#include "stdafx.h"

#include "DocViewCrystalPrint.h"

#include "MainFrm.h"

#include "DocViewCrystalPrintDoc.h"

#include "DocViewCrystalPrintView.h"

//{{AFX\_INCLUDES()

#include "crystalctrl.h"

//}}AFX\_INCLUDES

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE[] = \_\_FILE\_\_;

#endif

// CDocViewCrystalPrintApp

BEGIN\_MESSAGE\_MAP(CDocViewCrystalPrintApp, CWinApp)

//{{AFX\_MSG\_MAP(CDocViewCrystalPrintApp)

ON\_COMMAND(ID\_APP\_ABOUT, OnAppAbout)

// NOTE - the ClassWizard will add and remove mapping macros here.

// DO NOT EDIT what you see in these blocks of generated code!

//}}AFX\_MSG\_MAP

// Standard file based document commands

ON\_COMMAND(ID\_FILE\_NEW, CWinApp::OnFileNew)

ON\_COMMAND(ID\_FILE\_OPEN, CWinApp::OnFileOpen)

// Standard print setup command

ON\_COMMAND(ID\_FILE\_PRINT\_SETUP, CWinApp::OnFilePrintSetup)

END\_MESSAGE\_MAP()

// CDocViewCrystalPrintApp construction

CDocViewCrystalPrintApp::CDocViewCrystalPrintApp()

{

// TODO: add construction code here,

// Place all significant initialization in InitInstance

}

// The one and only CDocViewCrystalPrintApp object

CDocViewCrystalPrintApp theApp;

// CDocViewCrystalPrintApp initialization

BOOL CDocViewCrystalPrintApp::InitInstance()

{

AfxEnableControlContainer();

// Standard initialization

// If you are not using these features and wish to reduce the size

// of your final executable, you should remove from the following

// the specific initialization routines you do not need.

#ifdef \_AFXDLL

Enable3dControls(); // Call this when using MFC in a shared DLL

#else

Enable3dControlsStatic(); // Call this when linking to MFC statically

#endif

// Change the registry key under which our settings are stored.

// TODO: You should modify this string to be something appropriate

// such as the name of your company or organization.

SetRegistryKey(\_T("Local AppWizard-Generated Applications"));

LoadStdProfileSettings(); // Load standard INI file options (including MRU)

// Register the application's document templates. Document templates

// serve as the connection between documents, frame windows and views.

CSingleDocTemplate\* pDocTemplate;

pDocTemplate = new CSingleDocTemplate(

IDR\_MAINFRAME,

RUNTIME\_CLASS(CDocViewCrystalPrintDoc),

RUNTIME\_CLASS(CMainFrame), // main SDI frame window

RUNTIME\_CLASS(CDocViewCrystalPrintView));

AddDocTemplate(pDocTemplate);

// Parse command line for standard shell commands, DDE, file open

CCommandLineInfo cmdInfo;

ParseCommandLine(cmdInfo);

// 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;

}

// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog

{

public:

CAboutDlg();

// Dialog Data

//{{AFX\_DATA(CAboutDlg)

enum { IDD = IDD\_ABOUTBOX };

CCrystalCtrl m\_CrystalReport;

//}}AFX\_DATA

// ClassWizard generated virtual function overrides

//{{AFX\_VIRTUAL(CAboutDlg)

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV support

//}}AFX\_VIRTUAL

// Implementation

protected:

//{{AFX\_MSG(CAboutDlg)

virtual BOOL OnInitDialog();

//}}AFX\_MSG

DECLARE\_MESSAGE\_MAP()

};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)

{

//{{AFX\_DATA\_INIT(CAboutDlg)

//}}AFX\_DATA\_INIT

}

void CAboutDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialog::DoDataExchange(pDX);

//{{AFX\_DATA\_MAP(CAboutDlg)

DDX\_Control(pDX, IDC\_CRYSTALREPORT1, m\_CrystalReport);

//}}AFX\_DATA\_MAP

}

BEGIN\_MESSAGE\_MAP(CAboutDlg, CDialog)

//{{AFX\_MSG\_MAP(CAboutDlg)

//}}AFX\_MSG\_MAP

END\_MESSAGE\_MAP()

// App command to run the dialog

void CDocViewCrystalPrintApp::OnAppAbout()

{

CAboutDlg aboutDlg;

aboutDlg.DoModal();

}

// CDocViewCrystalPrintApp message handlers

#include "stdafx.h"

#include "crystalctrl.h"

// Dispatch interfaces referenced by this interface

#include "RowCursor.h"

// CCrystalCtrl

IMPLEMENT\_DYNCREATE(CCrystalCtrl, CWnd)

// CCrystalCtrl properties

// CCrystalCtrl operations

CString CCrystalCtrl::GetReportFileName()

{

CString result;

InvokeHelper(0x3, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetReportFileName(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x3, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

short CCrystalCtrl::GetWindowLeft()

{

short result;

InvokeHelper(0x4, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowLeft(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x4, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

short CCrystalCtrl::GetWindowTop()

{

short result;

InvokeHelper(0x5, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowTop(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x5, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

short CCrystalCtrl::GetWindowWidth()

{

short result;

InvokeHelper(0x6, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowWidth(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x6, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

short CCrystalCtrl::GetWindowHeight()

{

short result;

InvokeHelper(0x7, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowHeight(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x7, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

CString CCrystalCtrl::GetWindowTitle()

{

CString result;

InvokeHelper(0x8, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowTitle(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x8, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

BOOL CCrystalCtrl::GetWindowControlBox()

{

BOOL result;

InvokeHelper(0x9, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowControlBox(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x9, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCrystalCtrl::GetWindowMaxButton()

{

BOOL result;

InvokeHelper(0xa, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowMaxButton(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0xa, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCrystalCtrl::GetWindowMinButton()

{

BOOL result;

InvokeHelper(0xb, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowMinButton(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0xb, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

CString CCrystalCtrl::GetPrintFileName()

{

CString result;

InvokeHelper(0xc, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrintFileName(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0xc, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

CString CCrystalCtrl::GetSelectionFormula()

{

CString result;

InvokeHelper(0xd, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetSelectionFormula(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0xd, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

CString CCrystalCtrl::GetGroupSelectionFormula()

{

CString result;

InvokeHelper(0xe, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetGroupSelectionFormula(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0xe, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

long CCrystalCtrl::GetWindowParentHandle()

{

long result;

InvokeHelper(0xf, DISPATCH\_PROPERTYGET, VT\_I4, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowParentHandle(long nNewValue)

{

static BYTE parms[] =

VTS\_I4;

InvokeHelper(0xf, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

short CCrystalCtrl::GetCopiesToPrinter()

{

short result;

InvokeHelper(0x10, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetCopiesToPrinter(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x10, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

short CCrystalCtrl::GetAction()

{

short result;

InvokeHelper(0x11, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetAction(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x11, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

short CCrystalCtrl::GetLastErrorNumber()

{

short result;

InvokeHelper(0x12, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetLastErrorNumber(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x12, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

CString CCrystalCtrl::GetLastErrorString()

{

CString result;

InvokeHelper(0x13, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetLastErrorString(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x13, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

CString CCrystalCtrl::GetConnect()

{

CString result;

InvokeHelper(0x14, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetConnect(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x14, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

long CCrystalCtrl::GetSessionHandle()

{

long result;

InvokeHelper(0x15, DISPATCH\_PROPERTYGET, VT\_I4, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetSessionHandle(long nNewValue)

{

static BYTE parms[] =

VTS\_I4;

InvokeHelper(0x15, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

CString CCrystalCtrl::GetUserName\_()

{

CString result;

InvokeHelper(0x16, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetUserName(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x16, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

CString CCrystalCtrl::GetPassword()

