OLE Embedded Components and Containers part 3

Program examples compiled using Visual C++ 6.0 compiler on Windows XP Pro machine with Service Pack 2. The Excel version is Excel 2003/Office 11. Topics and sub topics for this tutorial are listed below. Don't forget to read Tenouk's small disclaimer. The supplementary notes for this tutorial are IOleObject and OLE.

Index:

The EX32C Example: An OLE Embedded Component

You've already seen an MFC embedded component with in-place-activation capability (EX32A). Now you'll see a bare-bones component program that activates an embedded object in a separate window. It doesn't do much except display text and graphics in the window, but you'll learn a lot if you study the code. The application started as an SDI AppWizard Automation component with the document as the creatable object. The document's `IDispatch` interface was ripped out and replaced with `IOleObject`, `IDataObject`, and `IPersistStorage` interfaces. All the template server code carries through, so the document, view, and main frame objects are created when the container starts the component.
Open and build the EX32C project now. Run the application to register it, and then try it with the EX32B container or any other container program.

The EX32C from scratch

This is SDI application with **Automation** support and no **ActiveX Controls**.

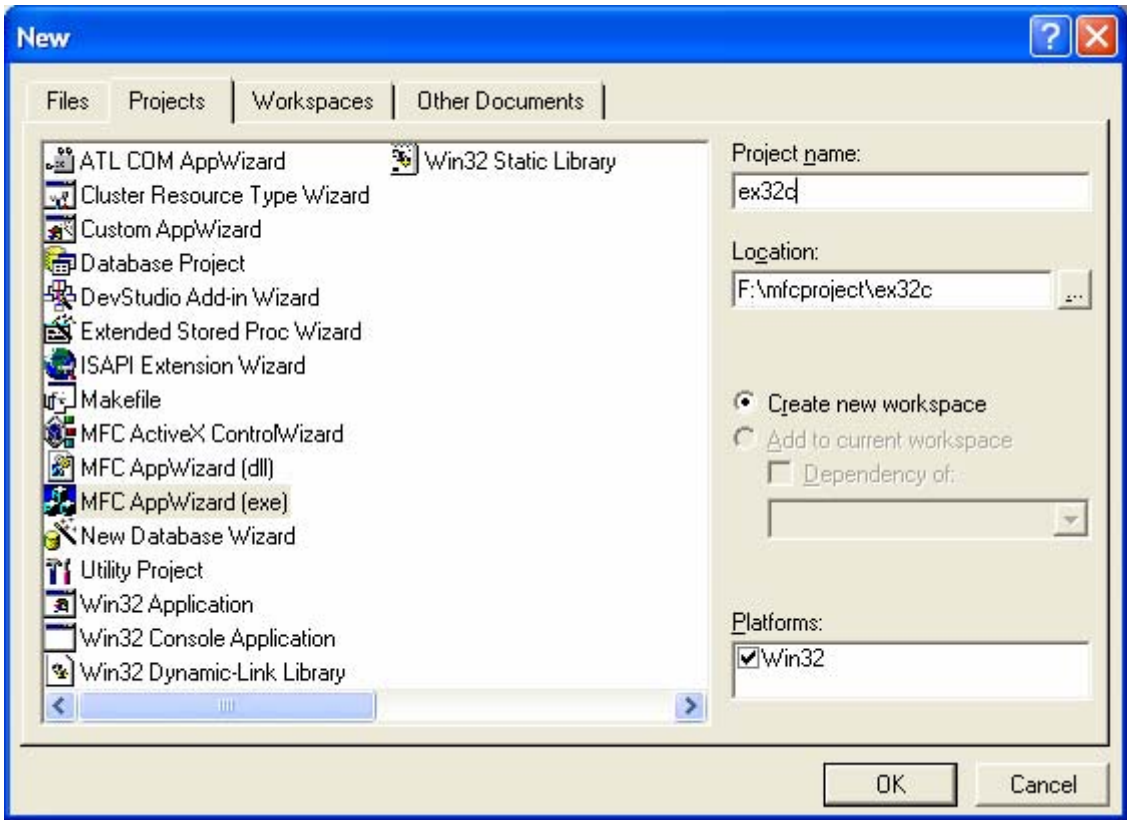

Figure 1: EX32C – Visual C++ new project dialog.

Select a **Single document** option.

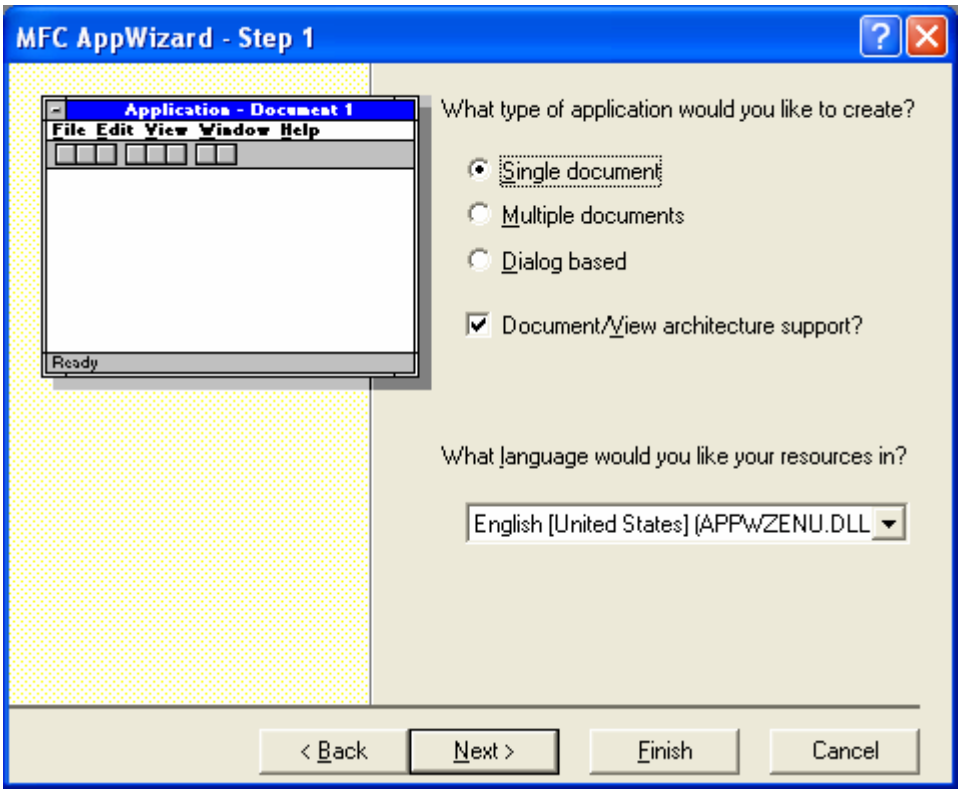

Figure 2: EX32C – AppWizard step 1 of 6.

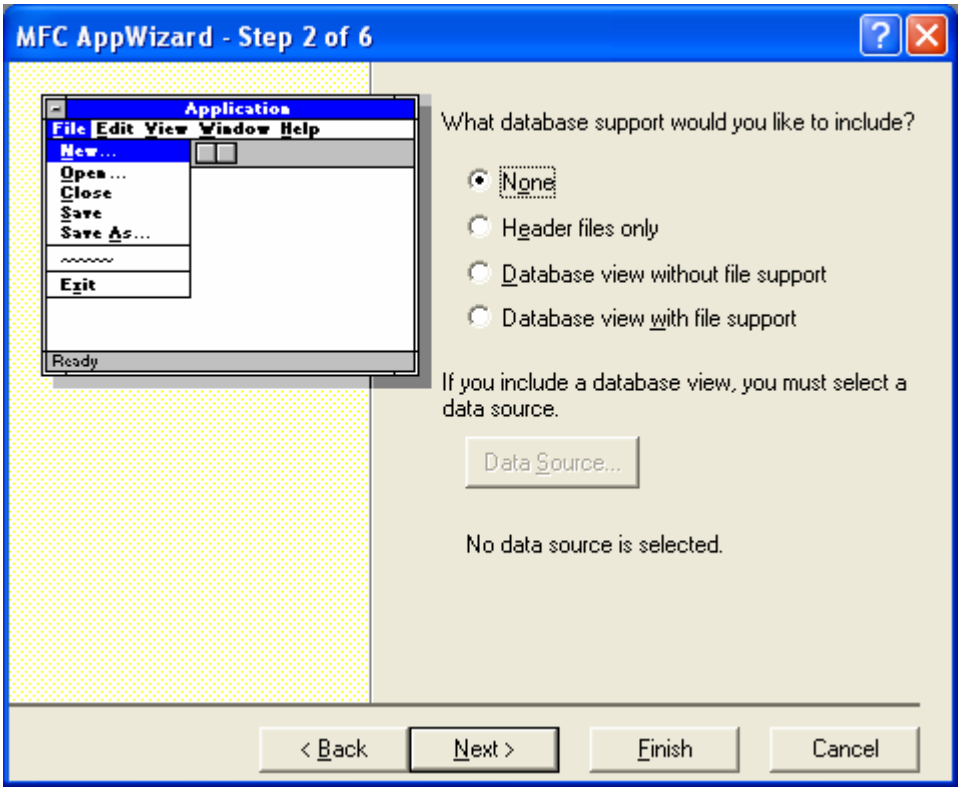

Figure 3: EX32C – AppWizard step 2 of 6.

Select **Automation** option and deselect **ActiveX Controls**.

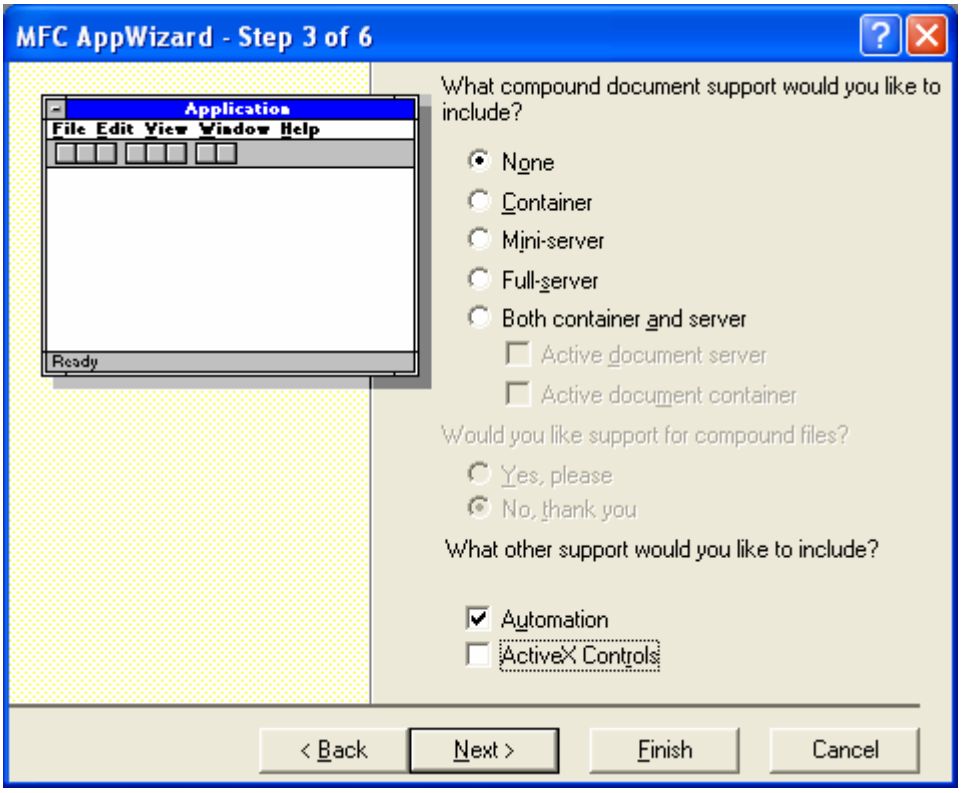

Figure 4: EX32C – AppWizard step 3 of 6.

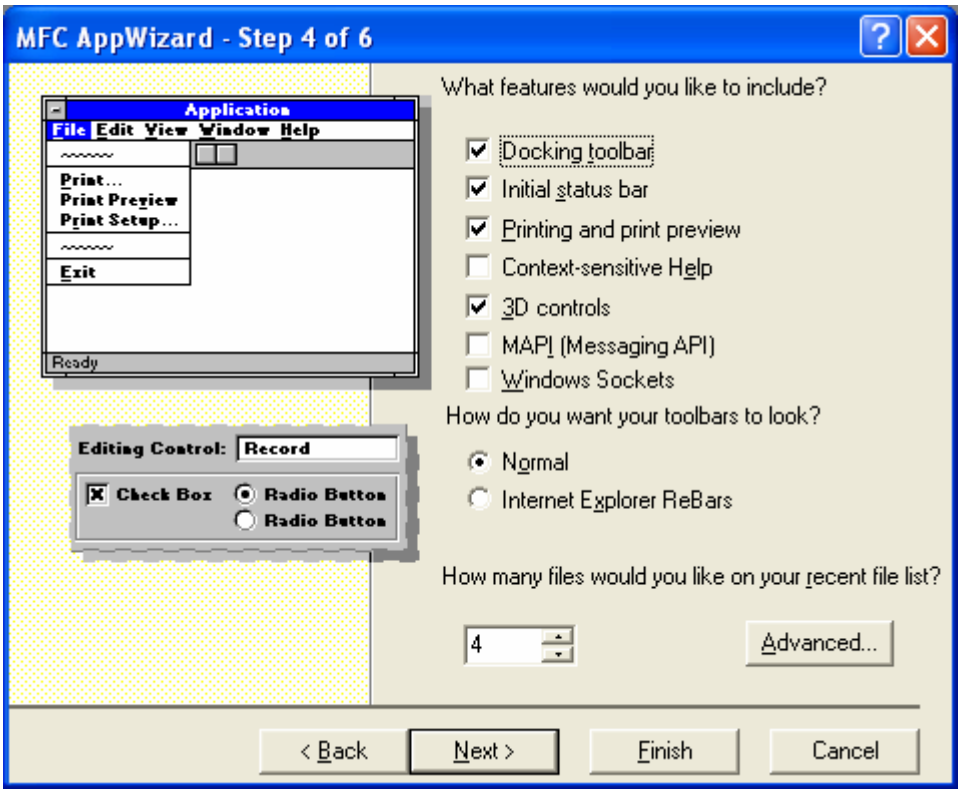

Figure 5: EX32C – AppWizard step 4 of 6.

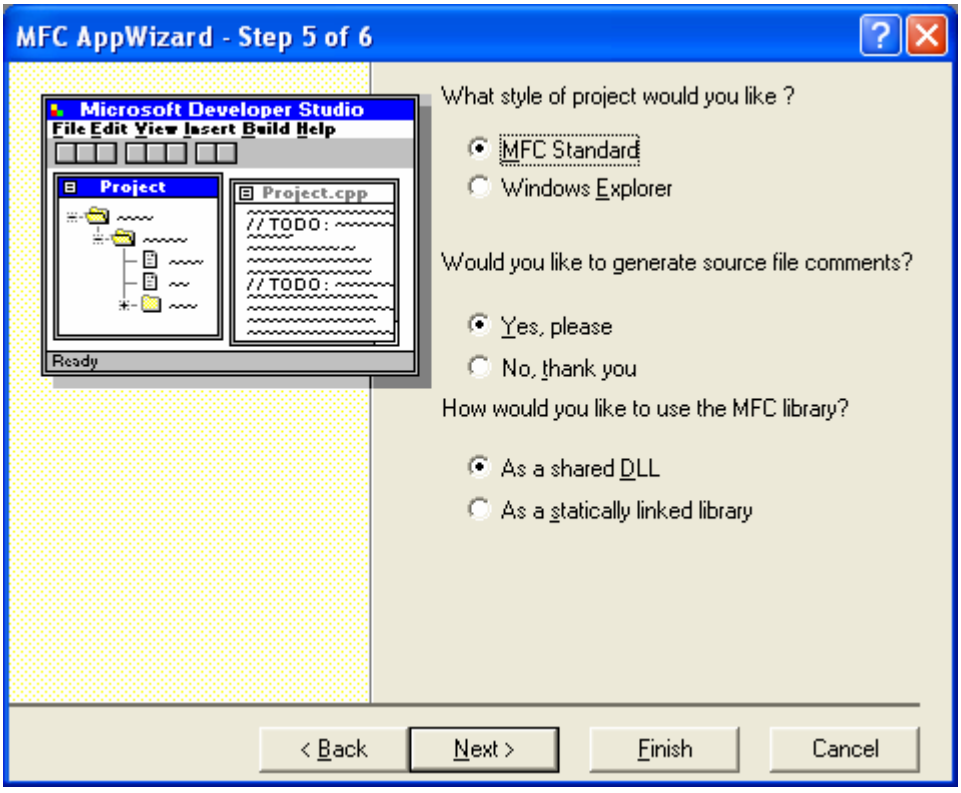

Figure 6: EX32C – AppWizard step 5 of 6.

