APPENDIX C: USING C/FRONT

Introduction

This appendix provides information on C/FRONT. However, this course is not a substitute for reading the C/FRONT Reference Guide. For information on C/FRONT installation and setup, please refer to the C/FRONT Reference Guide or the Microsoft Dynamics™ NAV Development I manual.

Microsoft Official Training Materials for Microsoft Dynamics ™
Your use of this content is subject to your current services agreement

C-1

Using C/FRONT

C/FRONT is an application programming interface used to access a C/SIDE database. C/FRONT facilitates high-level interaction with the C/SIDE database manager, and allows C developers to manipulate any C/SIDE database.

The central component of C/FRONT is a library of C functions. These functions provide access to every aspect of data storage and maintenance, and allow you to integrate both standard and custom applications with a C/SIDE database.

Since C/FRONT is a front end to Microsoft Dynamics NAV Attain, you may want to use C/FRONT to add, delete or view data in your own applications. Furthermore, you may need to create and manipulate table objects or to find out information from tables, which is something you cannot do when using the C/ODBC driver. C/FRONT is much faster than using the C/ODBC driver and is as close to the native connection as you can get with Microsoft Dynamics NAV Attain. However, to use C/FRONT, you must use some other programming language such as Delphi, Kylix, C#, C++, C, or Visual Basic. C/FRONT provides two methods of accessing Microsoft Dynamics NAV Attain:

- Through a C style DLL
- Through the use of the OCX interface.

The examples here use Microsoft® Visual Basic® 6.0 and the OCX.

Two Interfaces – DLL and OCX

Differences Between the C/FRONT API DLL and OCX

There are a number of differences between the C/FRONT API DLL and the OCX. The C/FRONT API DLL is primarily intended for C programmers, but can be used with any language and compiler that can load and use DLLs and that use the _CDECL calling convention. The OCX is primarily used for gaining easy access to Navision Attain databases from environments like Microsoft® Excel® and Microsoft® Word or Microsoft Visual Basic.

The C/FRONT Reference Guide is intended for the DLL and for the C programmer but can be used with the OCX as well. It provides some background information, and contains a considerable number of examples. If you are acquainted with the DLL, it is recommended that you read the online Help for a function before using through the OCX – there are a number of differences. Furthermore, some functions in the DLL have no equivalent in the OCX, and there are a couple of new functions in the OCX. Also note that errors are handled differently. For example, you cannot install your own error or message handler to replace the default one when you use the OCX.

General Differences

The following are the general difference between DCL and OCX:

- Where all the functions in the DLL are named DBL_*, the corresponding methods in the OCX do not use the DBL_ prefix. For example, the method that corresponds to the DBL_OpenDatabase function is OpenDatabase.

- The GetLastErrorCode in the DLL has an equivalent in the LastError method in the OCX.

- The Field_2_Str function in the DLL has an equivalent in the FieldToStr method in the OCX.

- Closing dates cannot be used in the OCX. If a closing date is retrieved from the database, it will be considered a normal date, and there is no way to put a closing date into the database from the OCX.

- There are no hundredths of seconds in Time values in the OCX.

- The Allow function is replaced by four methods − AllowKeyNotFound, AllowRecordExists, AllowRecordNotFound and AllowTableNotFound.

Functions Without Equivalents

There is no longer any need for the conversion functions, because functions such as GetFieldData and AssignField use the Variant data type. The following functions therefore have no equivalents in the OCX:

- Alpha_2_Str
- BCD_2_Str
- Date_2_Str
- Date_2_YMD
- HMST_2_Time
- Str_2_Alpha
- Str_2_BCD
- Str_2_Date
- Str_2_Time
- Time_2_HMST
- Time_2_Str
- YMD_2_Date

Error handling is different in the OCX (see Error Handling in the C/FRONT manual) and you cannot replace the default error or message handler with a function of your own. However, it is recommended that you make sure that there is plenty of error checking because the majority of errors that occur happen because incorrect values have been set. The following functions have therefore no equivalents in the OCX:

- SetExceptionHandler
- SetMessageShowHandler

When the OCX is used, there is no need for explicit initialization and deinitialization. Therefore, the following functions have no equivalents in the OCX:

- Exit
- Init

The following two functions have been omitted because the changed functionality of GetFieldData makes them less necessary and also because the environments from where the OCX is typically used does not support the use of pointers to the same degree as C does.

