliens	2	Science Fiction	☑	George Hill
9	46	Up	5	Animation	☑	Paul Russell
10	52	Fight Club	8	Drama	☐	
11	58	Wolf Creek	6	Horror	☐	
12	69	Brokeback Mountain	8	Drama	☐	
13	78	Slumdog Millionaire	8	Drama	☑	Kate Lane
14	9	Star Trek	2	Science Fiction	☐	
15	93	WALL-E	5	Animation	☑	Sarah Lowe
16					☐	

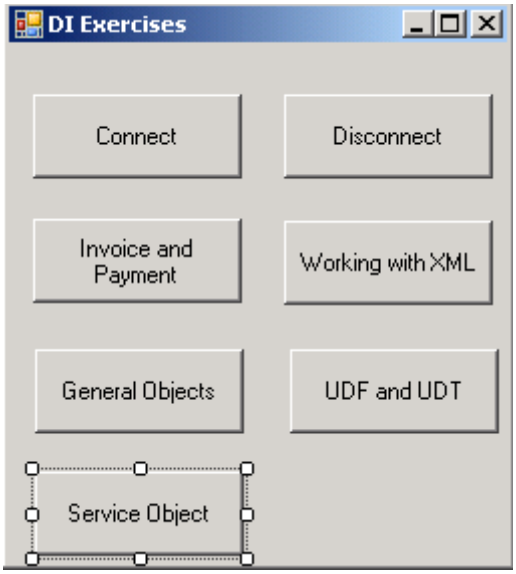
A similar solution can be found in the SDK samples (in the SDK Folder – see Appendix "SDK Installations" for more information), …\COM UI DI\VB.NET\UIDIBasicApp\CreateUserTables

Solution to Optional Exercise

Unit: Data Interface API

Topic: Services

7-1 On your Visual Studio project create a new button called “Service Object”



7-2 Get CompanyServices object.

7-3 Get structure which reflects information in table OADM.

7-4 Set the background color to purple.

7-5 Call the method which updates the information in the SAP Business One database. See the effect in the SAP Business One application

```
Dim oCompanyService As SAPbobsCOM.CompanyService
Dim oCompanyInfo As SAPbobsCOM.CompanyInfo
Dim oCompanyAdminInfo As SAPbobsCOM.AdminInfo

oCompanyService = oCompany.GetCompanyService
oCompanyAdminInfo = oCompanyService.GetAdminInfo
oCompanyAdminInfo.CompanyColor = 3

oCompanyService.UpdateAdminInfo(oCompanyAdminInfo)
```

A solution (+ more sample code around services) can be found in the SDK samples (in the SDK Folder – see Appendix "SDK Installations" for more information),
COM DI/ 11.Basic Company Settings