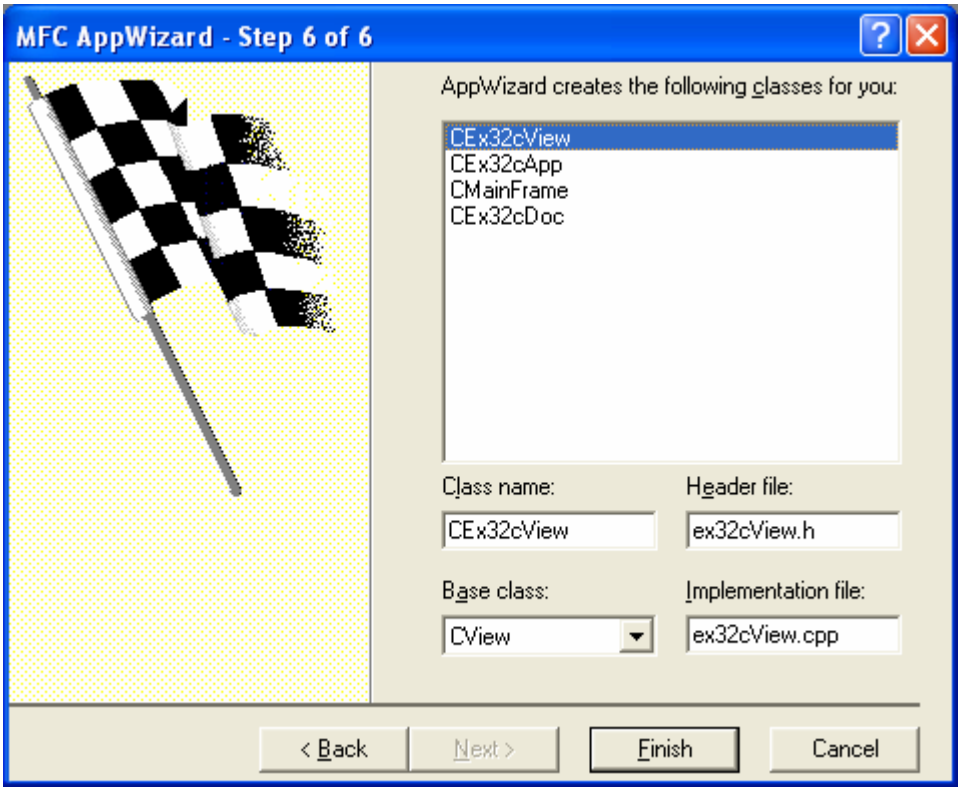

Figure 7: EX32C – AppWizard step 6 of 6.

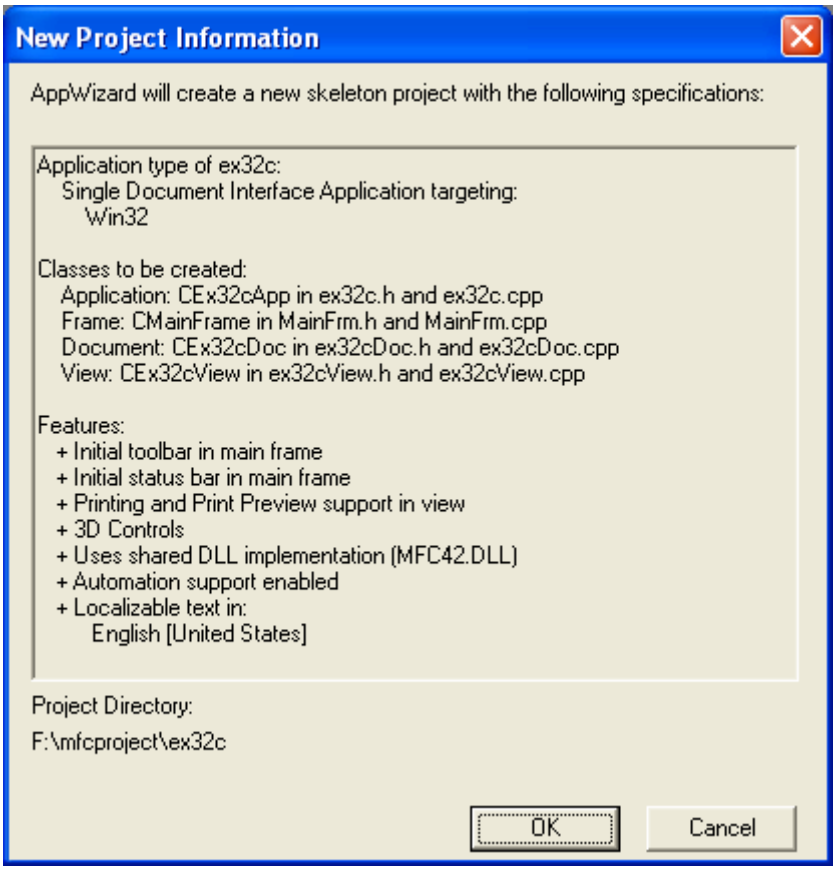

Figure 8: EX32C project summary.

Delete the file related menu items then replace them with **Update** menu item.

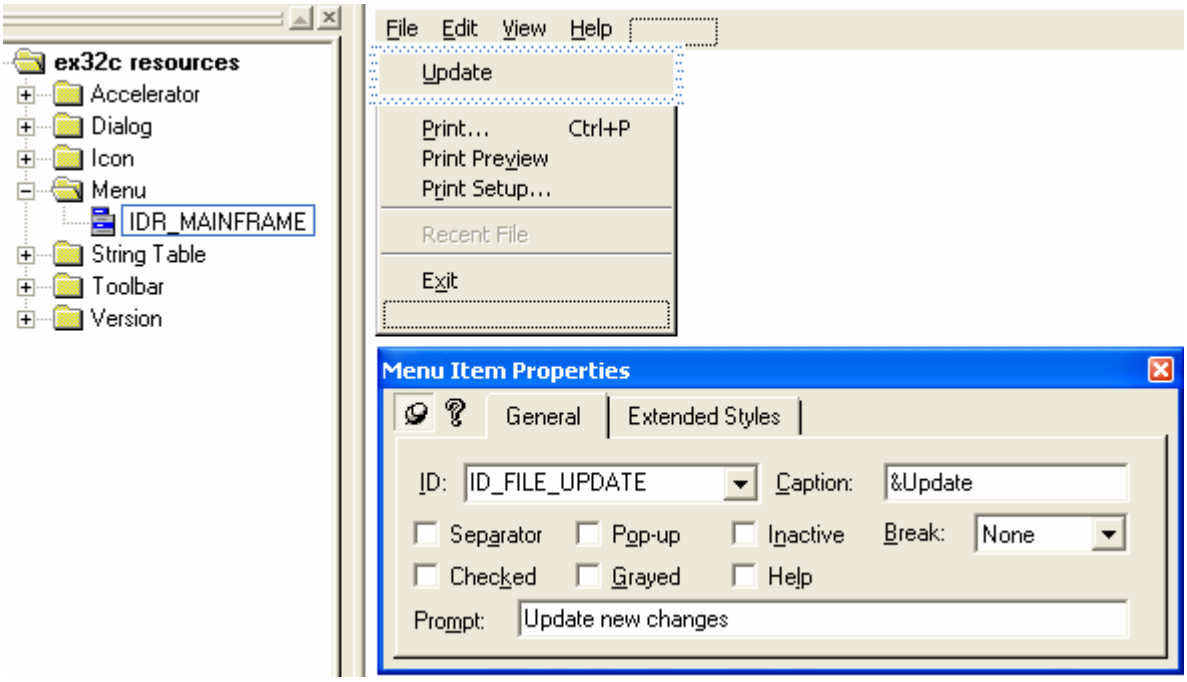

Figure 9: Replacing file related menus with **Update**.

Edit the **Exit** menu item.

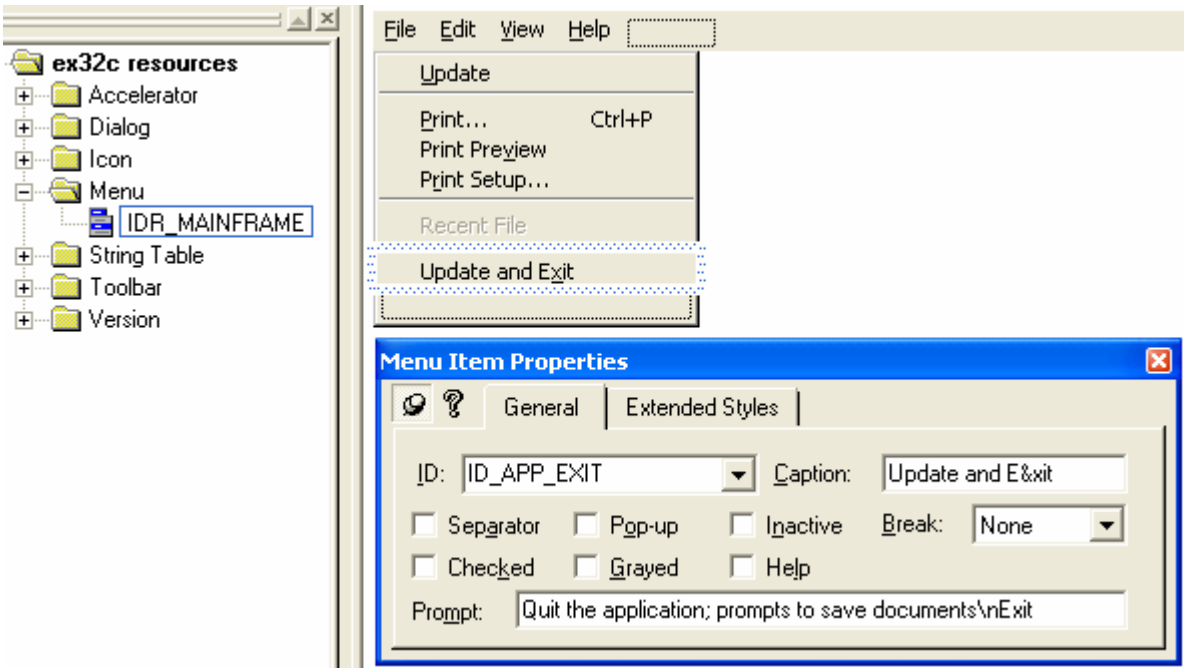

Figure 10: Editing the **Exit** menu item.

Add the following menu and its' items.

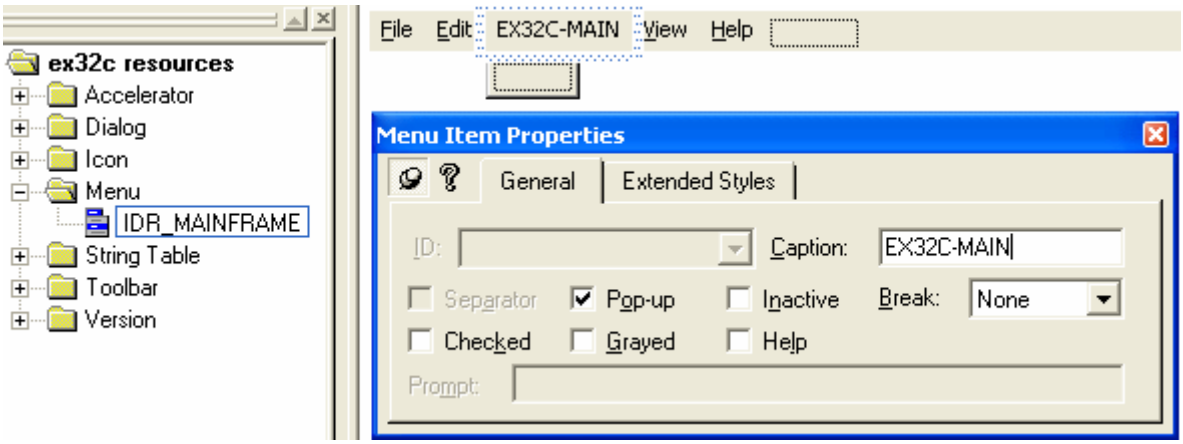

Figure 11: Adding menu and its' items.

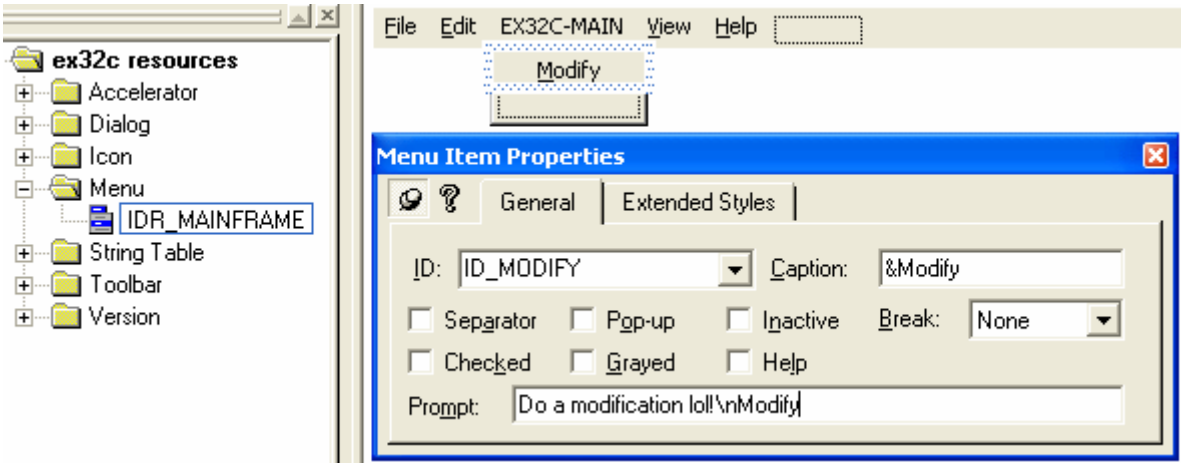

Figure 12: Adding **Modify** menu.

Add dialog and edit control.

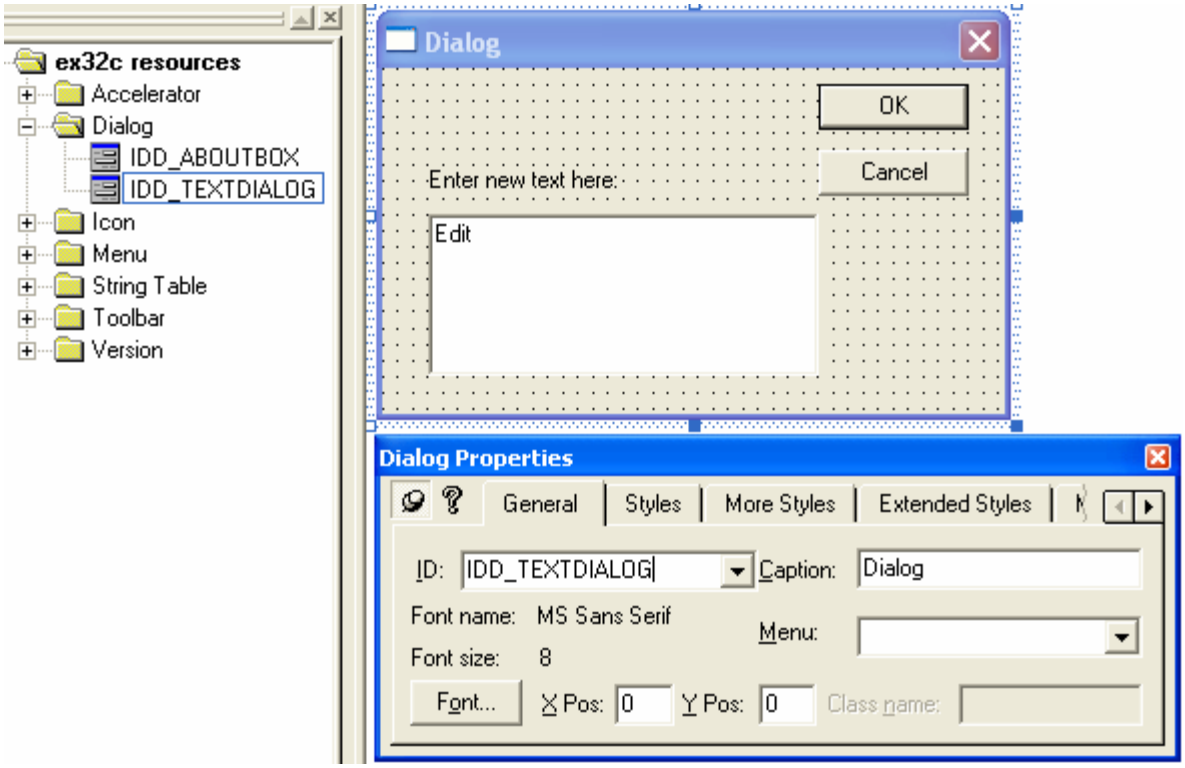

Figure 13: Adding dialog and Edit control.