{

CString result;

InvokeHelper(0x17, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPassword(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x17, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

long CCrystalCtrl::GetDestination()

{

long result;

InvokeHelper(0x18, DISPATCH\_PROPERTYGET, VT\_I4, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetDestination(long nNewValue)

{

static BYTE parms[] =

VTS\_I4;

InvokeHelper(0x18, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

long CCrystalCtrl::GetPrintFileType()

{

long result;

InvokeHelper(0x19, DISPATCH\_PROPERTYGET, VT\_I4, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrintFileType(long nNewValue)

{

static BYTE parms[] =

VTS\_I4;

InvokeHelper(0x19, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

long CCrystalCtrl::GetWindowBorderStyle()

{

long result;

InvokeHelper(0x1a, DISPATCH\_PROPERTYGET, VT\_I4, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowBorderStyle(long nNewValue)

{

static BYTE parms[] =

VTS\_I4;

InvokeHelper(0x1a, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

CString CCrystalCtrl::GetBoundReportHeading()

{

CString result;

InvokeHelper(0x1, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetBoundReportHeading(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x1, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

BOOL CCrystalCtrl::GetBoundReportFooter()

{

BOOL result;

InvokeHelper(0x2, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetBoundReportFooter(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x2, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

long CCrystalCtrl::GetReportSource()

{

long result;

InvokeHelper(0x1b, DISPATCH\_PROPERTYGET, VT\_I4, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetReportSource(long nNewValue)

{

static BYTE parms[] =

VTS\_I4;

InvokeHelper(0x1b, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

short CCrystalCtrl::GetDetailCopies()

{

short result;

InvokeHelper(0x21, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetDetailCopies(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x21, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

BOOL CCrystalCtrl::GetDiscardSavedData()

{

BOOL result;

InvokeHelper(0x22, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetDiscardSavedData(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x22, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

CString CCrystalCtrl::GetEMailCCList()

{

CString result;

InvokeHelper(0x23, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetEMailCCList(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x23, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

CString CCrystalCtrl::GetEMailMessage()

{

CString result;

InvokeHelper(0x24, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetEMailMessage(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x24, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

CString CCrystalCtrl::GetEMailSubject()

{

CString result;

InvokeHelper(0x25, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetEMailSubject(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x25, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

CString CCrystalCtrl::GetEMailToList()

{

CString result;

InvokeHelper(0x26, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetEMailToList(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x26, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

short CCrystalCtrl::GetMarginBottom()

{

short result;

InvokeHelper(0x28, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetMarginBottom(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x28, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

short CCrystalCtrl::GetMarginLeft()

{

short result;

InvokeHelper(0x29, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetMarginLeft(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x29, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

short CCrystalCtrl::GetMarginRight()

{

short result;

InvokeHelper(0x2a, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetMarginRight(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x2a, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

short CCrystalCtrl::GetMarginTop()

{

short result;

InvokeHelper(0x2b, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetMarginTop(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x2b, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

short CCrystalCtrl::GetPrintDay()

{

short result;

InvokeHelper(0x2c, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrintDay(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x2c, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

long CCrystalCtrl::GetPrinterCollation()

{

long result;

InvokeHelper(0x2d, DISPATCH\_PROPERTYGET, VT\_I4, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrinterCollation(long nNewValue)

{

static BYTE parms[] =

VTS\_I4;

InvokeHelper(0x2d, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

short CCrystalCtrl::GetPrinterCopies()

{

short result;

InvokeHelper(0x2e, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrinterCopies(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x2e, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

CString CCrystalCtrl::GetPrinterDriver()

{

CString result;

InvokeHelper(0x2f, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrinterDriver(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x2f, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

CString CCrystalCtrl::GetPrinterName()

{

CString result;

InvokeHelper(0x30, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrinterName(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x30, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

CString CCrystalCtrl::GetPrinterPort()

{

CString result;

InvokeHelper(0x31, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrinterPort(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x31, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

short CCrystalCtrl::GetPrinterStartPage()

{

short result;

InvokeHelper(0x32, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrinterStartPage(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x32, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

short CCrystalCtrl::GetPrinterStopPage()

{

short result;

InvokeHelper(0x33, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrinterStopPage(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x33, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

CString CCrystalCtrl::GetPrintFileCharSepQuote()

{

CString result;

InvokeHelper(0x34, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrintFileCharSepQuote(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x34, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

CString CCrystalCtrl::GetPrintFileCharSepSeparator()

{

CString result;

InvokeHelper(0x35, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrintFileCharSepSeparator(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x35, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

BOOL CCrystalCtrl::GetPrintFileUseRptDateFmt()

{

BOOL result;

InvokeHelper(0x36, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrintFileUseRptDateFmt(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x36, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCrystalCtrl::GetPrintFileUseRptNumberFmt()

{

BOOL result;

InvokeHelper(0x37, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrintFileUseRptNumberFmt(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x37, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

short CCrystalCtrl::GetPrintMonth()

{

short result;

InvokeHelper(0x38, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrintMonth(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x38, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

short CCrystalCtrl::GetPrintYear()

{

short result;

InvokeHelper(0x39, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrintYear(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x39, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

long CCrystalCtrl::GetRecordsPrinted()

{

long result;

InvokeHelper(0x3a, DISPATCH\_PROPERTYGET, VT\_I4, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetRecordsPrinted(long nNewValue)

{

static BYTE parms[] =

VTS\_I4;

InvokeHelper(0x3a, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

long CCrystalCtrl::GetRecordsRead()

{

long result;

InvokeHelper(0x3b, DISPATCH\_PROPERTYGET, VT\_I4, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetRecordsRead(long nNewValue)

{

static BYTE parms[] =

VTS\_I4;

InvokeHelper(0x3b, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

long CCrystalCtrl::GetRecordsSelected()

{

long result;

InvokeHelper(0x3c, DISPATCH\_PROPERTYGET, VT\_I4, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetRecordsSelected(long nNewValue)

{

static BYTE parms[] =

VTS\_I4;

InvokeHelper(0x3c, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

short CCrystalCtrl::GetReportDisplayPage()

{

short result;

InvokeHelper(0x3d, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetReportDisplayPage(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x3d, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

short CCrystalCtrl::GetReportLatestPage()

{

short result;

InvokeHelper(0x3e, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetReportLatestPage(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x3e, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

short CCrystalCtrl::GetReportStartPage()

{

short result;

InvokeHelper(0x3f, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetReportStartPage(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x3f, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

CString CCrystalCtrl::GetSQLQuery()

{

CString result;

InvokeHelper(0x40, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetSQLQuery(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x40, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

short CCrystalCtrl::GetStatus()

{

short result;

InvokeHelper(0x41, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetStatus(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x41, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

BOOL CCrystalCtrl::GetWindowControls()

{

BOOL result;

InvokeHelper(0x42, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowControls(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x42, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

long CCrystalCtrl::GetWindowState()

{

long result;

InvokeHelper(0x43, DISPATCH\_PROPERTYGET, VT\_I4, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowState(long nNewValue)

{

static BYTE parms[] =

VTS\_I4;

InvokeHelper(0x43, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

long CCrystalCtrl::GetDialogParentHandle()

{

long result;

InvokeHelper(0x4f, DISPATCH\_PROPERTYGET, VT\_I4, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetDialogParentHandle(long nNewValue)

{

static BYTE parms[] =

VTS\_I4;

InvokeHelper(0x4f, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

BOOL CCrystalCtrl::GetProgressDialog()

{

BOOL result;

InvokeHelper(0x50, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetProgressDialog(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x50, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

CString CCrystalCtrl::GetExchangeProfile()

{

CString result;

InvokeHelper(0x5b, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetExchangeProfile(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x5b, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

CString CCrystalCtrl::GetExchangePassword()