- GetFieldDataAddr
- GetFieldDataSize

The following function is still retained in the DLL in order to make it easier to port applications from the C-Toolkit for Microsoft Dynamics NAV 3.XX. This function is not needed in the OCX.

- FieldDataOffset

Accessing Data From the Database Using C/FRONT in Visual Basic

The following example assumes that you want to open a predefined database, the customer table, and to display the customers in a list form. The example does not explain how to use Visual Basic.

1. Open Visual Basic and choose the type of project (Standard.Exe) and then click **Open**.

2. Select Project1 inside the Project Window and right-click and select **Rename**. Rename your project to Cfront_Test.

3. Click on the Form1 in the Project Window and then select the properties. Set the following properties:

   Name = MainForm

   Height = 6450

   Width = 6650

   Startup Position = 2 – CenterScreen

4. Click **Projects** and select components or use the shortcut key sequence of (Ctrl + T). Next, select Cfront OLE Control Module, Microsoft Common Dialog SP3, and Microsoft FlexGrid Control 6.0 (SP3). Make sure that there is a check mark in the box beside each component name.

5. Declare some variables that you need. Click **View** from the main menu and select **Code**. The code window appears. Select (General) from the left drop-down list and (Declarations) from the right drop-down list and then create the following public variables.

```
Public NavisionPath As String
Public LicenseFileName As String
Public DatabaseFileName As String
Public CompanyName As String
Public UserId As String
Public TableNo As Long
Public TableHandle As Long
Public hRec As Long
Public GridRow As Integer
Public GridCol As Integer
```

Microsoft Official Training Materials for Microsoft Dynamics ™
Your use of this content is subject to your current services agreement

C-5

6. You need to create a menu. Click **Tools** from the main menu and select Menu Editor or press CTRL + E. Add the following menu options – File, Open, Exit, View, and Write with Open and Exit indented under file. Furthermore, note that each menu option must have a caption and a name. Therefore, give the menu options described and create the captions as shown in the box above.

7. Double-click on the **OCX** icon in the Components Window, this places the OCX (CFront) on the form. Select the **OCX** icon and move it up to the top left of the form. Next, set the following properties:

Visible = No
(Name) = Cfront

NOTE: You may want to shrink the size of the icon to where it is no larger than the command button.

---

8. Double-click on the CommonDialog icon in the Components Window; this places the CommonDialog component icon on the form. Select the CommonDialog component icon and move it to the top of the form on the right side of the Cfront icon. Change the name property from CommonDialog1 to CommonDialog.

9. Double-click on the MSFlexGrid icon in the components Window. This places the MSFlexGrid component on the form. Set the following properties:

```
(Name) = TableBox
Top = 520
Height = 5655
Width = 6495
Rows = 0
TabIndex = 0
```

10. Now it is time to add some code. Select **File** and then **Open** from the menu created. This should automatically open the Code window and create a procedure Open_Click. Add the following code to the Open_Click procedure.

```
'If connecting to a Navision database
DriverName = "NDBCN"
ServerName = ""
NetType = "tcp"
CacheSize = 1000
UseCommitCache = 0
NTAuthentication = 0
UserId = ""
Password = ""
```

```
'If connecting to a SQL Server database
'DriverName = "NDBCS"
'NetType = "Named Pipes"
'CacheSize = 0
'UseCommitCache = 0
'NTAuthentication = 1 'Yes
```

NOTE: If you are using Microsoft® SQL Server® you have to enter a zero value for CacheSize or UseCommitCache even though they only apply to the Microsoft Dynamics NAV Server.

```
CommonDialog.ShowOpen
DatabaseFileName = CommonDialog.FileName
NavisionPath = Mid(CommonDialog.FileName, 1,
InStr(1, CommonDialog.FileName,
CommonDialog.FileTitle, vbTextCompare) - 2)
DatabaseFileName = CommonDialog.FileTitle

'Set the license file to be in the location where
'the database is located
LicenseFileName = NavisionPath + "\fin.flf"
CFRONT.SetNavisionPath NavisionPath
CFRONT.LoadLicenseFile LicenseFileName
CFRONT.CheckLicenseFile (9110)
Call
CFRONT.ConnectServerAndOpenDatabase(DriverName,
ServerName, NetType, DatabaseFileName, CacheSize,
UseCommitCache, NTAuthentication, UserId,
Password)
CompanyName = "CRONUS International Ltd."
CFRONT.OpenCompany CompanyName
```

The driver name is very important. Note the code above uses NDBCN for the Microsoft Dynamics NAV server and NDBCS for the SQL Server. If the open/connect operation fails, the function raises an exception that terminates the application. Please note that only one database/server connection can be open at a time.