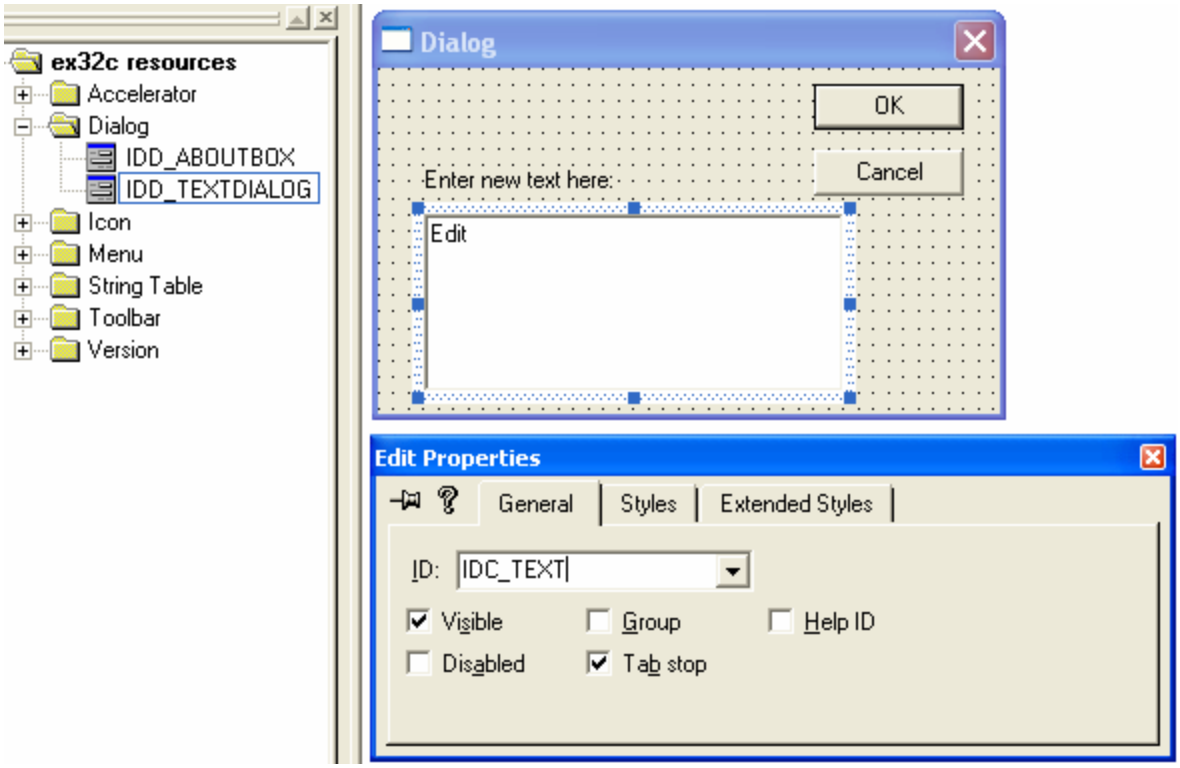

Figure 14: Edit control property.

Launch ClassWizard to add new class, `CTextDialog` for Dialog.

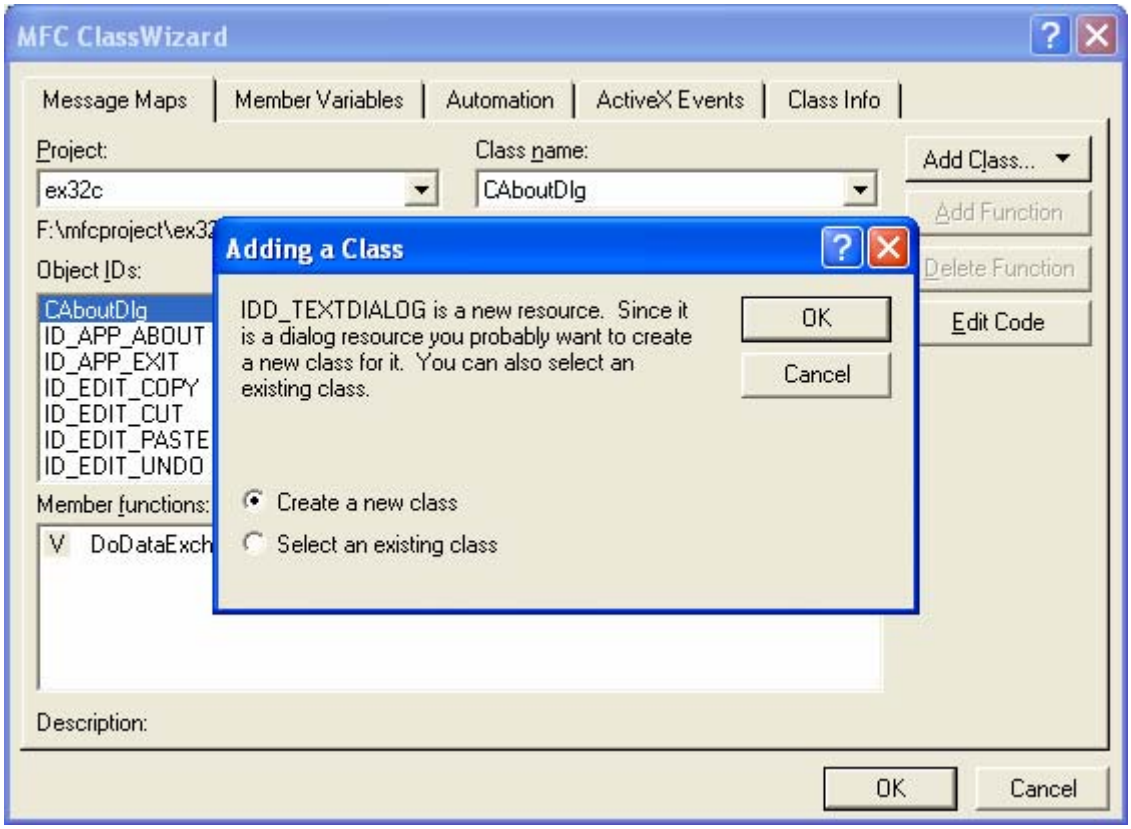

Figure 15: Adding new class for the previous dialog prompt.

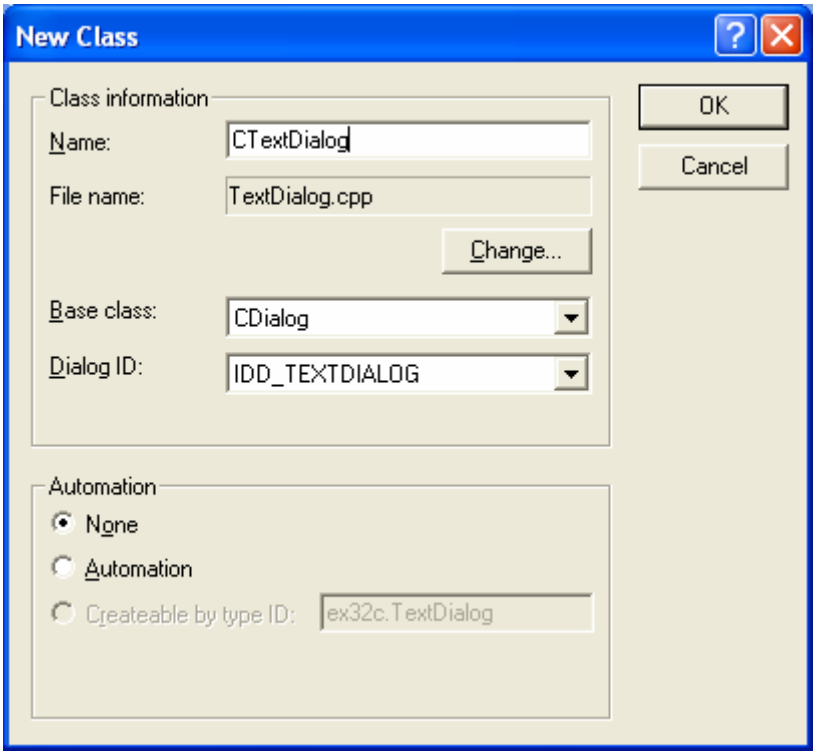

Figure 16: Adding `CTextDialog` class and its information.

Add member variable and set the maximum value to 100.

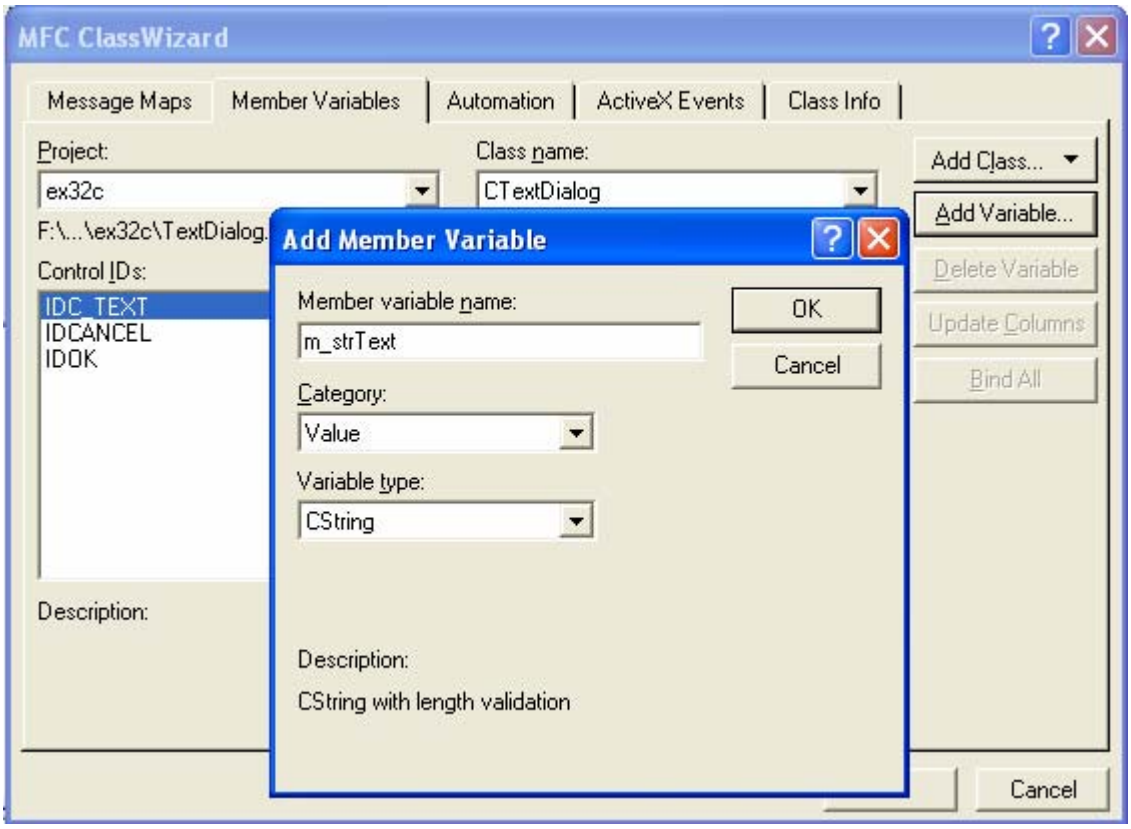

Figure 17: Adding `m_strText` member variable to `IDC_TEXT` Edit control of `CTextDialog` class.

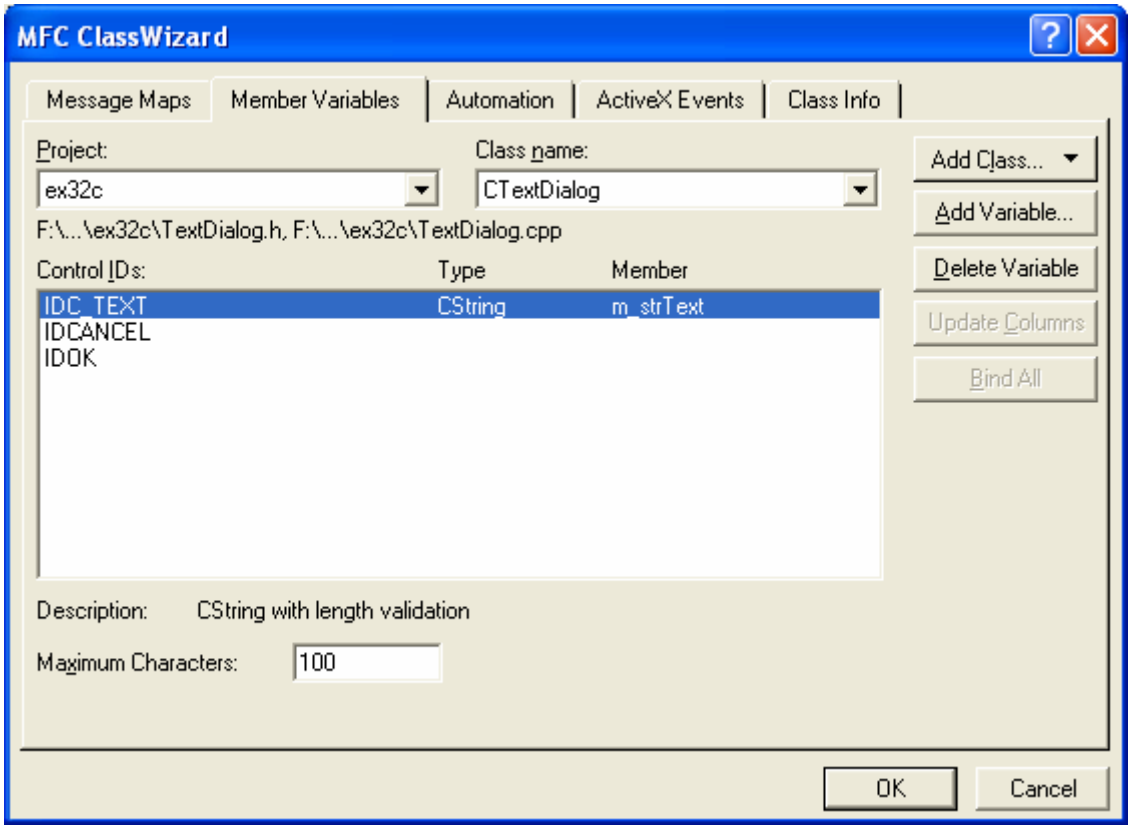

Figure 18: The added member variable.

Add/override `ExitInstance()` to `CEx32cApp` class.

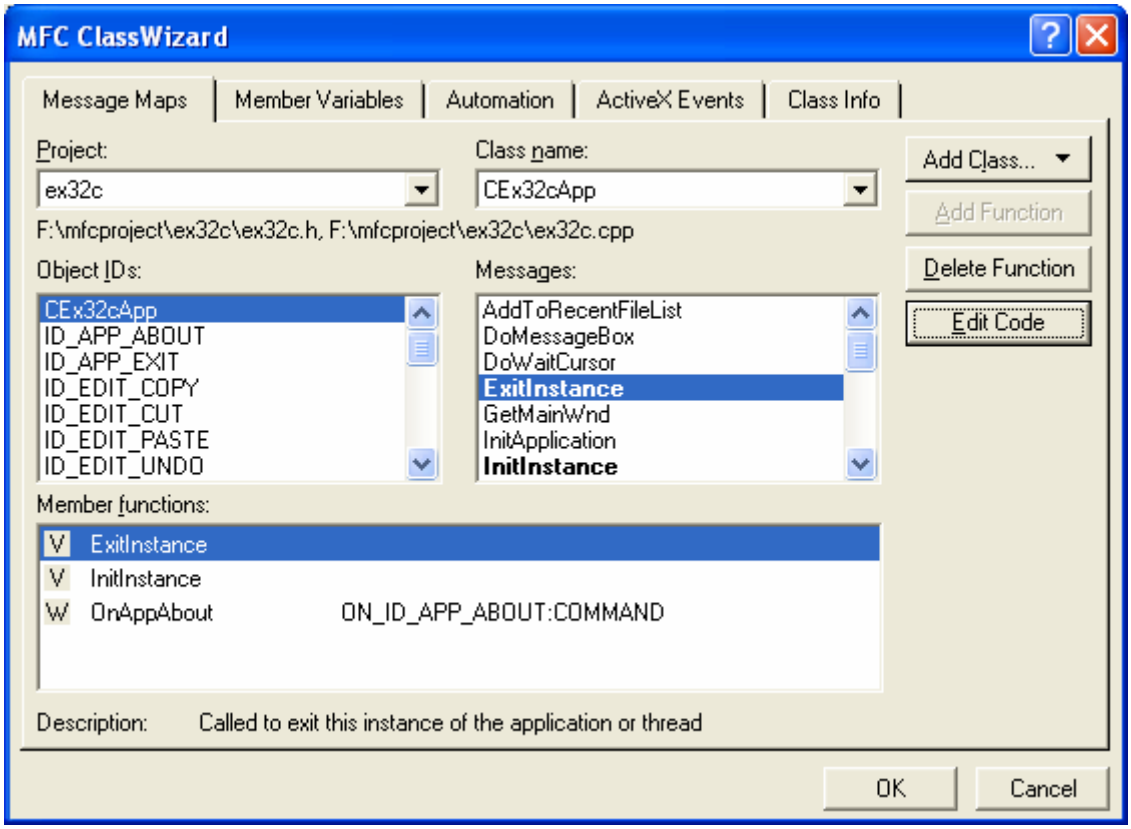

Figure 19: Adding `ExitInstance()` to `CEx32cApp` class.