{

CString result;

InvokeHelper(0x5c, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetExchangePassword(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x5c, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

CString CCrystalCtrl::GetExchangeFolder()

{

CString result;

InvokeHelper(0x5d, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetExchangeFolder(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x5d, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

CString CCrystalCtrl::GetPrintFileODBCSource()

{

CString result;

InvokeHelper(0x5e, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrintFileODBCSource(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x5e, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

CString CCrystalCtrl::GetPrintFileODBCUser()

{

CString result;

InvokeHelper(0x5f, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrintFileODBCUser(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x5f, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

CString CCrystalCtrl::GetPrintFileODBCPassword()

{

CString result;

InvokeHelper(0x60, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrintFileODBCPassword(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x60, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

CString CCrystalCtrl::GetPrintFileODBCTable()

{

CString result;

InvokeHelper(0x61, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrintFileODBCTable(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x61, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

short CCrystalCtrl::GetPrintFileLinesPerPage()

{

short result;

InvokeHelper(0x62, DISPATCH\_PROPERTYGET, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetPrintFileLinesPerPage(short nNewValue)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x62, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

CString CCrystalCtrl::GetSubreportToChange()

{

CString result;

InvokeHelper(0x6c, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetSubreportToChange(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x6c, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

CString CCrystalCtrl::GetReportTitle()

{

CString result;

InvokeHelper(0x6f, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetReportTitle(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x6f, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

CString CCrystalCtrl::GetGridSource()

{

CString result;

InvokeHelper(0x72, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetGridSource(LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x72, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

lpszNewValue);

}

BOOL CCrystalCtrl::GetWindowShowGroupTree()

{

BOOL result;

InvokeHelper(0x73, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowShowGroupTree(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x73, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCrystalCtrl::GetWindowAllowDrillDown()

{

BOOL result;

InvokeHelper(0x74, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowAllowDrillDown(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x74, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCrystalCtrl::GetWindowShowNavigationCtls()

{

BOOL result;

InvokeHelper(0x75, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowShowNavigationCtls(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x75, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCrystalCtrl::GetWindowShowCancelBtn()

{

BOOL result;

InvokeHelper(0x76, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowShowCancelBtn(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x76, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCrystalCtrl::GetWindowShowPrintBtn()

{

BOOL result;

InvokeHelper(0x77, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowShowPrintBtn(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x77, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCrystalCtrl::GetWindowShowExportBtn()

{

BOOL result;

InvokeHelper(0x78, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowShowExportBtn(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x78, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCrystalCtrl::GetWindowShowZoomCtl()

{

BOOL result;

InvokeHelper(0x79, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowShowZoomCtl(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x79, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCrystalCtrl::GetWindowShowCloseBtn()

{

BOOL result;

InvokeHelper(0x7a, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowShowCloseBtn(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x7a, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCrystalCtrl::GetWindowShowProgressCtls()

{

BOOL result;

InvokeHelper(0x7b, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowShowProgressCtls(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x7b, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCrystalCtrl::GetWindowShowSearchBtn()

{

BOOL result;

InvokeHelper(0x7c, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowShowSearchBtn(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x7c, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCrystalCtrl::GetWindowShowPrintSetupBtn()

{

BOOL result;

InvokeHelper(0x7d, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowShowPrintSetupBtn(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x7d, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCrystalCtrl::GetWindowShowRefreshBtn()

{

BOOL result;

InvokeHelper(0x7e, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::SetWindowShowRefreshBtn(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x7e, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

CRowCursor CCrystalCtrl::GetDataSource()

{

LPDISPATCH pDispatch;

InvokeHelper(0x100, DISPATCH\_PROPERTYGET, VT\_DISPATCH, (void\*)&pDispatch, NULL);

return CRowCursor(pDispatch);

}

void CCrystalCtrl::SetDataSource(LPDISPATCH newValue)

{

static BYTE parms[] =

VTS\_DISPATCH;

InvokeHelper(0x100, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

newValue);

}

CString CCrystalCtrl::GetFormulas(short index)

{

CString result;

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x1d, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, parms,

index);

return result;

}

void CCrystalCtrl::SetFormulas(short index, LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_I2 VTS\_BSTR;

InvokeHelper(0x1d, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

index, lpszNewValue);

}

CString CCrystalCtrl::GetSortFields(short index)

{

CString result;

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x1e, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, parms,

index);

return result;

}

void CCrystalCtrl::SetSortFields(short index, LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_I2 VTS\_BSTR;

InvokeHelper(0x1e, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

index, lpszNewValue);

}

CString CCrystalCtrl::GetGroupSortFields(short index)

{

CString result;

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x1f, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, parms,

index);

return result;

}

void CCrystalCtrl::SetGroupSortFields(short index, LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_I2 VTS\_BSTR;

InvokeHelper(0x1f, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

index, lpszNewValue);

}

CString CCrystalCtrl::GetDataFiles(short index)

{

CString result;

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x20, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, parms,

index);

return result;

}

void CCrystalCtrl::SetDataFiles(short index, LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_I2 VTS\_BSTR;

InvokeHelper(0x20, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

index, lpszNewValue);

}

CString CCrystalCtrl::GetGroupCondition(short index)

{

CString result;

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x44, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, parms,

index);

return result;

}

void CCrystalCtrl::SetGroupCondition(short index, LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_I2 VTS\_BSTR;

InvokeHelper(0x44, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

index, lpszNewValue);

}

CString CCrystalCtrl::GetSectionMinHeight(short index)

{

CString result;

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x45, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, parms,

index);

return result;

}

void CCrystalCtrl::SetSectionFormat(short index, LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_I2 VTS\_BSTR;

InvokeHelper(0x46, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

index, lpszNewValue);

}

CString CCrystalCtrl::GetSectionLineHeight(short index)

{

CString result;

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x47, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, parms,

index);

return result;

}

void CCrystalCtrl::SetSectionLineHeight(short index, LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_I2 VTS\_BSTR;

InvokeHelper(0x47, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

index, lpszNewValue);

}

CString CCrystalCtrl::GetSectionFont(short index)

{

CString result;

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x48, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, parms,

index);

return result;

}

void CCrystalCtrl::SetSectionFont(short index, LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_I2 VTS\_BSTR;

InvokeHelper(0x48, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

index, lpszNewValue);

}

CString CCrystalCtrl::GetStoredProcParam(short index)

{

CString result;

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x49, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, parms,

index);

return result;

}

void CCrystalCtrl::SetStoredProcParam(short index, LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_I2 VTS\_BSTR;

InvokeHelper(0x49, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

index, lpszNewValue);

}

CString CCrystalCtrl::GetGraphType(short index)

{

CString result;

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x4a, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, parms,

index);

return result;

}

void CCrystalCtrl::SetGraphType(short index, LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_I2 VTS\_BSTR;

InvokeHelper(0x4a, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

index, lpszNewValue);

}

CString CCrystalCtrl::GetGraphData(short index)

{

CString result;

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x4b, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, parms,

index);

return result;

}

void CCrystalCtrl::SetGraphData(short index, LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_I2 VTS\_BSTR;

InvokeHelper(0x4b, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

index, lpszNewValue);

}

CString CCrystalCtrl::GetGraphText(short index)

{

CString result;

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x4c, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, parms,

index);

return result;

}

void CCrystalCtrl::SetGraphText(short index, LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_I2 VTS\_BSTR;

InvokeHelper(0x4c, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

index, lpszNewValue);

}

CString CCrystalCtrl::GetGraphOptions(short index)

{

CString result;

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x4d, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, parms,

index);

return result;

}

void CCrystalCtrl::SetGraphOptions(short index, LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_I2 VTS\_BSTR;

InvokeHelper(0x4d, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

index, lpszNewValue);

