

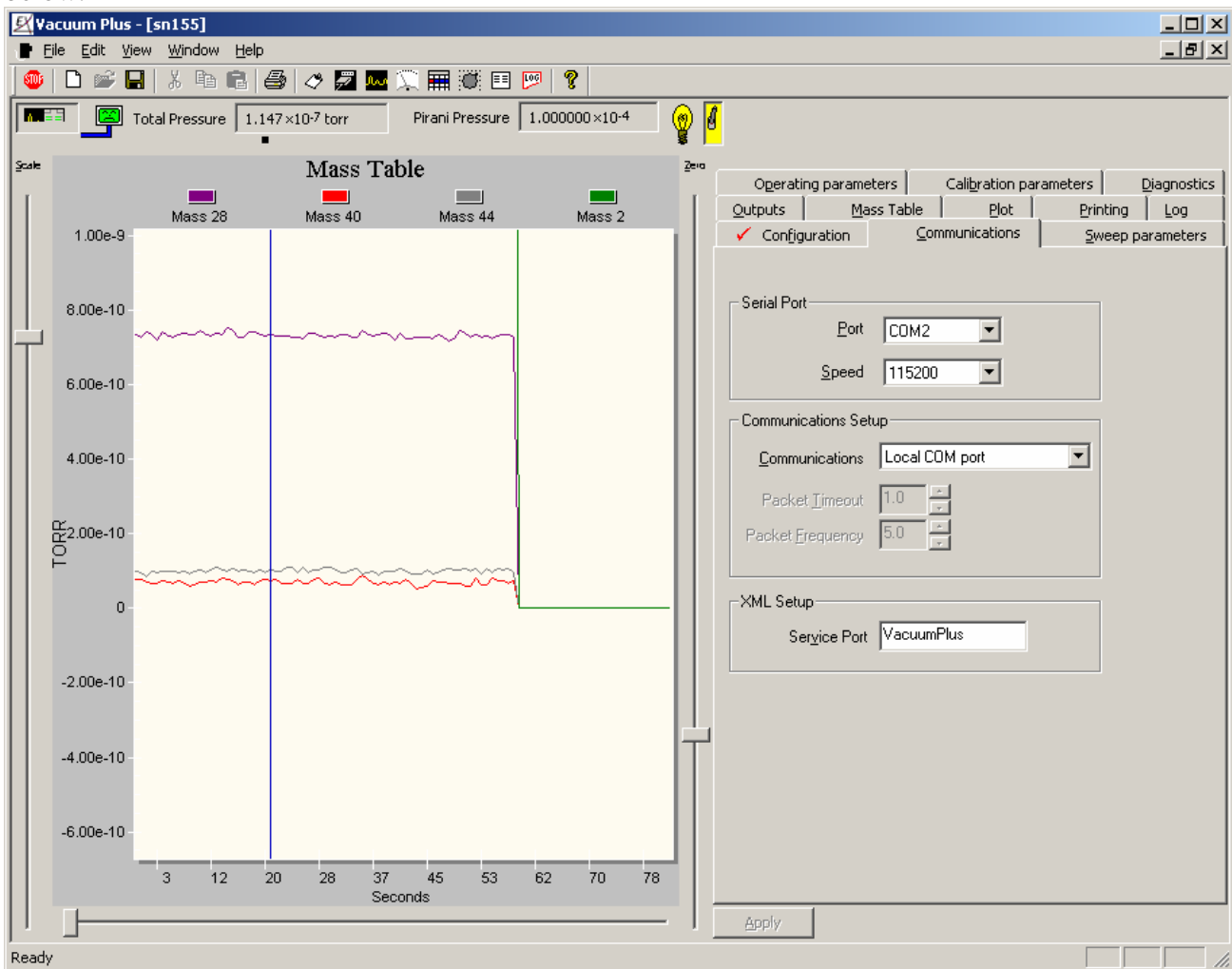


Extorr Inc.
307 Columbia Road
New Kensington, PA 15068
724-337-3000
Fax 724-337-8322

Application Note Number 105: Use of Steaming XML

Abstract: Streaming XML data is available from VacuumPlus. This allows data acquisition into user generated programs. This application note gives an example of how this is done.

The Extorr VacuumPlus software has capability of taking sweep and trend mode data out and to place configuration data into the Extorr manually using File>Save XML or Configuration>Open. The software also allows a streaming XML data output. After a program is written to take in the data, the first thing to do is to open a data pipe. This is done on the Communications Tab interface as shown below.



The Service Port is given the name of the port to which the streaming data is to be directed. In this case it is VacuumPlus.

The source data for a simple Visual C++ program that just collects and stores this data is given below.

```

// pipetestDlg.cpp : implementation file
//

#include "stdafx.h"
#include "stdio.h"
#include "pipetest.h"
#include "pipetestDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

    // Dialog Data
    //{{AFX_DATA(CAboutDlg)
    enum { IDD = IDD_ABOUTBOX };
    //}}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)
    //}}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)
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
//{{AFX_MSG_MAP(CAboutDlg)