Add/override `OnPrepareDC()` to `CEx32cView` class.

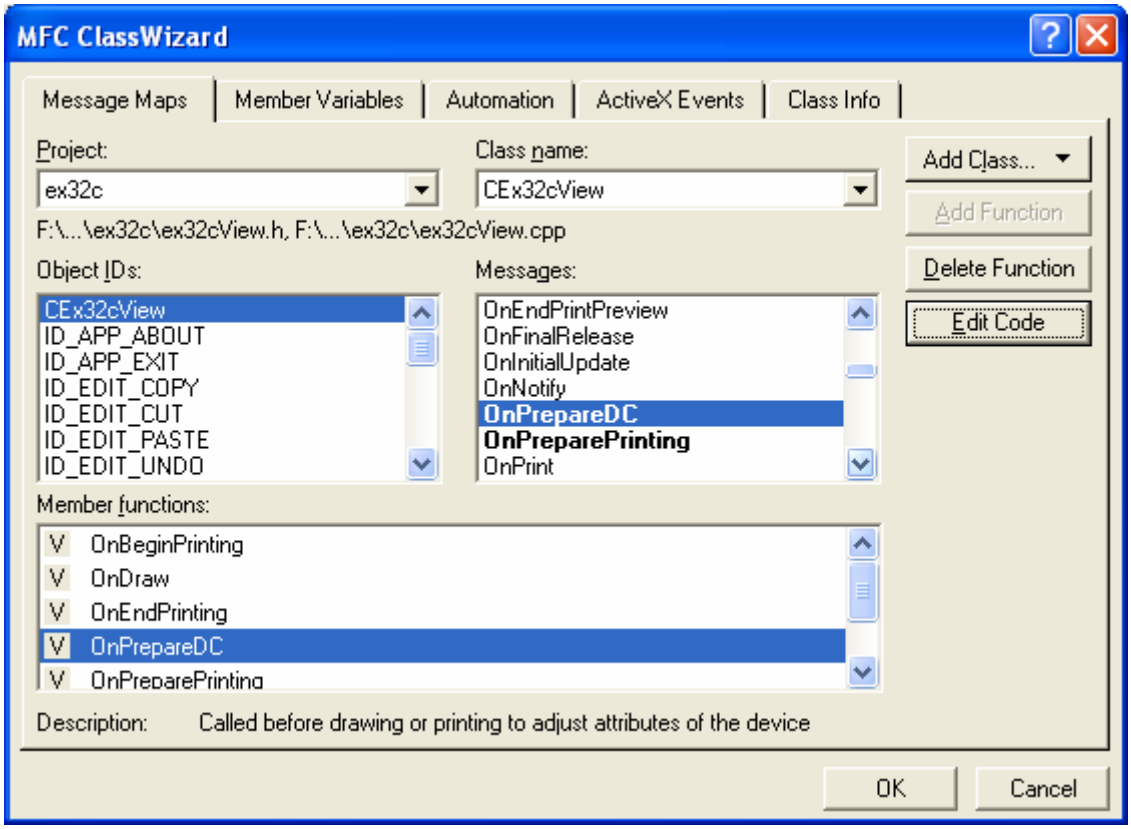

Figure 20: Adding `OnPrepareDC()` to `CEx32cView` class.

Add `OnCloseDocument()`, `OnFinalRelease()` and `SaveModified()` to `CEx32cDoc` class.

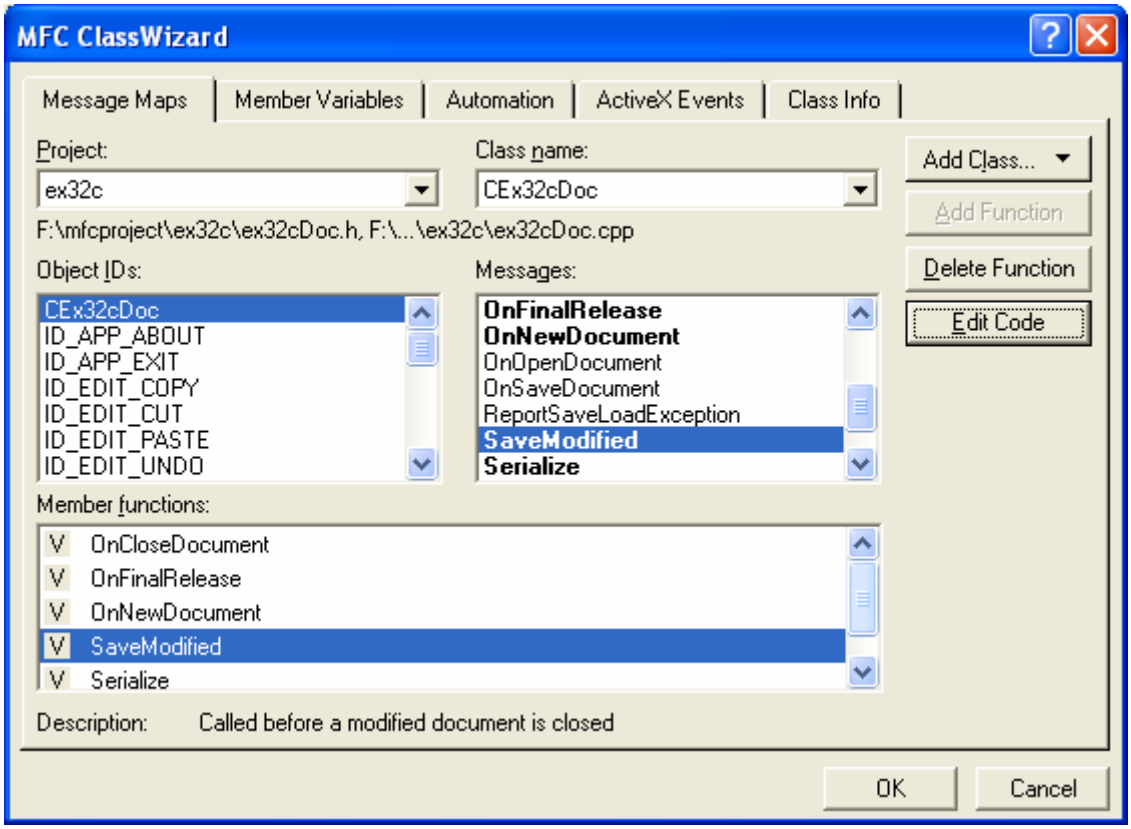

Figure 21: Add `OnCloseDocument()`, `OnFinalRelease()` and `SaveModified()` to `CEx32cDoc` class.

Add the following commands and update command.

ID	Type	Handler
`ID_MODIFY`	COMMAND	`OnModify()`
`ID_FILE_UPDATE`	COMMAND	`OnFileUpdate()`
`ID_FILE_UPDATE`	COMMAND UPDATE	`OnUpdateFileUpdate()`

Table 1.

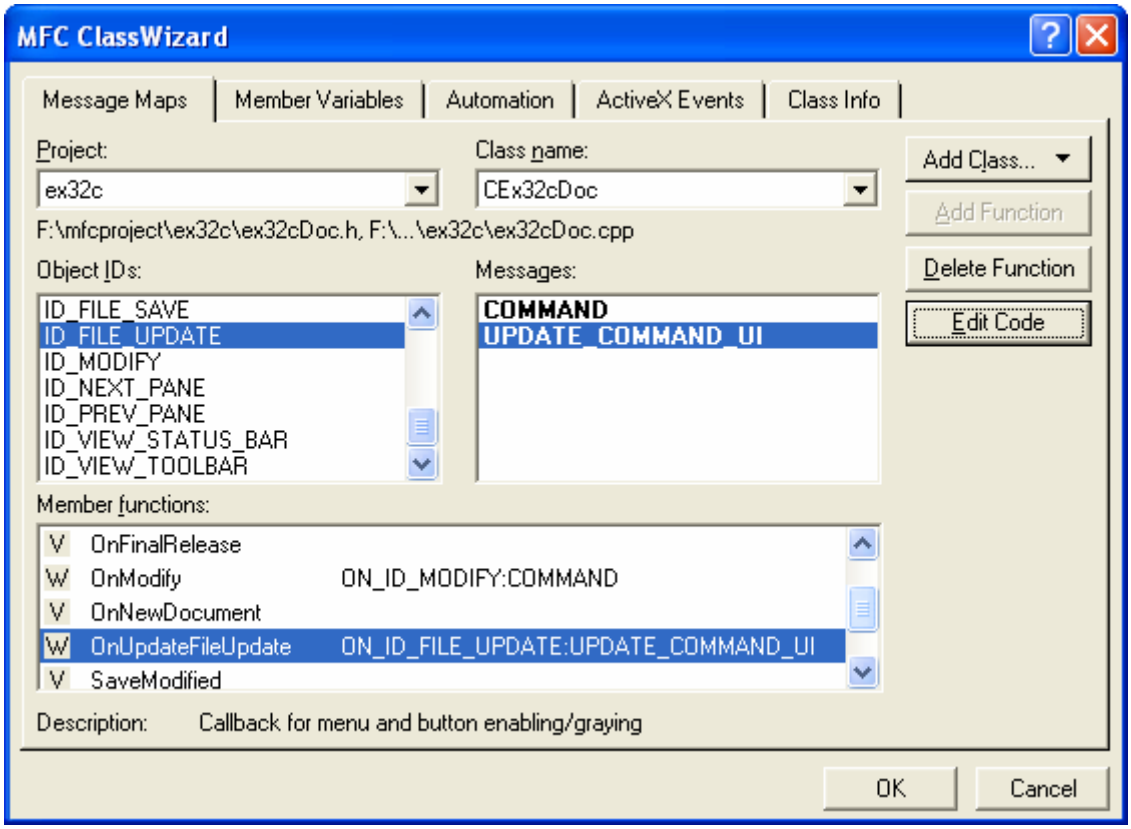

Figure 22: Adding commands and update command to `CEx32cDoc` class.

Add the following code to **ex32c.h**.

```
friend class CEx32cDoc;
```

```
class CEx32cApp : public CWinApp
{
friend class CEx32cDoc;

public:
    CEx32cApp();
```

Listing 1.

Add the following code to the `InitInstance()` of **ex32c.cpp**.

```
TRACE("CEx32cApp::InitInstance\n");
```

```
// CEx32cApp initialization

BOOL CEx32cApp::InitInstance()
{
    TRACE("CEx32cApp::InitInstance\n");
    // Initialize OLE libraries
    if (!AfxOleInit())
    {
```

Listing 2.

Edit the following code portion.

```
	// When a server application is launched stand-alone, it is a good idea
	//  to update the system registry in case it has been damaged.
	m_server.UpdateRegistry(OAT_SERVER);
	AfxMessageBox("Server can't be run stand-alone, use container -- Registry
updated");
	return FALSE;

//	// Dispatch commands specified on the command line
//	if (!ProcessShellCommand(cmdInfo))
//		return FALSE;

//	// The one and only window has been initialized, so show and update it.
//	m_pMainWnd->ShowWindow(SW_SHOW);
//	m_pMainWnd->UpdateWindow();

//	return TRUE;
```

```
    // When a server application is launched stand-alone, it is a good idea
    //  to update the system registry in case it has been damaged.
    m_server.UpdateRegistry(OAT_SERVER);
    AfxMessageBox("Server can't be run stand-alone, use container -- Registry updated");
    return FALSE;

//  // Dispatch commands specified on the command line
//  if (!ProcessShellCommand(cmdInfo))
//      return FALSE;

//  // The one and only window has been initialized, so show and update it.
//  m_pMainWnd->ShowWindow(SW_SHOW);
//  m_pMainWnd->UpdateWindow();

//  return TRUE;
}
```

Listing 3.

And the following code to the `ExitInstance()`.

```
int CEx32cApp::ExitInstance()
{
	// TODO: Add your specialized code here and/or call the base class
	TRACE("CEx32cApp::ExitInstance\n");
	return CWinApp::ExitInstance();
}
```

```
// CEx32cApp message handlers

int CEx32cApp::ExitInstance()
{
    // TODO: Add your specialized code here and/or call the base class
    TRACE("CEx32cApp::ExitInstance\n");
    return CWinApp::ExitInstance();
}
```

Listing 4.

Edit the `OnDraw()` code of **ex32cView.cpp**.

```
void CEx32cView::OnDraw(CDC* pDC)
{
	CEx32cDoc* pDoc = GetDocument();
	ASSERT_VALID(pDoc);
	pDC->Rectangle(CRect(500, -1000, 1500, -2000));
	pDC->TextOut(5, 5, pDoc->m_strText);
}
```

```
// CEx32cView drawing

void CEx32cView::OnDraw(CDC* pDC)
{
    CEx32cDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    pDC->Rectangle(CRect(500, -1000, 1500, -2000));
    pDC->TextOut(5, 5, pDoc->m_strText);
}
```

Listing 5.

Edit the `OnPrepareDC()` as shown below.

```
void CEx32cView::OnPrepareDC(CDC* pDC, CPrintInfo* pInfo)
{
	// TODO: Add your specialized code here and/or call the base class
	pDC->SetMapMode(MM_HIMETRIC);
}
```

```
// CEx32cView message handlers

void CEx32cView::OnPrepareDC(CDC* pDC, CPrintInfo* pInfo)
{
    // TODO: Add your specialized code here and/or call the base class
    pDC->SetMapMode(MM_HIMETRIC);
}
```

Listing 6.

Add the following codes to **ex32cDoc.h** just after the preprocessor directives

```
extern const CLSID clsid; // defined in ex32c.cpp
void ITrace(REFIID iid, const char* str);

#define SETFORMATETC(fe, cf, asp, td, med, li) \
	((fe).cfFormat=cf, \
	 (fe).dwAspect=asp, \
	 (fe).ptd=td, \
	 (fe).tymed=med, \
	 (fe).lindex=li)
```

```
#endif // _MSC_VER > 1000

extern const CLSID clsid; // defined in ex32c.cpp
void ITrace(REFIID iid, const char* str);

#define SETFORMATETC(fe, cf, asp, td, med, li) \
    ((fe).cfFormat=cf, \
     (fe).dwAspect=asp, \
     (fe).ptd=td, \
     (fe).tymed=med, \
     (fe).lindex=li)

class CEx32cDoc : public CDocument
```

Listing 7.

Add the following friend class, then adds those member variables and member function.

```
friend class CEx32cView;

private:
      CString m_strText;
      LPOLECLIENTSITE m_lpClientSite;
      LPOLEADVISEHOLDER m_lpOleAdviseHolder;
      LPDATAADVISEHOLDER m_lpDataAdviseHolder;
      CString m_strContainerApp;
      CString m_strContainerObj;

      HGLOBAL MakeMetaFile();
```

```
class CEx32cDoc : public CDocument
{
friend class CEx32cView;

private:
    CString m_strText;
    LPOLECLIENTSITE m_lpClientSite;
    LPOLEADVISEHOLDER m_lpOleAdviseHolder;
    LPDATAADVISEHOLDER m_lpDataAdviseHolder;
    CString m_strContainerApp;
    CString m_strContainerObj;

    HGLOBAL MakeMetaFile();

protected: // create from serialization only
```

Listing 8.