}

CString CCrystalCtrl::GetParameterFields(short index)

{

CString result;

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x51, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, parms,

index);

return result;

}

void CCrystalCtrl::SetParameterFields(short index, LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_I2 VTS\_BSTR;

InvokeHelper(0x51, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

index, lpszNewValue);

}

CString CCrystalCtrl::GetLogonInfo(short index)

{

CString result;

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x63, DISPATCH\_PROPERTYGET, VT\_BSTR, (void\*)&result, parms,

index);

return result;

}

void CCrystalCtrl::SetLogonInfo(short index, LPCTSTR lpszNewValue)

{

static BYTE parms[] =

VTS\_I2 VTS\_BSTR;

InvokeHelper(0x63, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

index, lpszNewValue);

}

void CCrystalCtrl::AboutBox()

{

InvokeHelper(0xfffffdd8, DISPATCH\_METHOD, VT\_EMPTY, NULL, NULL);

}

short CCrystalCtrl::PrintReport()

{

short result;

InvokeHelper(0x1c, DISPATCH\_METHOD, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::PageFirst()

{

InvokeHelper(0x52, DISPATCH\_METHOD, VT\_EMPTY, NULL, NULL);

}

void CCrystalCtrl::PageNext()

{

InvokeHelper(0x53, DISPATCH\_METHOD, VT\_EMPTY, NULL, NULL);

}

void CCrystalCtrl::PageLast()

{

InvokeHelper(0x54, DISPATCH\_METHOD, VT\_EMPTY, NULL, NULL);

}

void CCrystalCtrl::PagePrevious()

{

InvokeHelper(0x55, DISPATCH\_METHOD, VT\_EMPTY, NULL, NULL);

}

short CCrystalCtrl::PageCount()

{

short result;

InvokeHelper(0x56, DISPATCH\_METHOD, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::PageShow(short p)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x57, DISPATCH\_METHOD, VT\_EMPTY, NULL, parms,

p);

}

void CCrystalCtrl::PageZoom(short z)

{

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x58, DISPATCH\_METHOD, VT\_EMPTY, NULL, parms,

z);

}

void CCrystalCtrl::PageZoomNext()

{

InvokeHelper(0x59, DISPATCH\_METHOD, VT\_EMPTY, NULL, NULL);

}

void CCrystalCtrl::PrinterSelect()

{

InvokeHelper(0x5a, DISPATCH\_METHOD, VT\_EMPTY, NULL, NULL);

}

CString CCrystalCtrl::FetchSelectionFormula()

{

CString result;

InvokeHelper(0x64, DISPATCH\_METHOD, VT\_BSTR, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::ReplaceSelectionFormula(LPCTSTR s)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0x65, DISPATCH\_METHOD, VT\_EMPTY, NULL, parms,

s);

}

short CCrystalCtrl::RetrieveDataFiles()

{

short result;

InvokeHelper(0x66, DISPATCH\_METHOD, VT\_I2, (void\*)&result, NULL);

return result;

}

short CCrystalCtrl::RetrieveLogonInfo()

{

short result;

InvokeHelper(0x67, DISPATCH\_METHOD, VT\_I2, (void\*)&result, NULL);

return result;

}

void CCrystalCtrl::RetrieveSQLQuery()

{

InvokeHelper(0x68, DISPATCH\_METHOD, VT\_EMPTY, NULL, NULL);

}

short CCrystalCtrl::RetrieveStoredProcParams()

{

short result;

InvokeHelper(0x69, DISPATCH\_METHOD, VT\_I2, (void\*)&result, NULL);

return result;

}

short CCrystalCtrl::LogOnServer(LPCTSTR dll, LPCTSTR server, LPCTSTR database, LPCTSTR userid, LPCTSTR Password)

{

short result;

static BYTE parms[] =

VTS\_BSTR VTS\_BSTR VTS\_BSTR VTS\_BSTR VTS\_BSTR;

InvokeHelper(0x6a, DISPATCH\_METHOD, VT\_I2, (void\*)&result, parms,

dll, server, database, userid, Password);

return result;

}

void CCrystalCtrl::LogOffServer(short id, BOOL all)

{

static BYTE parms[] =

VTS\_I2 VTS\_BOOL;

InvokeHelper(0x6b, DISPATCH\_METHOD, VT\_EMPTY, NULL, parms,

id, all);

}

short CCrystalCtrl::GetNSubreports()

{

short result;

InvokeHelper(0x6d, DISPATCH\_METHOD, VT\_I2, (void\*)&result, NULL);

return result;

}

CString CCrystalCtrl::GetNthSubreportName(short n)

{

CString result;

static BYTE parms[] =

VTS\_I2;

InvokeHelper(0x6e, DISPATCH\_METHOD, VT\_BSTR, (void\*)&result, parms,

n);

return result;

}

void CCrystalCtrl::Reset()

{

InvokeHelper(0x70, DISPATCH\_METHOD, VT\_EMPTY, NULL, NULL);

}

void CCrystalCtrl::SpecifyDataSourceField(short n, LPCTSTR name, short width)

{

static BYTE parms[] =

VTS\_I2 VTS\_BSTR VTS\_I2;

InvokeHelper(0x71, DISPATCH\_METHOD, VT\_EMPTY, NULL, parms,

n, name, width);

}

void CCrystalCtrl::SetTablePrivateData(short TableIndex, long DataTag, const VARIANT& Data)

{

static BYTE parms[] =

VTS\_I2 VTS\_I4 VTS\_VARIANT;

InvokeHelper(0x7f, DISPATCH\_METHOD, VT\_EMPTY, NULL, parms,

TableIndex, DataTag, &Data);

}

// Machine generated IDispatch wrapper class(es) created by Microsoft Visual C++

// NOTE: Do not modify the contents of this file. If this class is regenerated by

// Microsoft Visual C++, your modifications will be overwritten.

#include "stdafx.h"

#include "crdesignerctrl.h"

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

// CCRDesignerCtrl

IMPLEMENT\_DYNCREATE(CCRDesignerCtrl, CWnd)

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

// CCRDesignerCtrl properties

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

// CCRDesignerCtrl operations

BOOL CCRDesignerCtrl::GetDisplayFieldView()

{

BOOL result;

InvokeHelper(0x1, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCRDesignerCtrl::SetDisplayFieldView(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x1, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCRDesignerCtrl::GetDisplayToolbar()

{

BOOL result;

InvokeHelper(0x2, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCRDesignerCtrl::SetDisplayToolbar(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x2, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCRDesignerCtrl::GetUseIndexForSpeed()

{

BOOL result;

InvokeHelper(0x3, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCRDesignerCtrl::SetUseIndexForSpeed(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x3, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCRDesignerCtrl::GetTranslateDosStrings()

{

BOOL result;

InvokeHelper(0x4, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCRDesignerCtrl::SetTranslateDosStrings(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x4, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCRDesignerCtrl::GetTranslateDosMemos()

{

BOOL result;

InvokeHelper(0x5, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCRDesignerCtrl::SetTranslateDosMemos(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x5, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

long CCRDesignerCtrl::GetConvertDateTimeType()

{

long result;

InvokeHelper(0x6, DISPATCH\_PROPERTYGET, VT\_I4, (void\*)&result, NULL);

return result;

}

void CCRDesignerCtrl::SetConvertDateTimeType(long nNewValue)

{

static BYTE parms[] =

VTS\_I4;

InvokeHelper(0x6, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

nNewValue);

}

BOOL CCRDesignerCtrl::GetConvertNullFieldToDefault()

{

BOOL result;

InvokeHelper(0x7, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCRDesignerCtrl::SetConvertNullFieldToDefault(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x7, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCRDesignerCtrl::GetCaseInsensitiveSQLData()

{

BOOL result;

