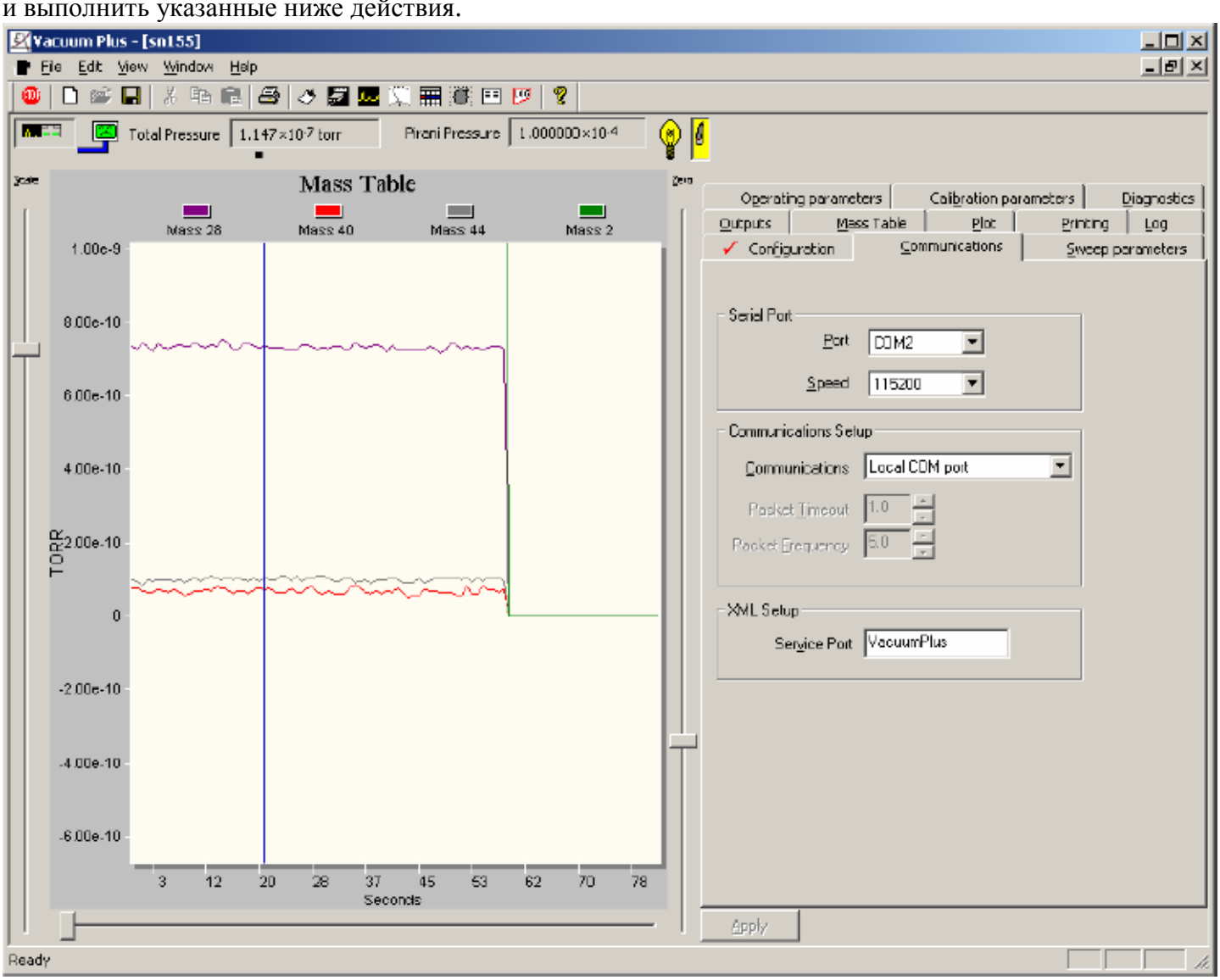
Указание по применению № 105: использование потоковой передачи XML-данных

Краткий обзор: Из ПО VacuumPlus можно осуществлять потоковую передачу XML-данных. Это позволяет передавать данные в программы, написанные пользователем. Данное указание по применению содержит пример осуществления такой передачи.

ПО Extorr VacuumPlus способно выводить данные режимов тренда и развертки и передавать данные конфигурации в Extorr вручную, используя «File (Файл) > Save XML (Сохранить как XML)» или «Configuration (Конфигурация) > Open (Открыть)». Кроме того, это ПО позволяет осуществлять потоковый вывод XML-данных. После написания программы для приема данных первое, что нужно сделать, это открыть конвейер данных. Для этого необходимо открыть вкладку Communications (Связь) и выполнить указанные ниже действия.



Параметру «Service Port (Порт службы)» задается имя порта, которому должны будут направляться потоковые данные. В данном случае это «VacuumPlus».

Ниже приведен исходный код простой программы на Visual C++, которая только собирает и сохраняет эти данные.

```
// 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 (("\\\\.\\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;
}
```

Эту программу в скомпилированной форме можно загрузить с веб-сайта компании Extorr в виде файла pipetest.exe
Программа считывает конвейер каждые 5 секунд и записывает данные в файл. Пользователь получает приглашение ввести имя файла. После ввода имени, программа начинает сбор данных из потока. Файл закрывается после нажатия кнопки «Disconnect the pipe and write the file (Отключить конвейер и записать файл)». В окне данных будут отображаться первые несколько символов данных в процессе их записи. После закрытия конвейера, чтобы сохранить дополнительные данные, программу необходимо запустить вновь.
Пользователь должен выполнить следующие действия:
1. Поместить файл xml1.exe на рабочий стол.
2. Задать параметру «service port» значение «VacuumPlus» и применить.
3. Дважды щелкнуть мышью файл xml1.exe.
Пользователь должен увидеть данные, поступающие от порта. После завершения работы приложения, данные можно обработать в Excel, как показано в указаниях по применению № 102 и 103.