Manually add the following interface for `IOleObject`, `IDataObject` and `IPersistStorage` just after the previous codes.

```
      BEGIN_INTERFACE_PART(OleObject, IOleObject)
            STDMETHOD(SetClientSite)(LPOLECLIENTSITE);
            STDMETHOD(GetClientSite)(LPOLECLIENTSITE*);
            STDMETHOD(SetHostNames)(LPCOLESTR, LPCOLESTR);
            STDMETHOD(Close)(DWORD);
            STDMETHOD(SetMoniker)(DWORD, LPMONIKER);
            STDMETHOD(GetMoniker)(DWORD, DWORD, LPMONIKER*);
            STDMETHOD(InitFromData)(LPDATAOBJECT, BOOL, DWORD);
            STDMETHOD(GetClipboardData)(DWORD, LPDATAOBJECT*);
```

```
      STDMETHOD(DoVerb)(LONG, LPMSG, LPOLECLIENTSITE, LONG,
            HWND, LPCRECT);
      STDMETHOD(EnumVerbs)(LPENUMOLEVERB*);
      STDMETHOD(Update)();
      STDMETHOD(IsUpToDate)();
      STDMETHOD(GetUserClassID)(LPCLSID);
      STDMETHOD(GetUserType)(DWORD, LPOLESTR*);
      STDMETHOD(SetExtent)(DWORD, LPSIZEL);
      STDMETHOD(GetExtent)(DWORD, LPSIZEL);
      STDMETHOD(Advise)(LPADVISESINK, LPDWORD);
      STDMETHOD(Unadvise)(DWORD);
      STDMETHOD(EnumAdvise)(LPENUMSTATDATA*);
      STDMETHOD(GetMiscStatus)(DWORD, LPDWORD);
      STDMETHOD(SetColorScheme)(LPLOGPALETTE);
END_INTERFACE_PART(OleObject)

BEGIN_INTERFACE_PART(DataObject, IDataObject)
      STDMETHOD(GetData)(LPFORMATETC, LPSTGMEDIUM);
      STDMETHOD(GetDataHere)(LPFORMATETC, LPSTGMEDIUM);
      STDMETHOD(QueryGetData)(LPFORMATETC);
      STDMETHOD(GetCanonicalFormatEtc)(LPFORMATETC, LPFORMATETC);
      STDMETHOD(SetData)(LPFORMATETC, LPSTGMEDIUM, BOOL);
      STDMETHOD(EnumFormatEtc)(DWORD, LPENUMFORMATETC*);
      STDMETHOD(DAdvise)(LPFORMATETC, DWORD, LPADVISESINK, LPDWORD);
      STDMETHOD(DUnadvise)(DWORD);
      STDMETHOD(EnumDAdvise)(LPENUMSTATDATA*);
END_INTERFACE_PART(DataObject)

BEGIN_INTERFACE_PART(PersistStorage, IPersistStorage)
      STDMETHOD(GetClassID)(LPCLSID);
      STDMETHOD(IsDirty)();
      STDMETHOD(InitNew)(LPSTORAGE);
      STDMETHOD(Load)(LPSTORAGE);
      STDMETHOD(Save)(LPSTORAGE, BOOL);
      STDMETHOD(SaveCompleted)(LPSTORAGE);
      STDMETHOD(HandsOffStorage)();
END_INTERFACE_PART(PersistStorage)

DECLARE_INTERFACE_MAP()
```

```
class CEx32cDoc : public CDocument
{
friend class CEx32cView;

private:
    CString m_strText;
    LPOLECLIENTSITE m_lpClientSite;
    LPOLEADVISEHOLDER m_lpOleAdviseHolder;
    LPDATAADVISEHOLDER m_lpDataAdviseHolder;
    CString m_strContainerApp;
    CString m_strContainerObj;

    HGLOBAL MakeMetaFile();

    BEGIN_INTERFACE_PART(OleObject, IOleObject)
        STDMETHOD(SetClientSite)(LPOLECLIENTSITE);
        STDMETHOD(GetClientSite)(LPOLECLIENTSITE*);
        STDMETHOD(SetHostNames)(LPCOLESTR, LPCOLESTR);
        STDMETHOD(Close)(DWORD);
        STDMETHOD(SetMoniker)(DWORD, LPMONIKER);
        STDMETHOD(GetMoniker)(DWORD, DWORD, LPMONIKER*);
        STDMETHOD(InitFromData)(LPDATAOBJECT, BOOL, DWORD);
        STDMETHOD(GetClipboardData)(DWORD, LPDATAOBJECT*);
        STDMETHOD(DoVerb)(LONG, LPMSG, LPOLECLIENTSITE, LONG,
            HWND, LPCRECT);
        STDMETHOD(EnumVerbs)(LPENUMOLEVERB*);
        STDMETHOD(Update)();
        STDMETHOD(IsUpToDate)();
        STDMETHOD(GetUserClassID)(LPCLSID);
        STDMETHOD(GetUserType)(DWORD, LPOLESTR*);
        STDMETHOD(SetExtent)(DWORD, LPSIZEL);
```

Listing 9.

Delete (or comment out) the AppWizard generated `DECLARE_DISPATCH_MAP` and `DECLARE_INTERFACE_MAP`.

```
//    // Generated OLE dispatch map functions
//    //{{AFX_DISPATCH(CEx32cDoc)
//          // NOTE - the ClassWizard will add and remove member functions here.
//          //    DO NOT EDIT what you see in these blocks of generated code !
//    //}}AFX_DISPATCH
//    DECLARE_DISPATCH_MAP()
//    DECLARE_INTERFACE_MAP()

    DECLARE_MESSAGE_MAP()

//  // Generated OLE dispatch map functions
//  //{{AFX_DISPATCH(CEx32cDoc)
//      // NOTE - the ClassWizard will add and remove member functions here.
//      //    DO NOT EDIT what you see in these blocks of generated code !
//  //}}AFX_DISPATCH
//  DECLARE_DISPATCH_MAP()
//  DECLARE_INTERFACE_MAP()
};
```

Listing 10.

Add the following `#include` directive to **ex32cDoc.cpp**.

```
#include "TextDialog.h"
```

```
#include "stdafx.h"
#include "ex32c.h"

#include "ex32cDoc.h"
#include "TextDialog.h"

#ifdef _DEBUG
```

Listing 11.

Manually add the following interface.

```
BEGIN_INTERFACE_MAP(CEx32cDoc, CDocument)
      INTERFACE_PART(CEx32cDoc, IID_IOleObject, OleObject)
      INTERFACE_PART(CEx32cDoc, IID_IDataObject, DataObject)
      INTERFACE_PART(CEx32cDoc, IID_IPersistStorage, PersistStorage)
END_INTERFACE_MAP()
```

```
// CEx32cDoc

IMPLEMENT_DYNCREATE(CEx32cDoc, CDocument)

BEGIN_INTERFACE_MAP(CEx32cDoc, CDocument)
    INTERFACE_PART(CEx32cDoc, IID_IOleObject, OleObject)
    INTERFACE_PART(CEx32cDoc, IID_IDataObject, DataObject)
    INTERFACE_PART(CEx32cDoc, IID_IPersistStorage, PersistStorage)
END_INTERFACE_MAP()

BEGIN_MESSAGE_MAP(CEx32cDoc, CDocument)
```

Listing 12.

Delete (or comment out) the auto AppWizard generated `DISPATCH_MAP` and `INTERFACE_MAP`.

```
// BEGIN_DISPATCH_MAP(CEx32cDoc, CDocument)
//    //{{AFX_DISPATCH_MAP(CEx32cDoc)
//          // NOTE - the ClassWizard will add and remove mapping macros here.
//          //      DO NOT EDIT what you see in these blocks of generated code!
//    //}}AFX_DISPATCH_MAP
// END_DISPATCH_MAP()

// // Note: we add support for IID_IEx32c to support typesafe binding
// //  from VBA.  This IID must match the GUID that is attached to the
// //  dispinterface in the .ODL file.

// // {100A83A5-C1FB-4EA7-8AFC-689433F842E8}
// static const IID IID_IEx32c =
// { 0x100a83a5, 0xc1fb, 0x4ea7, { 0x8a, 0xfc, 0x68, 0x94, 0x33, 0xf8, 0x42,
0xe8 } };

// BEGIN_INTERFACE_MAP(CEx32cDoc, CDocument)
//    INTERFACE_PART(CEx32cDoc, IID_IEx32c, Dispatch)
// END_INTERFACE_MAP()
```

```
END_MESSAGE_MAP()

// BEGIN_DISPATCH_MAP(CEx32cDoc, CDocument)
//  //{{AFX_DISPATCH_MAP(CEx32cDoc)
//      // NOTE - the ClassWizard will add and remove mapping macros here.
//      //      DO NOT EDIT what you see in these blocks of generated code!
//  //}}AFX_DISPATCH_MAP
// END_DISPATCH_MAP()

// // Note: we add support for IID_IEx32c to support typesafe binding
// //  from VBA.  This IID must match the GUID that is attached to the
// //  dispinterface in the .ODL file.

// // {100A83A5-C1FB-4EA7-8AFC-689433F842E8}
// static const IID IID_IEx32c =
// { 0x100a83a5, 0xc1fb, 0x4ea7, { 0x8a, 0xfc, 0x68, 0x94, 0x33, 0xf8,
// 0x42, 0xe8 } };

// BEGIN_INTERFACE_MAP(CEx32cDoc, CDocument)
//  INTERFACE_PART(CEx32cDoc, IID_IEx32c, Dispatch)
// END_INTERFACE_MAP()
```

Listing 13.

Add the implementation for the interfaces. Firstly add the implementation for `OleObject` just after the previous commented codes.

```
/////////////////////////////////////////////////////////////////////////////
// CEx32cDoc OLE interface functions
STDMETHODIMP_(ULONG) CEx32cDoc::XOleObject::AddRef()
{
	TRACE("CEx32cDoc::XOleObject::AddRef\n");
	METHOD_PROLOGUE(CEx32cDoc, OleObject)
	return pThis->InternalAddRef();
}

STDMETHODIMP_(ULONG) CEx32cDoc::XOleObject::Release()
{
	TRACE("CEx32cDoc::XOleObject::Release\n");
	METHOD_PROLOGUE(CEx32cDoc, OleObject)
	return pThis->InternalRelease();
}

STDMETHODIMP CEx32cDoc::XOleObject::QueryInterface(
	REFIID iid, LPVOID* ppvObj)
{
	ITrace(iid, "CEx32cDoc::XOleObject::QueryInterface");
	METHOD_PROLOGUE(CEx32cDoc, OleObject)
	return pThis->InternalQueryInterface(&iid, ppvObj);
}

STDMETHODIMP CEx32cDoc::XOleObject::SetClientSite(
	LPOLECLIENTSITE pClientSite)
{
	TRACE("CEx32cDoc::XOleObject::SetClientSite\n");
	METHOD_PROLOGUE(CEx32cDoc, OleObject)
	// linked objects do not support SetClientSite
	if (pClientSite != NULL)
		pClientSite->AddRef();
```

```
	if(pThis->m_lpClientSite != NULL) pThis->m_lpClientSite->Release();
	pThis->m_lpClientSite = pClientSite;

	return NOERROR;
}

STDMETHODIMP CEx32cDoc::XOleObject::GetClientSite(
	LPOLECLIENTSITE* ppClientSite)
{
	TRACE("CEx32cDoc::XOleObject::GetClientSite\n");
	METHOD_PROLOGUE(CEx32cDoc, OleObject)

	*ppClientSite = pThis->m_lpClientSite;
	  if (pThis->m_lpClientSite != NULL)
			pThis->m_lpClientSite->AddRef();  // IMPORTANT
	return NOERROR;
}

STDMETHODIMP CEx32cDoc::XOleObject::SetHostNames(
	LPCOLESTR szContainerApp, LPCOLESTR szContainerObj)
{
	TRACE("CEx32cDoc::XOleObject::SetHostNames\n");
	METHOD_PROLOGUE(CEx32cDoc, OleObject)

	CString strTitle = "EX28C object embedded in ";
	if(szContainerApp != NULL) {
		strTitle += CString(szContainerApp);
	}
	CWnd* pWnd = AfxGetMainWnd();
	pWnd->SetWindowText(strTitle);

	return NOERROR;
}

STDMETHODIMP CEx32cDoc::XOleObject::Close(DWORD /*dwSaveOption*/)
{
	TRACE("CEx32cDoc::XOleObject::Close\n");
	// linked objects do not support close

	return E_NOTIMPL;
}

STDMETHODIMP CEx32cDoc::XOleObject::SetMoniker(
	DWORD /*dwWhichMoniker*/, LPMONIKER /*pmk*/)
{
	TRACE("CEx32cDoc::XOleObject::SetMoniker\n");
	// linked objects do not support SetMoniker

	return E_NOTIMPL;
}

STDMETHODIMP CEx32cDoc::XOleObject::GetMoniker(
	DWORD dwAssign, DWORD dwWhichMoniker, LPMONIKER* ppMoniker)
{
	TRACE("CEx32cDoc::XOleObject::GetMoniker\n");
	return E_NOTIMPL;
}
```

```
STDMETHODIMP CEx32cDoc::XOleObject::InitFromData(
      LPDATAOBJECT /*pDataObject*/, BOOL /*fCreation*/, DWORD /*dwReserved*/)
{
      TRACE("CEx32cDoc::XOleObject::InitFromData\n");
      // linked objects do not support InitFromData

      return E_NOTIMPL;
}

STDMETHODIMP CEx32cDoc::XOleObject::GetClipboardData(
      DWORD /*dwReserved*/, LPDATAOBJECT* ppDataObject)
{
      TRACE("CEx32cDoc::XOleObject::GetClipboardData\n");
      return E_NOTIMPL;
}

STDMETHODIMP CEx32cDoc::XOleObject::DoVerb(
      LONG iVerb, LPMSG lpmsg, LPOLECLIENTSITE pActiveSite, LONG lindex,
      HWND hwndParent, LPCRECT lpPosRect)
{
      TRACE("CEx32cDoc::XOleObject::DoVerb - %d\n", iVerb);
      METHOD_PROLOGUE(CEx32cDoc, OleObject)
      ASSERT_VALID(pThis);

      pThis->InternalAddRef();    // protect this object
      CWnd* pWnd = AfxGetMainWnd();
      switch (iVerb)
      {
      // open - maps to OnOpen
      case OLEIVERB_OPEN:
      case -OLEIVERB_OPEN-1:  // allows positive OLEIVERB_OPEN-1 in registry
      case OLEIVERB_PRIMARY:  // OLEIVERB_PRIMARY is 0 and "Edit" in registry
      case OLEIVERB_SHOW:
            pWnd->ShowWindow(SW_SHOW);
            pWnd->SetActiveWindow();
            pWnd->SetForegroundWindow();
            break;

      // hide maps to OnHide
      case OLEIVERB_HIDE:
      case -OLEIVERB_HIDE-1:  // allows positive OLEIVERB_HIDE-1 in registry
            return E_NOTIMPL;

      default:
            // negative verbs not understood should return E_NOTIMPL
            if (iVerb < 0)
                  return E_NOTIMPL;

            AfxThrowOleException(OLEOBJ_S_INVALIDVERB);
      }
      pThis->InternalRelease();   // may 'delete this'
      pThis->m_lpClientSite->OnShowWindow(TRUE); // hatch

      return NOERROR;
}
```

```
STDMETHODIMP CEx32cDoc::XOleObject::EnumVerbs(
	IEnumOLEVERB** ppenumOleVerb)
{
	TRACE("CEx32cDoc::XOleObject::EnumVerbs\n");
	return E_NOTIMPL;
}

STDMETHODIMP CEx32cDoc::XOleObject::Update()
{
	TRACE("CEx32cDoc::XOleObject::Update\n");
	METHOD_PROLOGUE(CEx32cDoc, OleObject)
	return E_NOTIMPL;
}

STDMETHODIMP CEx32cDoc::XOleObject::IsUpToDate()
{
	TRACE("CEx32cDoc::XOleObject::IsUpToDate\n");
	return E_NOTIMPL;
}

STDMETHODIMP CEx32cDoc::XOleObject::GetUserClassID(CLSID* pClsid)
{
	TRACE("CEx32cDoc::XOleObject::GetUserClassID\n");
	METHOD_PROLOGUE(CEx32cDoc, OleObject)
	ASSERT_VALID(pThis);

	return pThis->m_xPersistStorage.GetClassID(pClsid);
}

STDMETHODIMP CEx32cDoc::XOleObject::GetUserType(
	DWORD dwFormOfType, LPOLESTR* ppszUserType)
{
	TRACE("CEx32cDoc::XOleObject::GetUserType\n");
	METHOD_PROLOGUE(CEx32cDoc, OleObject)
	ASSERT_VALID(pThis);

	*ppszUserType = NULL;

	CLSID clsid;
	pThis->m_xOleObject.GetUserClassID(&clsid);
	return OleRegGetUserType(clsid, dwFormOfType, ppszUserType);
}

STDMETHODIMP CEx32cDoc::XOleObject::SetExtent(
	DWORD /*dwDrawAspect*/, LPSIZEL /*lpsizel*/)
{
	TRACE("CEx32cDoc::XOleObject::SetExtent\n");
	return E_FAIL;
}

STDMETHODIMP CEx32cDoc::XOleObject::GetExtent(
	DWORD dwDrawAspect, LPSIZEL lpsizel)
{
	TRACE("CEx32cDoc::XOleObject::GetExtent\n");
	METHOD_PROLOGUE(CEx32cDoc, OleObject)
	ASSERT_VALID(pThis);
	// handler returns extent in metafilepict
```

```
	return E_NOTIMPL;
}

STDMETHODIMP CEx32cDoc::XOleObject::Advise(
	IAdviseSink* pAdvSink, DWORD* pdwConnection)
{
	TRACE("CEx32cDoc::XOleObject::Advise\n");
	METHOD_PROLOGUE(CEx32cDoc, OleObject)
	ASSERT_VALID(pThis);
	*pdwConnection = 0;
	if (pThis->m_lpOleAdviseHolder == NULL &&
		::CreateOleAdviseHolder(&pThis->m_lpOleAdviseHolder)
		!= NOERROR) {
		return E_OUTOFMEMORY;
	}
	ASSERT(pThis->m_lpOleAdviseHolder != NULL);
	return pThis->m_lpOleAdviseHolder->Advise(pAdvSink, pdwConnection);
}

STDMETHODIMP CEx32cDoc::XOleObject::Unadvise(DWORD dwConnection)
{
	TRACE("CEx32cDoc::XOleObject::Unadvise\n");
	METHOD_PROLOGUE(CEx32cDoc, OleObject)
	ASSERT_VALID(pThis);
	return E_NOTIMPL;
}

STDMETHODIMP CEx32cDoc::XOleObject::EnumAdvise(
	LPENUMSTATDATA* ppenumAdvise)
{
	TRACE("CEx32cDoc::XOleObject::EnumAdvise\n");
	METHOD_PROLOGUE(CEx32cDoc, OleObject)
	ASSERT_VALID(pThis);
	return E_NOTIMPL;
}

STDMETHODIMP CEx32cDoc::XOleObject::GetMiscStatus(
	DWORD dwAspect, DWORD* pdwStatus)
{
	TRACE("CEx32cDoc::XOleObject::GetMiscStatus\n");
	METHOD_PROLOGUE(CEx32cDoc, OleObject)
	ASSERT_VALID(pThis);

	*pdwStatus = 0;

	CLSID clsid;
	pThis->m_xOleObject.GetUserClassID(&clsid);
	return OleRegGetMiscStatus(clsid, dwAspect, pdwStatus);
}

STDMETHODIMP CEx32cDoc::XOleObject::SetColorScheme(LPLOGPALETTE lpLogpal)
{
	TRACE("CEx32cDoc::XOleObject::SetColorScheme\n");
	METHOD_PROLOGUE(CEx32cDoc, OleObject)
	ASSERT_VALID(pThis);

	return E_NOTIMPL;
```

```
}
```