InvokeHelper(0x8, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCRDesignerCtrl::SetCaseInsensitiveSQLData(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x8, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCRDesignerCtrl::GetVerifyOnEveryPrint()

{

BOOL result;

InvokeHelper(0x9, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCRDesignerCtrl::SetVerifyOnEveryPrint(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x9, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCRDesignerCtrl::GetDisplayRulers()

{

BOOL result;

InvokeHelper(0xa, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCRDesignerCtrl::SetDisplayRulers(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0xa, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCRDesignerCtrl::GetEnableSnapToGrid()

{

BOOL result;

InvokeHelper(0xc, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCRDesignerCtrl::SetEnableSnapToGrid(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0xc, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

BOOL CCRDesignerCtrl::GetDisplayGrid()

{

BOOL result;

InvokeHelper(0xd, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCRDesignerCtrl::SetDisplayGrid(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0xd, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

float CCRDesignerCtrl::GetGridSize()

{

float result;

InvokeHelper(0xe, DISPATCH\_PROPERTYGET, VT\_R4, (void\*)&result, NULL);

return result;

}

void CCRDesignerCtrl::SetGridSize(float newValue)

{

static BYTE parms[] =

VTS\_R4;

InvokeHelper(0xe, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

newValue);

}

void CCRDesignerCtrl::SaveReport(LPCTSTR reportName)

{

static BYTE parms[] =

VTS\_BSTR;

InvokeHelper(0xf, DISPATCH\_METHOD, VT\_EMPTY, NULL, parms,

reportName);

}

BOOL CCRDesignerCtrl::GetDisplayHiddenSections()

{

BOOL result;

InvokeHelper(0xb, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCRDesignerCtrl::SetDisplayHiddenSections(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0xb, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

LPUNKNOWN CCRDesignerCtrl::GetReportObject()

{

LPUNKNOWN result;

InvokeHelper(0x11, DISPATCH\_PROPERTYGET, VT\_UNKNOWN, (void\*)&result, NULL);

return result;

}

void CCRDesignerCtrl::SetReportObject(LPUNKNOWN newValue)

{

static BYTE parms[] =

VTS\_UNKNOWN;

InvokeHelper(0x11, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

newValue);

}

BOOL CCRDesignerCtrl::GetEnableHelp()

{

BOOL result;

InvokeHelper(0x12, DISPATCH\_PROPERTYGET, VT\_BOOL, (void\*)&result, NULL);

return result;

}

void CCRDesignerCtrl::SetEnableHelp(BOOL bNewValue)

{

static BYTE parms[] =

VTS\_BOOL;

InvokeHelper(0x12, DISPATCH\_PROPERTYPUT, VT\_EMPTY, NULL, parms,

bNewValue);

}

#if !defined(AFX\_CRDESIGNERCTRL\_H\_\_6D8B8A38\_4B76\_48C4\_BEEF\_C505DC0ACB66\_\_INCLUDED\_)

#define AFX\_CRDESIGNERCTRL\_H\_\_6D8B8A38\_4B76\_48C4\_BEEF\_C505DC0ACB66\_\_INCLUDED\_

#if \_MSC\_VER > 1000

#pragma once

#endif // \_MSC\_VER > 1000

// Machine generated IDispatch wrapper class(es) created by Microsoft Visual C++

// NOTE: Do not modify the contents of this file. If this class is regenerated by

// Microsoft Visual C++, your modifications will be overwritten.

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

// CCRDesignerCtrl wrapper class

class CCRDesignerCtrl : public CWnd

{

protected:

DECLARE\_DYNCREATE(CCRDesignerCtrl)

public:

CLSID const& GetClsid()

{

static CLSID const clsid

= { 0x2a3a1c00, 0xf9db, 0x11d3, { 0x80, 0x8e, 0x0, 0xa0, 0xc9, 0xda, 0xc8, 0x3f } };

return clsid;

}

virtual BOOL Create(LPCTSTR lpszClassName,

LPCTSTR lpszWindowName, DWORD dwStyle,

const RECT& rect,

CWnd\* pParentWnd, UINT nID,

CCreateContext\* pContext = NULL)

{ return CreateControl(GetClsid(), lpszWindowName, dwStyle, rect, pParentWnd, nID); }

BOOL Create(LPCTSTR lpszWindowName, DWORD dwStyle,

const RECT& rect, CWnd\* pParentWnd, UINT nID,

CFile\* pPersist = NULL, BOOL bStorage = FALSE,

BSTR bstrLicKey = NULL)

{ return CreateControl(GetClsid(), lpszWindowName, dwStyle, rect, pParentWnd, nID,

pPersist, bStorage, bstrLicKey); }

// Attributes

public:

// Operations

public:

BOOL GetDisplayFieldView();

void SetDisplayFieldView(BOOL bNewValue);

BOOL GetDisplayToolbar();

void SetDisplayToolbar(BOOL bNewValue);

BOOL GetUseIndexForSpeed();

void SetUseIndexForSpeed(BOOL bNewValue);

BOOL GetTranslateDosStrings();

void SetTranslateDosStrings(BOOL bNewValue);

BOOL GetTranslateDosMemos();

void SetTranslateDosMemos(BOOL bNewValue);

long GetConvertDateTimeType();

void SetConvertDateTimeType(long nNewValue);

BOOL GetConvertNullFieldToDefault();

void SetConvertNullFieldToDefault(BOOL bNewValue);

BOOL GetCaseInsensitiveSQLData();

void SetCaseInsensitiveSQLData(BOOL bNewValue);

BOOL GetVerifyOnEveryPrint();

void SetVerifyOnEveryPrint(BOOL bNewValue);

BOOL GetDisplayRulers();

void SetDisplayRulers(BOOL bNewValue);

BOOL GetEnableSnapToGrid();

void SetEnableSnapToGrid(BOOL bNewValue);

BOOL GetDisplayGrid();

void SetDisplayGrid(BOOL bNewValue);

float GetGridSize();

void SetGridSize(float newValue);

void SaveReport(LPCTSTR reportName);

BOOL GetDisplayHiddenSections();

void SetDisplayHiddenSections(BOOL bNewValue);

LPUNKNOWN GetReportObject();

void SetReportObject(LPUNKNOWN newValue);

BOOL GetEnableHelp();

void SetEnableHelp(BOOL bNewValue);

};

//{{AFX\_INSERT\_LOCATION}}

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

#endif

#if !defined(AFX\_CRYSTALREPORTVIEWER4\_H\_\_E9B14F18\_09B0\_4EC9\_A883\_815C5409A7EF\_\_INCLUDED\_)

#define AFX\_CRYSTALREPORTVIEWER4\_H\_\_E9B14F18\_09B0\_4EC9\_A883\_815C5409A7EF\_\_INCLUDED\_

#if \_MSC\_VER > 1000

#pragma once

#endif // \_MSC\_VER > 1000

// Machine generated IDispatch wrapper class(es) created by Microsoft Visual C++

// NOTE: Do not modify the contents of this file. If this class is regenerated by

// Microsoft Visual C++, your modifications will be overwritten.