```

```

// No message handlers
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CPipetestDlg dialog

CPipetestDlg::CPipetestDlg(CWnd* pParent /*=NULL*/)
: CDialog(CPipetestDlg::IDD, pParent)
{
    {{{AFX_DATA_INIT(CPipetestDlg)
    // NOTE: the ClassWizard will add member initialization here
    }}}AFX_DATA_INIT
    // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CPipetestDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    {{{AFX_DATA_MAP(CPipetestDlg)
    // NOTE: the ClassWizard will add DDX and DDV calls here
    }}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CPipetestDlg, CDialog)
//{{AFX_MSG_MAP(CPipetestDlg)
ON_WM_SYSCOMMAND()
ON_WM_PAINT()
ON_WM_QUERYDRAGICON()
ON_BN_CLICKED(IDC_BUTTON3, Ondisconnect)
ON_WM_TIMER()
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CPipetestDlg message handlers

#define Max_Buffer_Size 65535

HANDLE pipe;
CString strMessage;
CListBox *plistbox;
TCHAR buffer [Max_Buffer_Size];
DWORD bytesRead;

CStdioFile f;
CString s;
CString caption;

BOOL CPipetestDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    // Add "About..." menu item to system menu.

    // IDM_ABOUTBOX must be in the system command range.
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

```

```

CMenu* pSysMenu = GetSystemMenu(FALSE);
if (pSysMenu != NULL)
{
    CString strAboutMenu;
    strAboutMenu.LoadString(IDS_ABOUTBOX);
    if (!strAboutMenu.IsEmpty())
    {
        pSysMenu->AppendMenu(MF_SEPARATOR);
        pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
    }
}

```

```

// Set the icon for this dialog. The framework does this automatically
// when the application's main window is not a dialog
SetIcon(m_hIcon, TRUE); // Set big icon
SetIcon(m_hIcon, FALSE); // Set small icon

```

```

//HANDLE pipe = CreateFile (_T("\\\\.\\pipe\\VacuumPlus"), GENERIC_READ, 0, NULL, OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL, NULL);
pipe = CreateFile (_T("\\\\.\\pipe\\VacuumPlus"),
GENERIC_READ, 0, NULL,
OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL,
NULL);
if (pipe == INVALID_HANDLE_VALUE)
{
    DWORD err= ::GetLastError ();
    strMessage.Format(_T( "Error %d opening pipe\n", err)),
    plistbox = (CListBox *) GetDlgItem(IDC_LIST1);
    plistbox->AddString(strMessage);
}
else
{
    strMessage.Format(_T( "pipe opened")),
    plistbox = (CListBox *) GetDlgItem(IDC_LIST1);
    plistbox->AddString(strMessage);
}

```

```

CFileDialog dlg(TRUE, // file open
NULL,
NULL, // Initial filename
0, // flags
"", //filter
NULL
);
caption = "local file";
dlg.m_ofn.lpstrTitle = caption;

```

```

if(dlg.DoModal() != IDOK)
return false;

```

```

CString name = dlg.GetPathName();

```

```

if(!f.Open(name, CFile::modeCreate | CFile::modeWrite | CFile::shareExclusive))
{ /* open failed */
    MessageBox ( "Error occurred while opening file" );
    return false ;
} /* open failed */

```

```

// m_filename.SetWindowText( name );
// m_filename.LineScroll( 0, 100 );

SetTimer( 1, 5000, NULL );

return TRUE; // return TRUE unless you set the focus to a control
}

void CPipetestDlg::OnTimer(UINT nIDEvent)
{
if (ReadFile(pipe, buffer, Max_Buffer_Size-sizeof(TCHAR),&bytesRead,NULL))
{
if(bytesRead != 0)
{

f.Write( buffer, bytesRead);

buffer[100] = ('\0'); //limit display to 1st 20 char
strMessage.Format(_T( buffer),
plistbox = (CListBox *) GetDlgItem(IDC_LIST1);
plistbox->AddString(strMessage);
}

CDialog::OnTimer(nIDEvent);
}

}

void CPipetestDlg::OnDisconnect()
{
KillTimer( 1);
f.Close();
::CloseHandle (pipe);
strMessage.Format(_T( "pipe closed")),
plistbox = (CListBox *) GetDlgItem(IDC_LIST1);
plistbox->AddString(strMessage);

}

void CPipetestDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
if ((nID & 0xFFF0) == IDM_ABOUTBOX)
{
CAboutDlg dlgAbout;
dlgAbout.DoModal();
}
else
{
CDialog::OnSysCommand(nID, lParam);
}
}

```

// If you add a minimize button to your dialog, you will need the code below

```

// to draw the icon. For MFC applications using the document/view model,
// this is automatically done for you by the framework.

void CPipetestDlg::OnPaint()
{
if (IsIconic())
{
CPaintDC dc(this); // device context for painting

SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

// Center icon in client rectangle
int cxIcon = GetSystemMetrics(SM_CXICON);
int cyIcon = GetSystemMetrics(SM_CYICON);
CRect rect;
GetClientRect(&rect);
int x = (rect.Width() - cxIcon + 1) / 2;
int y = (rect.Height() - cyIcon + 1) / 2;

// Draw the icon
dc.DrawIcon(x, y, m_hIcon);
}
else
{
CDialog::OnPaint();
}
}

// The system calls this to obtain the cursor to display while the user drags
// the minimized window.
HCURSOR CPipetestDlg::OnQueryDragIcon()
{
return (HCURSOR) m_hIcon;
}

```

This program is available in compiled form on the Extorr web site as pipetest.exe.

The program reads the pipe every 5 seconds and writes the data to a file. The user is prompted for a file name. Upon supplying the name, the program starts gathering data from the stream. The file will close when the "Disconnect the pipe and write the file" button is pushed. The data box will display the first few characters of the data while it writes. The program must be started over again to save more after closing the pipe

Again, all the user has to do is:

1. put xml1.exe on your desktop
2. set the service port to "VacuumPlus" and apply.
3. double click on xml1.exe

The user should see the data coming from the port. After the application is stopped, the data may be handled in Excel as was shown in Application Notes 102 and 103.