Next, add the implementation codes for `DataObject`.

```
// CEx32cDoc::XDataObject
// delegate many calls to embedded COleDataSource object, which manages formats
STDMETHODIMP_(ULONG) CEx32cDoc::XDataObject::AddRef()
{
      TRACE("CEx32cDoc::XDataObject::AddRef\n");
      METHOD_PROLOGUE(CEx32cDoc, DataObject)
      return pThis->InternalAddRef();
}

STDMETHODIMP_(ULONG) CEx32cDoc::XDataObject::Release()
{
      TRACE("CEx32cDoc::XDataObject::Release\n");
      METHOD_PROLOGUE(CEx32cDoc, DataObject)
      return pThis->InternalRelease();
}

STDMETHODIMP CEx32cDoc::XDataObject::QueryInterface(
      REFIID iid, LPVOID* ppvObj)
{
      ITrace(iid, "CEx32cDoc::XDataObject::QueryInterface");
      METHOD_PROLOGUE(CEx32cDoc, DataObject)
      return pThis->InternalQueryInterface(&iid, ppvObj);
}

STDMETHODIMP CEx32cDoc::XDataObject::GetData(
      LPFORMATETC lpFormatEtc, LPSTGMEDIUM lpStgMedium)
{
      TRACE("CEx32cDoc::XDataObject::GetData -- %d\n",
              lpFormatEtc->cfFormat);
      METHOD_PROLOGUE(CEx32cDoc, DataObject)
      ASSERT_VALID(pThis);

      if(lpFormatEtc->cfFormat != CF_METAFILEPICT) {
            return S_FALSE;
      }
      HGLOBAL hPict = pThis->MakeMetaFile();
      lpStgMedium->tymed = TYMED_MFPICT;
      lpStgMedium->hMetaFilePict = hPict;
      lpStgMedium->pUnkForRelease = NULL;
      return S_OK;
}

STDMETHODIMP CEx32cDoc::XDataObject::GetDataHere(
      LPFORMATETC lpFormatEtc, LPSTGMEDIUM lpStgMedium)
{
      TRACE("CEx32cDoc::XDataObject::GetDataHere\n");
      METHOD_PROLOGUE(CEx32cDoc, DataObject)
      ASSERT_VALID(pThis);
      return E_NOTIMPL;
}

STDMETHODIMP CEx32cDoc::XDataObject::QueryGetData(LPFORMATETC lpFormatEtc)
{
```

```
	TRACE("CEx32cDoc::XDataObject::QueryGetData -- %d\n", lpFormatEtc-
>cfFormat);
	METHOD_PROLOGUE(CEx32cDoc, DataObject)
	ASSERT_VALID(pThis);
	if(lpFormatEtc->cfFormat != CF_METAFILEPICT) {
		return S_FALSE;
	}
	return S_OK;
}

STDMETHODIMP CEx32cDoc::XDataObject::GetCanonicalFormatEtc(
	LPFORMATETC /*lpFormatEtcIn*/, LPFORMATETC /*lpFormatEtcOut*/)
{
	TRACE("CEx32cDoc::XDataObject::GetCanonicalFormatEtc\n");
	return DATA_S_SAMEFORMATETC;
}

STDMETHODIMP CEx32cDoc::XDataObject::SetData(
	LPFORMATETC lpFormatEtc, LPSTGMEDIUM lpStgMedium, BOOL bRelease)
{
	TRACE("CEx32cDoc::XDataObject::SetData\n");
	METHOD_PROLOGUE(CEx32cDoc, DataObject)
	ASSERT_VALID(pThis);
	return E_NOTIMPL;
}

STDMETHODIMP CEx32cDoc::XDataObject::EnumFormatEtc(DWORD dwDirection,
                                                   LPENUMFORMATETC*
ppenumFormatEtc)
{
	TRACE("CEx32cDoc::XDataObject::EnumFormatEtc\n");
	METHOD_PROLOGUE(CEx32cDoc, DataObject)
	ASSERT_VALID(pThis);
	return E_NOTIMPL;
}

STDMETHODIMP CEx32cDoc::XDataObject::DAdvise(FORMATETC* pFormatEtc,
	DWORD advf, LPADVISESINK pAdvSink, DWORD* pdwConnection)
{
	TRACE("CEx32cDoc::XDataObject::DAdvise\n");
	METHOD_PROLOGUE(CEx32cDoc, DataObject)
	ASSERT_VALID(pThis);

	*pdwConnection = 555;
	// create the advise holder, if necessary
	if (pThis->m_lpDataAdviseHolder == NULL &&
		CreateDataAdviseHolder(&pThis->m_lpDataAdviseHolder) != S_OK) {
		return E_OUTOFMEMORY;
	}
	ASSERT(pThis->m_lpDataAdviseHolder != NULL);
	HRESULT hr = pThis->m_lpDataAdviseHolder->Advise(this, pFormatEtc, advf,
		pAdvSink, pdwConnection);
	return hr;
}

STDMETHODIMP CEx32cDoc::XDataObject::DUnadvise(DWORD dwConnection)
{
```

```
	TRACE("CEx32cDoc::XDataObject::DUnadvise\n");
	METHOD_PROLOGUE(CEx32cDoc, DataObject)
	ASSERT_VALID(pThis);

	return E_NOTIMPL;
}

STDMETHODIMP CEx32cDoc::XDataObject::EnumDAdvise(
	LPENUMSTATDATA* ppenumAdvise)
{
	TRACE("CEx32cDoc::XDataObject::EnumDAdvise\n");
	METHOD_PROLOGUE(CEx32cDoc, DataObject)
	ASSERT_VALID(pThis);

	return E_NOTIMPL;
}
```

Finally for `PersistStorage`.

```
////////////////////////////////////////////////////////
// XPersistStorage
STDMETHODIMP_(ULONG) CEx32cDoc::XPersistStorage::AddRef()
{
	TRACE("CEx32cDoc::XPersistStorage:::AddRef\n");
	METHOD_PROLOGUE(CEx32cDoc, PersistStorage)
	return pThis->InternalAddRef();
}

STDMETHODIMP_(ULONG) CEx32cDoc::XPersistStorage::Release()
{
	TRACE("CEx32cDoc::XPersistStorage::Release\n");
	METHOD_PROLOGUE(CEx32cDoc, PersistStorage)
	return pThis->InternalRelease();
}

STDMETHODIMP CEx32cDoc::XPersistStorage::QueryInterface(
	REFIID iid, LPVOID* ppvObj)
{
	ITrace(iid, "CEx32cDoc::XPersistStorage::QueryInterface");
	METHOD_PROLOGUE(CEx32cDoc, PersistStorage)
	return pThis->InternalQueryInterface(&iid, ppvObj);
}

STDMETHODIMP CEx32cDoc::XPersistStorage::GetClassID(LPCLSID lpClassID)
{
	TRACE("CEx32cDoc::XPersistStorage::GetClassID\n");
	METHOD_PROLOGUE(CEx32cDoc, PersistStorage)
	ASSERT_VALID(pThis);
	*lpClassID = clsid;
	return NOERROR;
}

STDMETHODIMP CEx32cDoc::XPersistStorage::IsDirty()
{
	TRACE("CEx32cDoc::XPersistStorage::IsDirty\n");
	METHOD_PROLOGUE(CEx32cDoc, PersistStorage)
	ASSERT_VALID(pThis);
```

```
	return pThis->IsModified() ? NOERROR : S_FALSE;
}

STDMETHODIMP CEx32cDoc::XPersistStorage::InitNew(LPSTORAGE pStg)
{
	TRACE("CEx32cDoc::XPersistStorage::InitNew\n");
	METHOD_PROLOGUE(CEx32cDoc, PersistStorage)
	ASSERT_VALID(pThis);
	ASSERT(pStg != NULL);

	pThis->SetModifiedFlag();  // new storage-based documents are dirty!
	pThis->SendInitialUpdate(); // in CDocument
	return NOERROR;
}

STDMETHODIMP CEx32cDoc::XPersistStorage::Load(LPSTORAGE pStgLoad)
{
	TRACE("CEx32cDoc::XPersistStorage::Load\n");
	METHOD_PROLOGUE(CEx32cDoc, PersistStorage)
	ASSERT_VALID(pThis);

	ASSERT(pStgLoad != NULL);
	LPSTREAM pStream;
	ULONG nBytesRead;
	char buffer[101]; // 100 characters max for m_strText
	try {
		pThis->DeleteContents();
		if(pStgLoad->OpenStream(L"Ex28c Text", NULL,
				STGM_READ|STGM_SHARE_EXCLUSIVE,
				0, &pStream) == NOERROR) {
			pStream->Read(buffer, 100, &nBytesRead);
			pStream->Release();
			pThis->m_strText = buffer;
		}
	}
	catch(CException* pe) {
		pe->Delete();
		return E_FAIL;
	}

	pThis->SetModifiedFlag();  // new storage-based documents are dirty!
	pThis->SendInitialUpdate(); // in CDocument
	return NOERROR;
}

STDMETHODIMP CEx32cDoc::XPersistStorage::Save(LPSTORAGE pStgSave,
                                              BOOL fSameAsLoad)
{
	TRACE("CEx32cDoc::XPersistStorage::Save\n");
	METHOD_PROLOGUE(CEx32cDoc, PersistStorage)
	ASSERT_VALID(pThis);

	// don't bother saving if destination is up-to-date
	if (fSameAsLoad && !pThis->IsModified())
		return NOERROR;
```

```
	ASSERT(pStgSave != NULL);
	LPSTREAM pStream;
	ULONG nBytesWritten;
	try   {
		if(pStgSave->CreateStream(L"Ex28c Text",
				STGM_CREATE|STGM_READWRITE|STGM_SHARE_EXCLUSIVE,
				0, 0, &pStream) == NOERROR) {
			pStream->Write((const char*) pThis->m_strText,
					pThis->m_strText.GetLength() + 1, &nBytesWritten);
			pStream->Release();
		}
		else return E_FAIL;
	}
	catch(CException* pe) {
		pe->Delete();
		return E_FAIL;
	}

	pThis->SetModifiedFlag();  // new storage-based documents are dirty!
	pThis->SendInitialUpdate(); // in CDocument
	return NOERROR;
}

STDMETHODIMP CEx32cDoc::XPersistStorage::SaveCompleted(LPSTORAGE pStgSaved)
{
	TRACE("CEx32cDoc::XPersistStorage::SaveCompleted\n");
	METHOD_PROLOGUE(CEx32cDoc, PersistStorage)
	ASSERT_VALID(pThis);

	return E_NOTIMPL;
}

STDMETHODIMP CEx32cDoc::XPersistStorage::HandsOffStorage()
{
	TRACE("CEx32cDoc::XPersistStorage::HandsOffStorage\n");
	METHOD_PROLOGUE(CEx32cDoc, PersistStorage)
	ASSERT_VALID(pThis);

	return E_NOTIMPL;
}
```

Add and/or edit other codes starting from the constructor.

```
CEx32cDoc::CEx32cDoc()
{
	// TODO: add one-time construction code here
	TRACE("CEx32cDoc ctor\n");
	m_lpClientSite = NULL;
	m_lpOleAdviseHolder = NULL;
	m_lpDataAdviseHolder = NULL;
}

CEx32cDoc::~CEx32cDoc()
{
	TRACE("CEx32cDoc dtor\n");
}
```

```
BOOL CEx32cDoc::OnNewDocument()
{
	TRACE("CEx32cDoc::OnNewDocument\n");
	if (!CDocument::OnNewDocument())
		return FALSE;
	m_strText = "Default text";
	return TRUE;
}
```

```
// CEx32cDoc construction/destruction

CEx32cDoc::CEx32cDoc()
{
    // TODO: add one-time construction code here
    TRACE("CEx32cDoc ctor\n");
    m_lpClientSite = NULL;
    m_lpOleAdviseHolder = NULL;
    m_lpDataAdviseHolder = NULL;
}

CEx32cDoc::~CEx32cDoc()
{
    TRACE("CEx32cDoc dtor\n");
}

BOOL CEx32cDoc::OnNewDocument()
{
    TRACE("CEx32cDoc::OnNewDocument\n");
    if (!CDocument::OnNewDocument())
        return FALSE;
    m_strText = "Default text";
    return TRUE;
}
```

Listing 14.