// Dispatch interfaces referenced by this interface

class CCRVTrackCursorInfo;

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

// CCrystalReportViewer4 wrapper class

class CCrystalReportViewer4 : public CWnd

{

protected:

DECLARE\_DYNCREATE(CCrystalReportViewer4)

public:

CLSID const& GetClsid()

{

static CLSID const clsid

= { 0xc4847596, 0x972c, 0x11d0, { 0x95, 0x67, 0x0, 0xa0, 0xc9, 0x27, 0x3c, 0x2a } };

return clsid;

}

virtual BOOL Create(LPCTSTR lpszClassName,

LPCTSTR lpszWindowName, DWORD dwStyle,

const RECT& rect,

CWnd\* pParentWnd, UINT nID,

CCreateContext\* pContext = NULL)

{ return CreateControl(GetClsid(), lpszWindowName, dwStyle, rect, pParentWnd, nID); }

BOOL Create(LPCTSTR lpszWindowName, DWORD dwStyle,

const RECT& rect, CWnd\* pParentWnd, UINT nID,

CFile\* pPersist = NULL, BOOL bStorage = FALSE,

BSTR bstrLicKey = NULL)

{ return CreateControl(GetClsid(), lpszWindowName, dwStyle, rect, pParentWnd, nID,

pPersist, bStorage, bstrLicKey); }

// Attributes

public:

// Operations

public:

LPUNKNOWN GetReportSource();

void SetReportSource(LPUNKNOWN newValue);

BOOL GetDisplayGroupTree();

void SetDisplayGroupTree(BOOL bNewValue);

BOOL GetDisplayToolbar();

void SetDisplayToolbar(BOOL bNewValue);

BOOL GetEnableGroupTree();

void SetEnableGroupTree(BOOL bNewValue);

BOOL GetEnableNavigationControls();

void SetEnableNavigationControls(BOOL bNewValue);

BOOL GetEnableStopButton();

void SetEnableStopButton(BOOL bNewValue);

BOOL GetEnablePrintButton();

void SetEnablePrintButton(BOOL bNewValue);

BOOL GetEnableZoomControl();

void SetEnableZoomControl(BOOL bNewValue);

BOOL GetEnableCloseButton();

void SetEnableCloseButton(BOOL bNewValue);

BOOL GetEnableProgressControl();

void SetEnableProgressControl(BOOL bNewValue);

BOOL GetEnableSearchControl();

void SetEnableSearchControl(BOOL bNewValue);

BOOL GetEnableRefreshButton();

void SetEnableRefreshButton(BOOL bNewValue);

BOOL GetEnableDrillDown();

void SetEnableDrillDown(BOOL bNewValue);

BOOL GetEnableAnimationCtrl();

void SetEnableAnimationCtrl(BOOL bNewValue);

BOOL GetEnableSelectExpertButton();

void SetEnableSelectExpertButton(BOOL bNewValue);

void ViewReport();

BOOL GetEnableToolbar();

void SetEnableToolbar(BOOL bNewValue);

BOOL GetDisplayBorder();

void SetDisplayBorder(BOOL bNewValue);

BOOL GetDisplayTabs();

void SetDisplayTabs(BOOL bNewValue);

BOOL GetDisplayBackgroundEdge();

void SetDisplayBackgroundEdge(BOOL bNewValue);

CCRVTrackCursorInfo GetTrackCursorInfo();

short GetActiveViewIndex();

short GetViewCount();

void ActivateView(const VARIANT& Index);

void AddView(const VARIANT& GroupPath);

void CloseView(const VARIANT& Index);

VARIANT GetViewPath(short Index);

void PrintReport();

void Refresh();

void SearchForText(LPCTSTR Text);

void ShowFirstPage();

void ShowNextPage();

void ShowPreviousPage();

void ShowLastPage();

void ShowNthPage(short PageNumber);

void Zoom(short ZoomLevel);

long GetCurrentPageNumber();

void ShowGroup(const VARIANT& GroupPath);

BOOL GetIsBusy();

BOOL GetEnablePopupMenu();

void SetEnablePopupMenu(BOOL bNewValue);

BOOL GetEnableExportButton();

void SetEnableExportButton(BOOL bNewValue);

BOOL GetEnableSearchExpertButton();

void SetEnableSearchExpertButton(BOOL bNewValue);

void SearchByFormula(LPCTSTR formula);

CString GetViewName(BSTR\* pTabName);

BOOL GetEnableHelpButton();

void SetEnableHelpButton(BOOL bNewValue);

VARIANT GetGroup();

void GetLastPageNumber(long\* pageN, BOOL\* lastPageKnown);

void RefreshEx(BOOL refreshServerData);

};

//{{AFX\_INSERT\_LOCATION}}

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

#endif

// DesignerViewerDlg.h : header file

//

//{{AFX\_INCLUDES()

#include "crdesignerctrl.h"

#include "crystalreportviewer4.h"

//}}AFX\_INCLUDES

#if !defined(AFX\_DESIGNERVIEWERDLG\_H\_\_2DDE3256\_5266\_4608\_9BA5\_848E28988028\_\_INCLUDED\_)

#define AFX\_DESIGNERVIEWERDLG\_H\_\_2DDE3256\_5266\_4608\_9BA5\_848E28988028\_\_INCLUDED\_

#if \_MSC\_VER > 1000

#pragma once

#endif // \_MSC\_VER > 1000

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

// CDesignerViewerDlg dialog

class CDesignerViewerDlg : public CDialog

{

// Construction

public:

IApplicationPtr m\_Application;

IReportPtr m\_Report ;

public:

void InitReport(BOOL bNewOrOpen);

public:

CDesignerViewerDlg(CWnd\* pParent = NULL); // standard constructor

// Dialog Data

//{{AFX\_DATA(CDesignerViewerDlg)

enum { IDD = IDD\_DESIGNERVIEWER\_DIALOG };

CButton m\_DesignerReportBtn;

CButton m\_PreviewReportBtn;

CButton m\_SaveReportBtn;

CButton m\_OpenReportBtn;

CButton m\_NewReportBtn;

CCRDesignerCtrl m\_ReportDesigner;

CCrystalReportViewer4 m\_ReportViewer;

//}}AFX\_DATA

// ClassWizard generated virtual function overrides

//{{AFX\_VIRTUAL(CDesignerViewerDlg)

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV support

//}}AFX\_VIRTUAL

// Implementation

protected:

HICON m\_hIcon;

// Generated message map functions

//{{AFX\_MSG(CDesignerViewerDlg)

virtual BOOL OnInitDialog();

afx\_msg void OnSysCommand(UINT nID, LPARAM lParam);

afx\_msg void OnPaint();

afx\_msg HCURSOR OnQueryDragIcon();

afx\_msg void OnBtnnewreport();

afx\_msg void OnBtnopenreport();

afx\_msg void OnBtnsavereport();

afx\_msg void OnBtnpreviewreport();

afx\_msg void OnBtndesignreport();

virtual void OnCancel();

//}}AFX\_MSG

DECLARE\_MESSAGE\_MAP()

};

//{{AFX\_INSERT\_LOCATION}}

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

#endif

#if !defined(AFX\_CRYSTALREPORTVIEWER4\_H\_\_E9B14F18\_09B0\_4EC9\_A883\_815C5409A7EF\_\_INCLUDED\_)

#define AFX\_CRYSTALREPORTVIEWER4\_H\_\_E9B14F18\_09B0\_4EC9\_A883\_815C5409A7EF\_\_INCLUDED\_

#if \_MSC\_VER > 1000

#pragma once

#endif // \_MSC\_VER > 1000

// Machine generated IDispatch wrapper class(es) created by Microsoft Visual C++

// NOTE: Do not modify the contents of this file. If this class is regenerated by

// Microsoft Visual C++, your modifications will be overwritten.