Only one company can be open at a time. You must open a company before the application can access the data in the database tables. However, you can have any number of tables within a single company open at the same time. A table is identified by a unique number. When a table is opened, the database manager returns a unique handle, which remains valid until the table is closed. This handle must be passed to all operations carried out on the table. Note the code above also checks to ensure that the license file includes permissions for C/FRONT. If not, an error message appears.

11. Next, in the left drop-down list of the Code Window, select Exit and add the following code:

```
If TableHandle <> 0 Then
  CFRONT.CloseTable (TableHandle)
End If
If CompanyName <> "" Then
  CFRONT.CloseCompany
End If
If Connected Then
  CFRONT.DisconnectServer
End If
CFRONT.ReleaseAllObjects
MainForm.Hide
```

Note that you must do some error checking to avoid errors when trying to exit the application if you have not opened the company or viewed the records. If you do not have the code for the TableHandle, an Error 12000 – Invalid Handle message is displayed. If you do not have the code to check for the CompanyName, an Error 1046 No Company Selected message is displayed. After disconnecting from the server, make sure that you release all objects to ensure that memory used by C/FRONT is freed up.

12. Next, in the left drop-down list of the Code Window select **View** and add the following code:

```
Dim CurRow As Long
TableNo = 23 'Vendor
If Not MainForm.CFRONT.OpenTable(TableHandle,
TableNo)
Then
  MsgBox "Unable to Open Table 23 – Vendor."
  Return
End If
```

If the table number is not found, an error message that the table cannot be found is displayed and then an Error 1001 – Table does not exist message is displayed.

```
GridRow = 0
NumColumns = 1
FieldNo = MainForm.CFRONT.NextField(TableHandle,
0)
While FieldNo > 0
  NumColumns = NumColumns + 1
  TableBox.Cols = NumColumns
  TableBox.Row = GridRow
```

```
  TableBox.Col = NumColumns - 1
  TableBox.Text =
  MainForm.CFRONT.FieldName(TableHandle, FieldNo)
  FieldNo = MainForm.CFRONT.NextField(TableHandle,
  FieldNo)
Wend
'load the data
hRec = MainForm.CFRONT.AllocRec(TableHandle)
If MainForm.CFRONT.FindRec(TableHandle, hRec, "-")
Then
  GridRow = 1
  Do
    RecFields = "" & GridRow
    GridRow = GridRow + 1
    FieldNo =
MainForm.CFRONT.NextField(TableHandle,0)
  While FieldNo > 0
  RecFields = RecFields & Chr(9) &
  MainForm.CFRONT.FieldToStr(TableHandle, hRec,
    FieldNo)
  FieldNo =
MainForm.CFRONT.NextField(TableHandle, FieldNo)
  Wend
  TableBox.AddItem RecFields
  Loop Until MainForm.CFRONT.NextRec(TableHandle,
hRec, 1) = 0
End If
MainForm.CFRONT.FreeRec hRec
```

13. If you use a row value that does not exist, a 3009 − Invalid Row Value error message is displayed.

Writing Data Back to the Database Using C/FRONT in Microsoft Excel

Microsoft Dynamics NAV Attain comes with an excellent example that demonstrates both reading and writing to the Budget Table. The example shows you how to create your own error routine, unlike the OCX example used in Visual Basic, which is not allowed to have a custom error routine.

Limitations of C/FRONT

C/FRONT suffers from these limitations:

- Windows does not go into standby or hibernation if there is an open server connection from C/FRONT.
- Entries made by C/FRONT do not fire triggers to execute code validation.
- C/FRONT is very particular and looks in different places at different times for a license. Check and make sure you have the license loaded everywhere it needs to be, that is client, server, and so on.
- Besides the limitations, a common misconception that many have is that C/FRONT can be used in place of multiple sessions. However, this is not the case. Every connection in C/FRONT is treated as a session.
- As with regular Microsoft Dynamics NAV, the client and ODBC, the versions of CFRONT must match the version of the SERVER that you are trying to connect to.
- Keys can be active or inactive. You cannot activate keys from within C/FRONT; therefore, only active keys are available to C/FRONT.