This function is manually added if you not using ClassView during the adding of the `MakeMetaFile()` declaration in **ex32cDoc.h**.

```
HGLOBAL CEx32cDoc::MakeMetaFile()
{
	HGLOBAL hPict;
	CMetaFileDC dcm;
	VERIFY(dcm.Create());
	CSize size(5000, 5000); // initial size of object in Excel & Word
	dcm.SetMapMode(MM_ANISOTROPIC);
	dcm.SetWindowOrg(0,0);
	dcm.SetWindowExt(size.cx, -size.cy);
	// drawing code
	dcm.Rectangle(CRect(500, -1000, 1500, -2000));
    CFont font;
    font.CreateFont(-500, 0, 0, 0, 400, FALSE, FALSE, 0,
                    ANSI_CHARSET, OUT_DEFAULT_PRECIS,
                    CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
                    DEFAULT_PITCH | FF_SWISS, "Arial");
    CFont* pFont = dcm.SelectObject(&font);
	dcm.TextOut(0, 0, m_strText);
    dcm.SelectObject(pFont);

	HMETAFILE hMF = dcm.Close();
```

```
    ASSERT(hMF != NULL);
    hPict = ::GlobalAlloc(GMEM_SHARE|GMEM_MOVEABLE, sizeof(METAFILEPICT));
    ASSERT(hPict != NULL);
    LPMETAFILEPICT lpPict;
    lpPict = (LPMETAFILEPICT) ::GlobalLock(hPict);
    ASSERT(lpPict != NULL);
    lpPict->mm = MM_ANISOTROPIC;
    lpPict->hMF = hMF;
    lpPict->xExt = size.cx;
    lpPict->yExt = size.cy;  // HIMETRIC height
    ::GlobalUnlock(hPict);
    return hPict;
}
```

```
HGLOBAL CEx32cDoc::MakeMetaFile()
{
    HGLOBAL hPict;
    CMetaFileDC dcm;
    VERIFY(dcm.Create());
    CSize size(5000, 5000); // initial size of object in Excel & Word
    dcm.SetMapMode(MM_ANISOTROPIC);
    dcm.SetWindowOrg(0,0);
    dcm.SetWindowExt(size.cx, -size.cy);
    // drawing code
    dcm.Rectangle(CRect(500, -1000, 1500, -2000));
    CFont font;
    font.CreateFont(-500, 0, 0, 0, 400, FALSE, FALSE, 0,
                    ANSI_CHARSET, OUT_DEFAULT_PRECIS,
                    CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
                    DEFAULT_PITCH | FF_SWISS, "Arial");
    CFont* pFont = dcm.SelectObject(&font);
    dcm.TextOut(0, 0, m_strText);
    dcm.SelectObject(pFont);

    HMETAFILE hMF = dcm.Close();
    ASSERT(hMF != NULL);
    hPict = ::GlobalAlloc(GMEM_SHARE|GMEM_MOVEABLE, sizeof(METAFILEPICT));
    ASSERT(hPict != NULL);
    LPMETAFILEPICT lpPict;
    lpPict = (LPMETAFILEPICT) ::GlobalLock(hPict);
    ASSERT(lpPict != NULL);
    lpPict->mm = MM_ANISOTROPIC;
    lpPict->hMF = hMF;
    lpPict->xExt = size.cx;
    lpPict->yExt = size.cy;  // HIMETRIC height
    ::GlobalUnlock(hPict);
    return hPict;
}
```

Listing 15.

Add the following global diagnostic jus after the `Dump()`.

```
// global diagnostic function
void ITrace(REFIID iid, const char* str)
{
    LPOLESTR lpszIID;
    ::StringFromIID(iid, &lpszIID);
    CString strIID = lpszIID;
    TRACE("%s - %s\n", (const char*) strIID, (const char*) str);
    AfxFreeTaskMem(lpszIID);
}
```

Listing 16.

Edit the `OnCloseDocument()`.

```
void CEx32cDoc::OnCloseDocument()
{
    // TODO: Add your specialized code here and/or call the base class
    InternalAddRef();

    if(m_lpClientSite != NULL) {
        m_lpClientSite->OnShowWindow(FALSE); // no hatch
        m_lpClientSite->Release();
    }

    if (m_lpOleAdviseHolder != NULL)
    // you need to send a close notification
        m_lpOleAdviseHolder->SendOnClose();
    // finish closing the document (before m_lpClientSite->Release)
    BOOL bAutoDelete = m_bAutoDelete;
    m_bAutoDelete = FALSE;
    CDocument::OnCloseDocument();

    // disconnect the object
    LPUNKNOWN lpUnknown = (LPUNKNOWN)GetInterface(&IID_IUnknown);
    ASSERT(lpUnknown != NULL);
    // this is very important to close circular references
    CoDisconnectObject(lpUnknown, 0);

    if(m_lpOleAdviseHolder != NULL)
        m_lpOleAdviseHolder->Release();
    if(m_lpDataAdviseHolder != NULL)
        m_lpDataAdviseHolder->Release();

    m_lpClientSite = NULL;
    m_lpOleAdviseHolder = NULL;
    m_lpDataAdviseHolder = NULL;
    // remove InternalAddRef above
    InterlockedDecrement(&m_dwRef);
    if (bAutoDelete) {
        delete this;
      }
}
```

```
void CEx32cDoc::OnCloseDocument()
{
    // TODO: Add your specialized code here and/or call the base class
    InternalAddRef();

    if(m_lpClientSite != NULL) {
        m_lpClientSite->OnShowWindow(FALSE); // no hatch
        m_lpClientSite->Release();
    }

    if (m_lpOleAdviseHolder != NULL)
    // you need to send a close notification
        m_lpOleAdviseHolder->SendOnClose();
    // finish closing the document (before m_lpClientSite->Release)
    BOOL bAutoDelete = m_bAutoDelete;
    m_bAutoDelete = FALSE;
    CDocument::OnCloseDocument();

    // disconnect the object
    LPUNKNOWN lpUnknown = (LPUNKNOWN)GetInterface(&IID_IUnknown);
    ASSERT(lpUnknown != NULL);
    // this is very important to close circular references
    CoDisconnectObject(lpUnknown, 0);

    if(m_lpOleAdviseHolder != NULL)
        m_lpOleAdviseHolder->Release();
    if(m_lpDataAdviseHolder != NULL)
        m_lpDataAdviseHolder->Release();

    m_lpClientSite = NULL;
    m_lpOleAdviseHolder = NULL;
    m_lpDataAdviseHolder = NULL;
    // remove InternalAddRef above
    InterlockedDecrement(&m_dwRef);
    if (bAutoDelete) {
        delete this;
    }
}
```

Listing 17.

Add/edit other codes.

```
void CEx32cDoc::OnModify()
{
      // TODO: Add your command handler code here
      CTextDialog dlg;
      dlg.m_strText = m_strText;
      if(dlg.DoModal() == IDOK) {
            m_strText = dlg.m_strText;
            UpdateAllViews(NULL); // redraw view
            // Notify the client that the metafile has changed.
            //  Client must call IViewObject::SetAdvise.
            LPDATAOBJECT lpDataObject =
                  (LPDATAOBJECT) GetInterface(&IID_IDataObject);
            HRESULT hr =
                  m_lpDataAdviseHolder->SendOnDataChange(lpDataObject, 0, NULL);
            ASSERT(hr == NOERROR);
            SetModifiedFlag(); // won't update without this
      }
}
```

```
void CEx32cDoc::OnModify()
{
    // TODO: Add your command handler code here
    CTextDialog dlg;
    dlg.m_strText = m_strText;
    if(dlg.DoModal() == IDOK) {
        m_strText = dlg.m_strText;
        UpdateAllViews(NULL); // redraw view
        // Notify the client that the metafile has changed.
        //  Client must call IViewObject::SetAdvise.
        LPDATAOBJECT lpDataObject =
            (LPDATAOBJECT) GetInterface(&IID_IDataObject);
        HRESULT hr =
            m_lpDataAdviseHolder->SendOnDataChange(lpDataObject, 0, NULL);
        ASSERT(hr == NOERROR);
        SetModifiedFlag(); // won't update without this
    }
}
```

Listing 18.

```
void CEx32cDoc::OnFinalRelease()
{
      // TODO: Add your specialized code here and/or call the base class
      TRACE("CEx32cDoc::OnFinalRelease\n"); // so we can see it happen
      CDocument::OnFinalRelease();
}
```

```
void CEx32cDoc::OnFinalRelease()
{
    // TODO: Add your specialized code here and/or call the base class
    TRACE("CEx32cDoc::OnFinalRelease\n"); // so we can see it happen
    CDocument::OnFinalRelease();
}
```

Listing 19.

```
BOOL CEx32cDoc::SaveModified()
{
      // TODO: Add your specialized code here and/or call the base class
      OnFileUpdate();
      return TRUE;
}
```

```
BOOL CEx32cDoc::SaveModified()
{
    // TODO: Add your specialized code here and/or call the base class
    OnFileUpdate();
    return TRUE;
}
```

Listing 20.

```
void CEx32cDoc::OnFileUpdate()
{
      // TODO: Add your command handler code here
      if(m_lpClientSite == NULL) return;
      VERIFY(m_lpClientSite->SaveObject() == NOERROR);
      if (m_lpOleAdviseHolder != NULL)
            m_lpOleAdviseHolder->SendOnSave();
```

```
	SetModifiedFlag(FALSE);
}

void CEx32cDoc::OnUpdateFileUpdate(CCmdUI* pCmdUI)
{
	// TODO: Add your command update UI handler code here
	pCmdUI->Enable(IsModified());
}
```

```
void CEx32cDoc::OnFileUpdate()
{
    // TODO: Add your command handler code here
    if(m_lpClientSite == NULL) return;
    VERIFY(m_lpClientSite->SaveObject() == NOERROR);
      if (m_lpOleAdviseHolder != NULL)
            m_lpOleAdviseHolder->SendOnSave();
    SetModifiedFlag(FALSE);
}

void CEx32cDoc::OnUpdateFileUpdate(CCmdUI* pCmdUI)
{
    // TODO: Add your command update UI handler code here
    pCmdUI->Enable(IsModified());
}
```

Listing 21.

Add the following `#include` directive in **StdAfx.h**.

```
#include <afxole.h>         // MFC OLE classes
```

And delete (or comment out) the following `#include` directives.

```
#include <afxdisp.h>
#include <afxdtctl.h>
```

```
#include <afxwin.h>          // MFC core and standard components
#include <afxext.h>          // MFC extensions
// #include <afxdisp.h>        // MFC Automation classes
// #include <afxdtctl.h>        // MFC support for Internet Exp
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>          // MFC support for Windows Common C
#endif // _AFX_NO_AFXCMN_SUPPORT

#include <afxole.h>         // MFC OLE classes

//{{AFX_INSERT_LOCATION}}
```

Listing 22.

Finally, just delete the ODL file (**ex32c.odl**) or leave it empty as shown below. Select the file and use the **Edit Delete** menu.

```
// ex32c.odl : type library source for ex32c.exe

// This file will be processed by the MIDL compiler to produce the
// type library (ex32c.tlb).

[ uuid(85CD200B-4130-4077-9B26-F97F6B22A7CC), version(1.0) ]
library Ex32c
{
    importlib("stdole32.tlb");
    importlib("stdole2.tlb");


        //{{AFX_APPEND_ODL}}
    //}}AFX_APPEND_ODL}}
};
```

Listing 23.

Build the program. When the program is run standalone, the following prompt will be displayed.

Figure 23: EX32C output.

You can use EX32B program to test this server program. Launch EX32B and select **Edit Insert Object** menu as shown below.

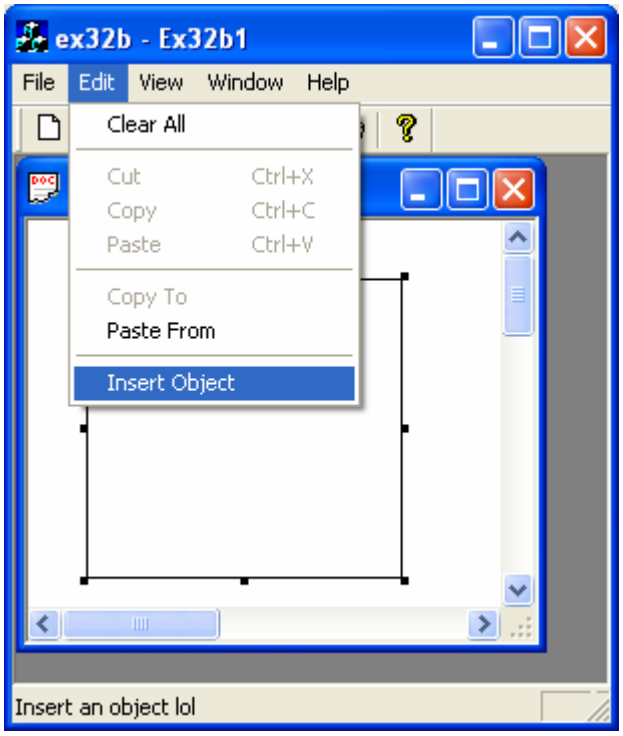

Figure 24: Using EX32B to test EX32C.

Select EX32c's object, **Ex32c Document**.

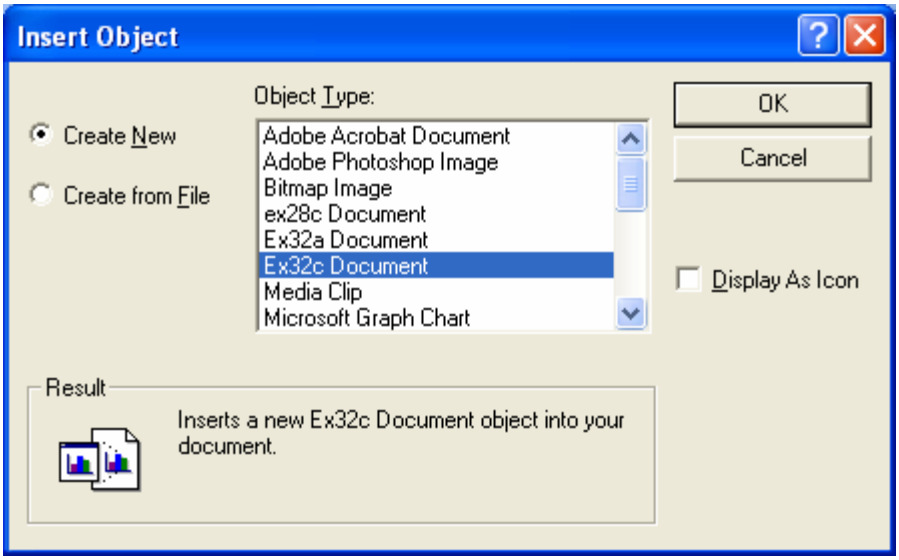

Figure 25: Selecting EX32C's object.

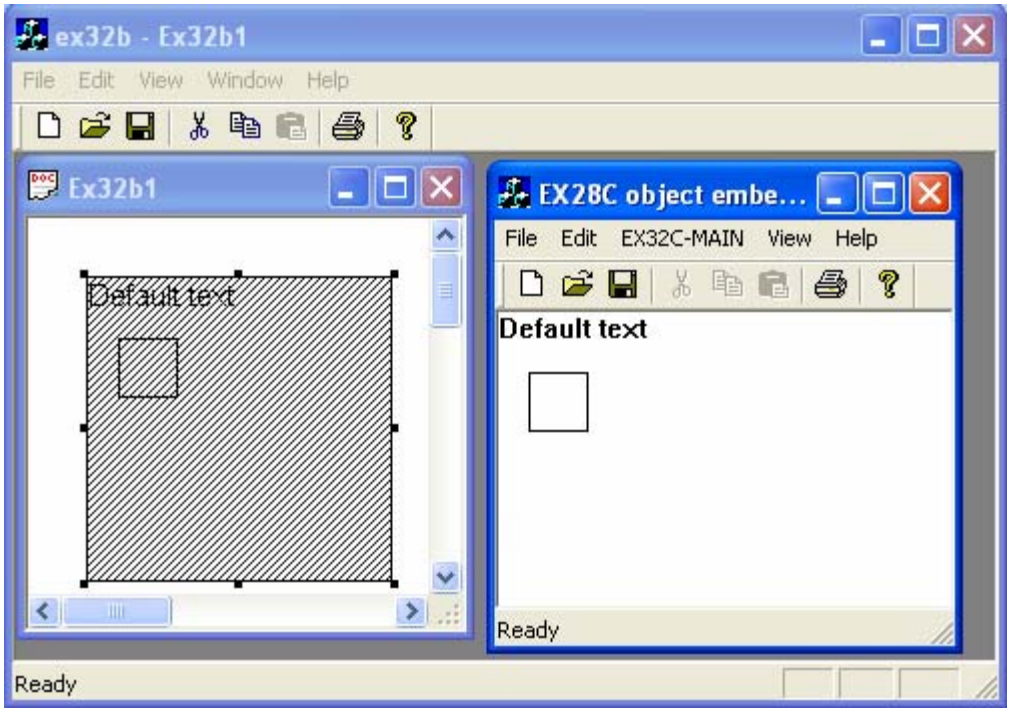

Figure 26: EX32C object in embedded mode, side-by-side with EX32B.

Select **Modify** menu.

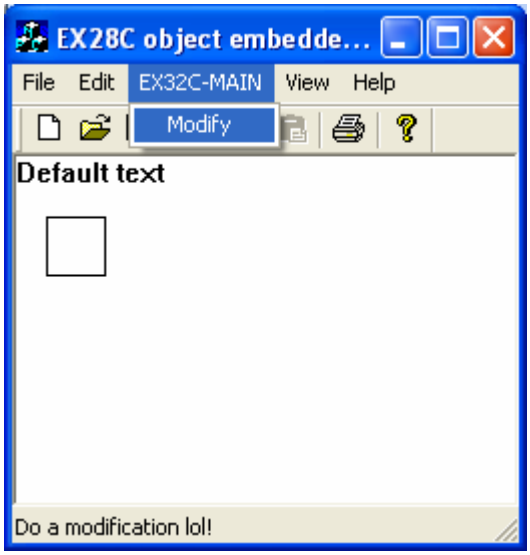

Figure 27: Testing the **Modify** menu in embedded mode.