// Dispatch interfaces referenced by this interface

class CCRVTrackCursorInfo;

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

// CCrystalReportViewer4 wrapper class

class CCrystalReportViewer4 : public CWnd

{

protected:

DECLARE\_DYNCREATE(CCrystalReportViewer4)

public:

CLSID const& GetClsid()

{

static CLSID const clsid

= { 0xc4847596, 0x972c, 0x11d0, { 0x95, 0x67, 0x0, 0xa0, 0xc9, 0x27, 0x3c, 0x2a } };

return clsid;

}

virtual BOOL Create(LPCTSTR lpszClassName,

LPCTSTR lpszWindowName, DWORD dwStyle,

const RECT& rect,

CWnd\* pParentWnd, UINT nID,

CCreateContext\* pContext = NULL)

{ return CreateControl(GetClsid(), lpszWindowName, dwStyle, rect, pParentWnd, nID); }

BOOL Create(LPCTSTR lpszWindowName, DWORD dwStyle,

const RECT& rect, CWnd\* pParentWnd, UINT nID,

CFile\* pPersist = NULL, BOOL bStorage = FALSE,

BSTR bstrLicKey = NULL)

{ return CreateControl(GetClsid(), lpszWindowName, dwStyle, rect, pParentWnd, nID,

pPersist, bStorage, bstrLicKey); }

// Attributes

public:

// Operations

public:

LPUNKNOWN GetReportSource();

void SetReportSource(LPUNKNOWN newValue);

BOOL GetDisplayGroupTree();

void SetDisplayGroupTree(BOOL bNewValue);

BOOL GetDisplayToolbar();

void SetDisplayToolbar(BOOL bNewValue);

BOOL GetEnableGroupTree();

void SetEnableGroupTree(BOOL bNewValue);

BOOL GetEnableNavigationControls();

void SetEnableNavigationControls(BOOL bNewValue);

BOOL GetEnableStopButton();

void SetEnableStopButton(BOOL bNewValue);

BOOL GetEnablePrintButton();

void SetEnablePrintButton(BOOL bNewValue);

BOOL GetEnableZoomControl();

void SetEnableZoomControl(BOOL bNewValue);

BOOL GetEnableCloseButton();

void SetEnableCloseButton(BOOL bNewValue);

BOOL GetEnableProgressControl();

void SetEnableProgressControl(BOOL bNewValue);

BOOL GetEnableSearchControl();

void SetEnableSearchControl(BOOL bNewValue);

BOOL GetEnableRefreshButton();

void SetEnableRefreshButton(BOOL bNewValue);

BOOL GetEnableDrillDown();

void SetEnableDrillDown(BOOL bNewValue);

BOOL GetEnableAnimationCtrl();

void SetEnableAnimationCtrl(BOOL bNewValue);

BOOL GetEnableSelectExpertButton();

void SetEnableSelectExpertButton(BOOL bNewValue);

void ViewReport();

BOOL GetEnableToolbar();

void SetEnableToolbar(BOOL bNewValue);

BOOL GetDisplayBorder();

void SetDisplayBorder(BOOL bNewValue);

BOOL GetDisplayTabs();

void SetDisplayTabs(BOOL bNewValue);

BOOL GetDisplayBackgroundEdge();

void SetDisplayBackgroundEdge(BOOL bNewValue);

CCRVTrackCursorInfo GetTrackCursorInfo();

short GetActiveViewIndex();

short GetViewCount();

void ActivateView(const VARIANT& Index);

void AddView(const VARIANT& GroupPath);

void CloseView(const VARIANT& Index);

VARIANT GetViewPath(short Index);

void PrintReport();

void Refresh();

void SearchForText(LPCTSTR Text);

void ShowFirstPage();

void ShowNextPage();

void ShowPreviousPage();

void ShowLastPage();

void ShowNthPage(short PageNumber);

void Zoom(short ZoomLevel);

long GetCurrentPageNumber();

void ShowGroup(const VARIANT& GroupPath);

BOOL GetIsBusy();

BOOL GetEnablePopupMenu();

void SetEnablePopupMenu(BOOL bNewValue);

BOOL GetEnableExportButton();

void SetEnableExportButton(BOOL bNewValue);

BOOL GetEnableSearchExpertButton();

void SetEnableSearchExpertButton(BOOL bNewValue);

void SearchByFormula(LPCTSTR formula);

CString GetViewName(BSTR\* pTabName);

BOOL GetEnableHelpButton();

void SetEnableHelpButton(BOOL bNewValue);

VARIANT GetGroup();

void GetLastPageNumber(long\* pageN, BOOL\* lastPageKnown);

void RefreshEx(BOOL refreshServerData);

};

//{{AFX\_INSERT\_LOCATION}}

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

#endif

#if !defined(AFX\_CRVTRACKCURSORINFO\_H\_\_DD218E31\_BEDD\_4A17\_A789\_56CA094AEEC3\_\_INCLUDED\_)

#define AFX\_CRVTRACKCURSORINFO\_H\_\_DD218E31\_BEDD\_4A17\_A789\_56CA094AEEC3\_\_INCLUDED\_

#if \_MSC\_VER > 1000

#pragma once

#endif // \_MSC\_VER > 1000

// Machine generated IDispatch wrapper class(es) created by Microsoft Visual C++

// NOTE: Do not modify the contents of this file. If this class is regenerated by

// Microsoft Visual C++, your modifications will be overwritten.

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

// CCRVTrackCursorInfo wrapper class

class CCRVTrackCursorInfo : public COleDispatchDriver

{

public:

CCRVTrackCursorInfo() {} // Calls COleDispatchDriver default constructor

CCRVTrackCursorInfo(LPDISPATCH pDispatch) : COleDispatchDriver(pDispatch) {}

CCRVTrackCursorInfo(const CCRVTrackCursorInfo& dispatchSrc) : COleDispatchDriver(dispatchSrc) {}

// Attributes

public:

// Operations

public:

long GetDetailAreaCursor();

void SetDetailAreaCursor(long nNewValue);

long GetDetailAreaFieldCursor();

void SetDetailAreaFieldCursor(long nNewValue);

long GetGraphCursor();

void SetGraphCursor(long nNewValue);

long GetGroupAreaCursor();

void SetGroupAreaCursor(long nNewValue);

long GetGroupAreaFieldCursor();

void SetGroupAreaFieldCursor(long nNewValue);

};

//{{AFX\_INSERT\_LOCATION}}

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

#endif

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <fcntl.h>

#include <getopt.h>

#include <errno.h>

#include <time.h>

#include <sys/ioctl.h>

#include <linux/config.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include "cgivars.h"

#include "htmllib.h"

#include "config\_phase.h"

#include "sysconf.h"

#define DEBUG 0

#define ID\_NTPFLAG 147

#define ID\_NTPADD 148

#define ID\_NTPTM 149

static config\_item \*\* items;

static int config\_list[] = {147,148,149};

static char \*config\_name[] = {"NTPFLAG", "NTPADD", "NTPTM"};

void date\_set(char \*\*postvars, int form\_method)

{

int config\_num = sizeof(config\_list)/sizeof(int);

int i;

int r\_fd, w\_fd;

int temp\_int, day, hour, min, sec;

char \*temp\_string;

time\_t the\_time;

struct tm \*tm\_ptr;

char data\_string[MAX\_LINE\_LENGTH];

char java[]="<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 Transitional//EN\">

<HTML><HEAD>

<META http-equiv=Content-Type content=\"text/html; charset=gb2312\">

<SCRIPT language=JavaScript src=\"js/javascript.js\">

</SCRIPT>

<SCRIPT language=javascript>

<!--

function manual\_time()

{

parent.frames[\"msgFrame\"].location.replace(\"/action?action=settime&lang=chs&systime=\" + document.getElementById('systime').value)

}

function auto\_time()

{

parent.frames[\"msgFrame\"].location.replace(\"/action?action=settime&lang=chs&systime=\" + document.getElementById('pcsystime').value)

}

var timerID = null

var timerRunning = false

function stopclock(){

if(timerRunning)

clearTimeout(timerID)

timerRunning = false

}

function startclock(){

stopclock()

showtime()