Enter some string to see the update and click the **OK** button.

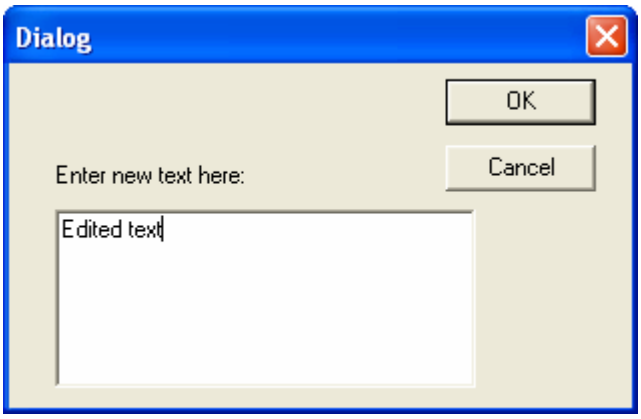

Figure 28: Trying some new string.

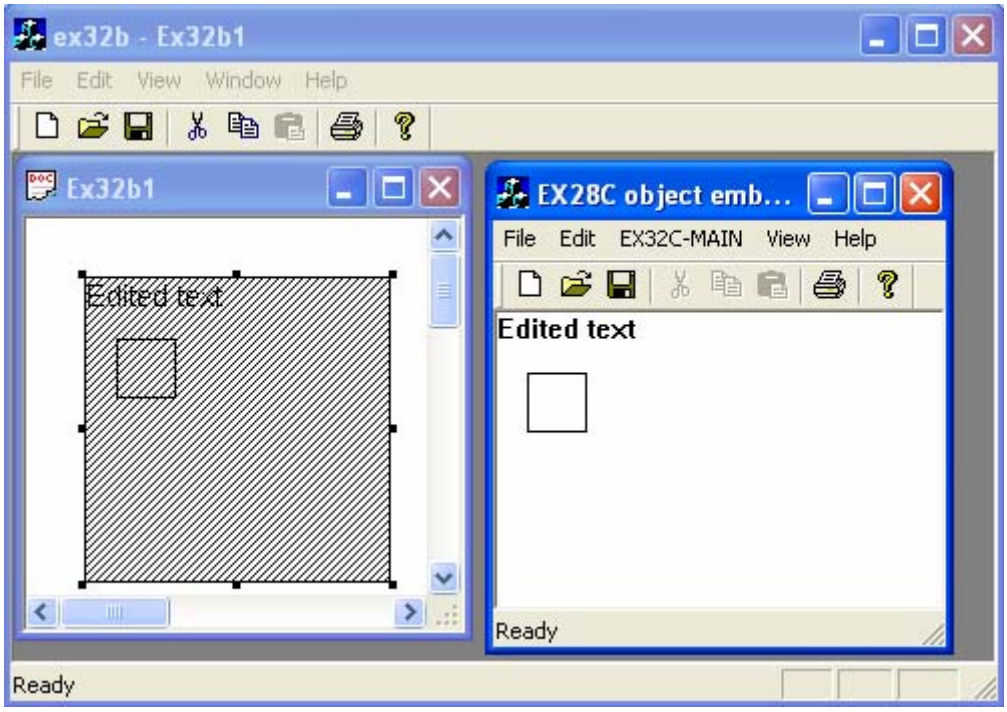

Figure 29: The new string was updated, in-place and embedded, side-by-side.

The Story

The `CEx32cView` Class

This class is straightforward. The only member functions of interest are the `OnDraw()` function and the `OnPrepareDC()` function, shown here:

```
void CEx32cView::OnDraw(CDC* pDC)
{
    CEx32cDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDC->Rectangle(CRect(500, -1000, 1500, -2000));
    pDC->TextOut(0, 0, pDoc->m_strText);
```

```
}

void CEx32cView::OnPrepareDC(CDC* pDC, CPrintInfo* pInfo)
{
    pDC->SetMapMode(MM_HIMETRIC);

}
```

The CEx32cDoc Class

This class does most of the component's work and is too big to list here. Listing 24 lists the header file, for other source codes, please refer to the program. A few of the important functions are listed here, however.

EX32CDOC.H

```
// ex32cDoc.h : interface of the CEx32cDoc class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_EX32CDOC_H__17B0D8FA_4B3A_45C6_A94F_3BC4EA49A0DE__INCLUDED_)
#define AFX_EX32CDOC_H__17B0D8FA_4B3A_45C6_A94F_3BC4EA49A0DE__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

extern const CLSID clsid; // defined in ex32c.cpp
void ITrace(REFIID iid, const char* str);

#define SETFORMATETC(fe, cf, asp, td, med, li) \
       ((fe).cfFormat=cf, \
        (fe).dwAspect=asp, \
        (fe).ptd=td, \
        (fe).tymed=med, \
        (fe).lindex=li)

class CEx32cDoc : public CDocument
{
friend class CEx32cView;

private:
       CString m_strText;
       LPOLECLIENTSITE m_lpClientSite;
       LPOLEADVISEHOLDER m_lpOleAdviseHolder;
       LPDATAADVISEHOLDER m_lpDataAdviseHolder;
       CString m_strContainerApp;
       CString m_strContainerObj;

       HGLOBAL MakeMetaFile();

       BEGIN_INTERFACE_PART(OleObject, IOleObject)
              STDMETHOD(SetClientSite)(LPOLECLIENTSITE);
              STDMETHOD(GetClientSite)(LPOLECLIENTSITE*);
              STDMETHOD(SetHostNames)(LPCOLESTR, LPCOLESTR);
              STDMETHOD(Close)(DWORD);
              STDMETHOD(SetMoniker)(DWORD, LPMONIKER);
              STDMETHOD(GetMoniker)(DWORD, DWORD, LPMONIKER*);
              STDMETHOD(InitFromData)(LPDATAOBJECT, BOOL, DWORD);
              STDMETHOD(GetClipboardData)(DWORD, LPDATAOBJECT*);
              STDMETHOD(DoVerb)(LONG, LPMSG, LPOLECLIENTSITE, LONG,
                     HWND, LPCRECT);
```

```
		STDMETHOD(EnumVerbs)(LPENUMOLEVERB*);
		STDMETHOD(Update)();
		STDMETHOD(IsUpToDate)();
		STDMETHOD(GetUserClassID)(LPCLSID);
		STDMETHOD(GetUserType)(DWORD, LPOLESTR*);
		STDMETHOD(SetExtent)(DWORD, LPSIZEL);
		STDMETHOD(GetExtent)(DWORD, LPSIZEL);
		STDMETHOD(Advise)(LPADVISESINK, LPDWORD);
		STDMETHOD(Unadvise)(DWORD);
		STDMETHOD(EnumAdvise)(LPENUMSTATDATA*);
		STDMETHOD(GetMiscStatus)(DWORD, LPDWORD);
		STDMETHOD(SetColorScheme)(LPLOGPALETTE);
	END_INTERFACE_PART(OleObject)

	BEGIN_INTERFACE_PART(DataObject, IDataObject)
		STDMETHOD(GetData)(LPFORMATETC, LPSTGMEDIUM);
		STDMETHOD(GetDataHere)(LPFORMATETC, LPSTGMEDIUM);
		STDMETHOD(QueryGetData)(LPFORMATETC);
		STDMETHOD(GetCanonicalFormatEtc)(LPFORMATETC, LPFORMATETC);
		STDMETHOD(SetData)(LPFORMATETC, LPSTGMEDIUM, BOOL);
		STDMETHOD(EnumFormatEtc)(DWORD, LPENUMFORMATETC*);
		STDMETHOD(DAdvise)(LPFORMATETC, DWORD, LPADVISESINK, LPDWORD);
		STDMETHOD(DUnadvise)(DWORD);
		STDMETHOD(EnumDAdvise)(LPENUMSTATDATA*);
	END_INTERFACE_PART(DataObject)

	BEGIN_INTERFACE_PART(PersistStorage, IPersistStorage)
		STDMETHOD(GetClassID)(LPCLSID);
		STDMETHOD(IsDirty)();
		STDMETHOD(InitNew)(LPSTORAGE);
		STDMETHOD(Load)(LPSTORAGE);
		STDMETHOD(Save)(LPSTORAGE, BOOL);
		STDMETHOD(SaveCompleted)(LPSTORAGE);
		STDMETHOD(HandsOffStorage)();
	END_INTERFACE_PART(PersistStorage)

	DECLARE_INTERFACE_MAP()

protected: // create from serialization only
	CEx32cDoc();
	DECLARE_DYNCREATE(CEx32cDoc)

// Attributes
public:

// Operations
public:

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CEx32cDoc)
	public:
	virtual BOOL OnNewDocument();
	virtual void Serialize(CArchive& ar);
	virtual void OnCloseDocument();
	virtual void OnFinalRelease();
	protected:
	virtual BOOL SaveModified();
	//}}AFX_VIRTUAL

// Implementation
public:
	virtual ~CEx32cDoc();
```

```
#ifdef _DEBUG
       virtual void AssertValid() const;
       virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
       //{{AFX_MSG(CEx32cDoc)
       afx_msg void OnModify();
       afx_msg void OnFileUpdate();
       afx_msg void OnUpdateFileUpdate(CCmdUI* pCmdUI);
       //}}AFX_MSG
       DECLARE_MESSAGE_MAP()

//     // Generated OLE dispatch map functions
//     //{{AFX_DISPATCH(CEx32cDoc)
//            // NOTE - the ClassWizard will add and remove member functions here.
//            //    DO NOT EDIT what you see in these blocks of generated code !
//     //}}AFX_DISPATCH
//     DECLARE_DISPATCH_MAP()
//     DECLARE_INTERFACE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif // !defined(AFX_EX32CDOC_H__17B0D8FA_4B3A_45C6_A94F_3BC4EA49A0DE__INCLUDED_)
```

Listing 24: The component's `CEx32cDoc` class handler file listing.

Here's a list of the important interface functions in **ex32cDoc.cpp**:

```
XOleObject::SetClientSite
XOleObject::DoVerb
XOleObject::Advise
XDataObject::GetData
XDataObject::QueryGetData
XDataObject::DAdvise
XPersistStorage::GetClassID
XPersistStorage::InitNew
XPersistStorage::Load
XPersistStorage::Save
```

You've seen the container code that draws a metafile. Here's the component code that creates it. The object handler calls the component's `XDataObject::GetData` function when it needs a metafile. This `GetData()` implementation calls a helper function, `MakeMetaFile()`, which creates the metafile picture. Compare the drawing code with the drawing code in `CEx32cView::OnDraw`.

```
STDMETHODIMP CEx32cDoc::XDataObject::GetData(
    LPFORMATETC lpFormatEtc, LPSTGMEDIUM lpStgMedium)
{
    TRACE("CEx32cDoc::XDataObject::GetData -- %d\n", lpFormatEtc->cfFormat);
    METHOD_PROLOGUE(CEx32cDoc, DataObject)
    ASSERT_VALID(pThis);
```

```
    if (lpFormatEtc->cfFormat != CF_METAFILEPICT)
    {
        return S_FALSE;
    }
    HGLOBAL hPict = pThis->MakeMetaFile();
    lpStgMedium->tymed = TYMED_MFPICT;
    lpStgMedium->hMetaFilePict = hPict;
    lpStgMedium->pUnkForRelease = NULL;
    return S_OK;
}

HGLOBAL CEx32cDoc::MakeMetaFile
{
    HGLOBAL hPict;
    CMetaFileDC dcm;
    VERIFY(dcm.Create());
    CSize size(5000, 5000); // initial size of object in Excel & Word
    dcm.SetMapMode(MM_ANISOTROPIC);
    dcm.SetWindowOrg(0, 0);
    dcm.SetWindowExt(size.cx, -size.cy);
    // drawing code
    dcm.Rectangle(CRect(500, -1000, 1500, -2000));
    CFont font;
    font.CreateFont(-500, 0, 0, 0, 400, FALSE, FALSE, 0,
                    ANSI_CHARSET, OUT_DEFAULT_PRECIS,
                    CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
                    DEFAULT_PITCH | FF_SWISS, "Arial");
    CFont* pFont = dcm.SelectObject(&font);
    dcm.TextOut(0, 0, m_strText);
    dcm.SelectObject(pFont);

    HMETAFILE hMF = dcm.Close();
    ASSERT(hMF != NULL);
    hPict = ::GlobalAlloc(GMEM_SHARE|GMEM_MOVEABLE, sizeof(METAFILEPICT));
    ASSERT(hPict != NULL);
    LPMETAFILEPICT lpPict;
    lpPict = (LPMETAFILEPICT)::GlobalLock(hPict);
    ASSERT(lpPict != NULL);
    lpPict->mm = MM_ANISOTROPIC;
    lpPict->hMF = hMF;
    lpPict->xExt = size.cx;
    lpPict->yExt = size.cy;  // HIMETRIC height
    ::GlobalUnlock(hPict);
    return hPict;
}
```

The `XOleObject::Advise` and the `XDataObject::DAdvise` functions are similar. Both functions call global OLE functions to set up OLE advise holder objects that can manage multiple advise sinks. In this program, there is only one advise sink per OLE advise holder object. The `XOleObject::Advise` function, listed below, establishes an OLE advise holder object with the `IOleAdviseHolder` interface. Other document functions call `IOleAdviseHolder::SendOnClose` and `SendOnSave()`, which in turn call `IAdviseSink::OnClose` and `OnSave()` for each attached sink.

```
STDMETHODIMP CEx32cDoc::XOleObject::Advise(
    IAdviseSink* pAdvSink, DWORD* pdwConnection)
```

```
{
    TRACE("CEx32cDoc::XOleObject::Advise\n");
    METHOD_PROLOGUE(CEx32cDoc, OleObject)
    ASSERT_VALID(pThis);
    *pdwConnection = 0;
    if (pThis->m_lpOleAdviseHolder == NULL &&
        ::CreateOleAdviseHolder(&pThis->m_lpOleAdviseHolder)
        != NOERROR) {
            return E_OUTOFMEMORY;
    }
    ASSERT(pThis->m_lpOleAdviseHolder != NULL);
    return pThis->m_lpOleAdviseHolder->Advise(pAdvSink, pdwConnection);
}
```

The framework calls the `OnModify()` function when the user chooses **Modify** from the **EX32C-MAIN** menu. The user enters a string through a dialog, and the function sends the `OnDataChange()` notification to the object handler's data advise sink. Figure 28-5 illustrates the advisory connections. Here is the `OnModify()` function code:

```
void CEx32cDoc::OnModify()
{
    CTextDialog dlg;
    dlg.m_strText = m_strText;
    if (dlg.DoModal() == IDOK) {
        m_strText = dlg.m_strText;
        UpdateAllViews(NULL); // redraw view
        // Notify the client that the metafile has changed.
        //  Client must call IViewObject::SetAdvise.
        LPDATAOBJECT lpDataObject =
            (LPDATAOBJECT) GetInterface(&IID_IDataObject);
        HRESULT hr =
            m_lpDataAdviseHolder->SendOnDataChange(lpDataObject, 0, NULL);
        ASSERT(hr == NOERROR);
        SetModifiedFlag(); // won't update without this
    }
}
```

The framework calls the `OnFileUpdate()` function when the user chooses **Update** from the **File** menu. The function calls `IOleClientSite::SaveObject`, which in turn causes the container to save the metafile and the object's native data in the storage. The function also sends the `OnSave()` notification back to the client's advise sink. Here is the `OnFileUpdate()` function code:

```
void CEx32cDoc::OnFileUpdate()
{
    if (m_lpClientSite == NULL) return;
    VERIFY(m_lpClientSite->SaveObject() == NOERROR);
    if (m_lpOleAdviseHolder != NULL)
        m_lpOleAdviseHolder->SendOnSave();
    SetModifiedFlag(FALSE);
}
```

----------End part 3----------

Further reading and digging:

1. MSDN MFC 6.0 class library online documentation - used throughout this Tutorial.

2. MSDN MFC 7.0 class library online documentation - used in .Net framework and also backward compatible with 6.0 class library
3. MSDN Library
4. DCOM at MSDN.
5. COM+ at MSDN.
6. COM at MSDN.
7. Windows data type.
8. Win32 programming Tutorial.
9. The best of C/C++, MFC, Windows and other related books.
10. Unicode and Multibyte character set: Story and program examples.