}

function showtime(){

var now = new Date()

var years = now.getFullYear()

var months = now.getMonth() + 1

var days = now.getDate()

var hours = now.getHours()

var minutes = now.getMinutes()

var seconds = now.getSeconds()

var timeValue = "" + years

timeValue += ((months < 10) ? \"-0\" : \"-\") + months

timeValue += ((days < 10) ? \"-0\" : \"-\") + days

timeValue += \" \"+((hours < 10) ? \"0\" : \"\") + hours

timeValue += ((minutes < 10) ? \":0\" : \":\") + minutes

timeValue += ((seconds < 10) ? \":0\" : \":\") + seconds

document.getElementById('pcsystime').value = timeValue

timerID = setTimeout(\"showtime()\",1000)

timerRunning = true

}

//-->

</SCRIPT>

<STYLE type=text/css>BODY {

FONT-SIZE: 10pt; FONT-FAMILY: Arial

}

TD {

FONT-SIZE: 10pt; FONT-FAMILY: Arial

}

FORM {

FONT-SIZE: 10pt; FONT-FAMILY: Arial

}

OPTION {

FONT-SIZE: 10pt; FONT-FAMILY: Arial

}

TEXTAREA {

FONT-SIZE: 10pt; FONT-FAMILY: Arial

}

P {

FONT-SIZE: 10pt; FONT-FAMILY: Arial

}

OL {

FONT-SIZE: 10pt; FONT-FAMILY: Arial

}

INPUT {

FONT-SIZE: 10pt; FONT-FAMILY: Arial

}

SELECT {

FONT-SIZE: 10pt; FONT-FAMILY: Arial

}

.style1 {

FONT-WEIGHT: bold; FONT-SIZE: 14pt

}

</STYLE>

<META content=\"MSHTML 6.00.6000.16414\" name=GENERATOR></HEAD>";

items = init\_config\_item(items, config\_num, config\_list);

printf("%s\n", java);

// r\_fd = open\_read\_fifo();

// If we got post, write it into flash

if(form\_method == POST)

{

GetConfValues(config\_list, items, config\_num);

SaveConfig(postvars, items, config\_list, config\_name, config\_num);

/\*

if(SaveConfig(postvars, items, config\_list, config\_name, config\_num))

{

w\_fd = open\_write\_fifo();

notify\_config(w\_fd);

close\_write\_fifo(w\_fd);

read\_fifo(r\_fd);

}

\*/

for (i=0; postvars[i]; i+= 2)

{

if(!strcmp(postvars[i], "PCTIME"))

{

char MM[2],DD[2],hh[2],mm[2],YY[4],ss[2];

temp\_string = strchr(postvars[i+1], '-');

temp\_string -= 4;

strncpy(YY, temp\_string, 4);

temp\_string += 5;

strncpy(MM, temp\_string, 2);

temp\_string += 3;

strncpy(DD, temp\_string, 2);

temp\_string += 3;

strncpy(hh, temp\_string, 2);

temp\_string += 3;

strncpy(mm, temp\_string, 2);

temp\_string += 3;

strncpy(ss, temp\_string, 2);

memset(data\_string, 0, MAX\_LINE\_LENGTH);

strcpy(data\_string, "date ");

strncat(data\_string, MM, 2);

strncat(data\_string, DD, 2);

strncat(data\_string, hh, 2);

strncat(data\_string, mm, 2);

strncat(data\_string, YY, 4);

strcat(data\_string, ".");

strncat(data\_string, ss, 2);

if(system(data\_string)<0)

{

;

}

}

}

}

GetConfValues(config\_list, items, config\_num);

printf("<BODY leftMargin=30 topMargin=24 onload=startclock() rightMargin=30>\n");

printf("<P><SPAN class=style1></SPAN></P>\n");

printf("<TABLE style=\"BORDER-COLLAPSE: collapse\" borderColor=#dae3eb cellSpacing=0 cellPadding=0 width=\"100%\" border=1>\n");

printf("<TBODY><TR bgColor=#cccccc><TD colSpan=3><STRONG></STRONG></TD></TR>\n");

memset(data\_string, 0, MAX\_LINE\_LENGTH);

time(&the\_time);

tm\_ptr = localtime(&the\_time);

sprintf(data\_string, "%04d-%02d-%02d %02d:%02d:%02d\n", tm\_ptr->tm\_year+1900, tm\_ptr->tm\_mon+1, tm\_ptr->tm\_mday,

tm\_ptr->tm\_hour, tm\_ptr->tm\_min, tm\_ptr->tm\_sec);

printf("<FORM name=settime action=\"/date.cgi\" method=post target=msgFrame>\n");

printf("<TR><TD noWrap width=\"32%\"> (%s)</TD>\n", data\_string);

printf("<TD><INPUT style=\"WIDTH: 180px\" type=submit value=PC></TD>\n");

printf("<TD width=\"30%\">&nbsp;<INPUT id=pcsystime maxLength=19 size=19 value=\"PC TIME\" name=PCTIME></TD></TR>\n");

printf("</FORM>");

printf("<FORM name=date action=\"/date.cgi\" method=post target=msgFrame>\n");

printf("<TR bgColor=#cccccc><TD colSpan=3><STRONG>NTP</STRONG></TD></TR>\n");

//NTPFLAG

printf("<TR><TD noWrap width=\"29%\">NTP</FONT></TD>\n");

printf("<TD align=middle colspan=3><DIV align=left><SELECT size=1 name=NTPFLAG>\n");

if(temp\_string = getdata(ID\_NTPFLAG, items, config\_num))

{

temp\_int = atoi(temp\_string);

if(temp\_int == 0)

{

printf("<option value=\"0\" selected></OPTION> \n");

printf("<option value=\"1\"></OPTION> \n");

}

else

{

printf("<option value=\"0\"></OPTION> \n");

printf("<option value=\"1\" selected></OPTION> \n");

}

}

printf("</SELECT></DIV></TD></TR>\n");

//ID\_NTPADD

printf("<TR><TD noWrap width=\"29%\">NTP</FONT></TD>\n");

if(temp\_string = getdata(ID\_NTPADD, items, config\_num))

printf("<TD align=middle colspan=3><DIV align=left><INPUT size=15 name=NTPADD value=%s></DIV></TD></TR>\n", temp\_string);

else

printf("<TD align=middle colspan=3><DIV align=left><INPUT size=15 name=NTPADD></DIV></TD></TR>\n");

//ID\_NTPTM

printf("<TR><TD noWrap width=\"29%\">()</FONT></TD>\n");

if(temp\_string = getdata(ID\_NTPTM, items, config\_num))

printf("<TD align=middle colspan=3><DIV align=left><INPUT size=15 name=NTPTM value=%s></DIV></TD></TR>\n", temp\_string);

else

printf("<TD align=middle colspan=3><DIV align=left><INPUT size=15 name=NTPTM></DIV></TD></TR>\n");

printf("<TR><TD colSpan=3><P align=left><INPUT type=submit value=><INPUT type=reset value=></P>\n");

printf("</TD></TR></FORM></TBODY></TABLE></BODY></HTML>\n");

free\_config\_item(items, config\_num);

//char \*loginName = getenv("REQUEST\_METHOD");

//printf("loginName :%s\n",loginName);

//int second = time((time\_t \*)NULL);

//printf("the time is :%d \n ",second);

//printf("%d\n",getpid());

//close\_read\_fifo();

}

int main(void)

{

char \*\*postvars = NULL; // POST request data repository

char \*\*getvars = NULL; // GET request data repository

int form\_method; // POST = 1, GET = 0

// CGI part start here

adminCheck();

form\_method = getRequestMethod();

if(form\_method == POST)

{

// getvars = getGETvars();

postvars = getPOSTvars();

}

else if(form\_method == GET)

{

getvars = getGETvars();

}

// CGI start here, set COM port

date\_set(postvars, form\_method);

//htmlFooter();

cleanUp(form\_method, getvars, postvars);

fflush(stdout);

exit(0);

} // end of main