// tewDlg.cpp : 实现文件

//

#include "stdafx.h"

#include "tew.h"

#include "tewDlg.h"

#include "trmDlg.h"

#include "aboutdlg.h"

#include "nodepropertydlg.h"

#include "afxdialogex.h"

#include "dlgdutmsg.h"

#include "dlgproductresult.h"

#include "dlgprocessing.h"

#include "tcpserver.h"

#include "udp.h"

#include "dlgconfig.h"

#include "mmsystem.h"

#pragma comment(lib,"winmm")

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// CtewDlg 对话框

CtewDlg::CtewDlg(CWnd\* pParent /\*=NULL\*/)

: CDialogEx(CtewDlg::IDD, pParent)

, m\_modctrlscaleflag(0)

, m\_pcuritemdata(0)

, m\_timer(0)

, m\_pmysql(0)

, m\_hAccTable(0)

, m\_pmenu(0)

, m\_ptoolbarctrl(0)

, m\_hsemaphore(0)

, m\_ptcpserver(0)

, m\_pudp(0)

{

m\_hIcon=AfxGetApp()->LoadIcon(IDR\_MAINFRAME);

m\_dlocalver.vmah=0;

m\_configflag.loop=0;

m\_ctrlpositionarray[0][0]=0;

m\_ptrmtabpageitemdataarray[0]=NULL;

}

void CtewDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialogEx::DoDataExchange(pDX);

DDX\_Control(pDX, IDC\_cfgtree, m\_cfgtree);

DDX\_Control(pDX, IDC\_trmtab, m\_trmtab);

DDX\_Control(pDX, IDC\_loglist, m\_loglist);

}

BEGIN\_MESSAGE\_MAP(CtewDlg, CDialogEx)

ON\_WM\_PAINT()

ON\_WM\_QUERYDRAGICON()

ON\_COMMAND(ID\_dlgexit, &CtewDlg::Ondlgexit)

ON\_WM\_SIZE()

ON\_WM\_GETMINMAXINFO()

ON\_COMMAND(ID\_ctrlloglistclear,&CtewDlg::Onctrlloglistclear)

ON\_MESSAGE(WM\_appdlgctrladdloglist,&CtewDlg::Onappdlgctrladdloglist)

ON\_MESSAGE(WM\_appdlgctrlsetstatusbar,&CtewDlg::Onappdlgctrlsetstatusbar)

ON\_MESSAGE(WM\_appdlgnodepropertymodify,&CtewDlg::Onappdlgnodepropertymodify)

ON\_MESSAGE(WM\_appdlgtrmdlgclose,&CtewDlg::Onappdlgtrmdlgclose)

ON\_NOTIFY(NM\_RCLICK, IDC\_loglist, &CtewDlg::OnNMRClickloglist)

ON\_WM\_DROPFILES()

ON\_WM\_MOUSEMOVE()

ON\_WM\_LBUTTONDOWN()

ON\_WM\_LBUTTONUP()

ON\_COMMAND(ID\_dlgopenfile, &CtewDlg::OnDlgopenfile)

ON\_COMMAND(ID\_dlgopendir, &CtewDlg::OnDlgopendir)

ON\_COMMAND(ID\_dlgpause, &CtewDlg::OnDlgpause)

ON\_COMMAND(ID\_dlgstart, &CtewDlg::OnDlgstart)

ON\_COMMAND(ID\_dlgstop, &CtewDlg::OnDlgstop)

ON\_COMMAND(ID\_dlghelpdoc, &CtewDlg::OnDlghelpdoc)

ON\_COMMAND(ID\_about, &CtewDlg::OnAbout)

ON\_COMMAND(ID\_dlgexplorer, &CtewDlg::OnDlgexplorer)

ON\_NOTIFY(NM\_RCLICK, IDC\_cfgtree, &CtewDlg::OnNMRClickcfgtree)

ON\_COMMAND(ID\_dlgenablenode, &CtewDlg::Ondlgenablenode)

ON\_COMMAND(ID\_dlgdisablenode, &CtewDlg::Ondlgdisablenode)

ON\_COMMAND(ID\_ctrlcfgtreeproperty, &CtewDlg::Onctrlcfgtreeproperty)

ON\_NOTIFY(NM\_DBLCLK, IDC\_cfgtree, &CtewDlg::OnNMDblclkcfgtree)

ON\_COMMAND(ID\_dlgsaveas, &CtewDlg::Ondlgsaveas)

ON\_COMMAND(ID\_dlgsave, &CtewDlg::Ondlgsave)

ON\_COMMAND(ID\_dlgedit, &CtewDlg::Ondlgedit)

ON\_NOTIFY(TVN\_SELCHANGED, IDC\_cfgtree, &CtewDlg::OnTvnSelchangedcfgtree)

ON\_NOTIFY(TCN\_SELCHANGE, IDC\_trmtab, &CtewDlg::OnTcnSelchangetrmtab)

ON\_COMMAND(ID\_dlglog, &CtewDlg::Ondlglog)

ON\_COMMAND(ID\_dlgcsv, &CtewDlg::Ondlgcsv)

ON\_MESSAGE(WM\_appdlgtrmdlgtrmfaultcounter, &CtewDlg::Onappdlgtrmdlgtrmfaultcounter)

ON\_MESSAGE(WM\_appdlgsetsavevalue, &CtewDlg::Onappdlgsetsavevalue)

ON\_MESSAGE(WM\_appdlggetsavevalue, &CtewDlg::Onappdlggetsavevalue)

ON\_MESSAGE(WM\_appdlgsetcsv, &CtewDlg::Onappdlgsetcsv)

ON\_MESSAGE(WM\_appdlgreplacelabel, &CtewDlg::Onappdlgreplacelabel)

ON\_MESSAGE(WM\_appdlgmodule, &CtewDlg::Onappdlgmodule)

ON\_WM\_TIMER()

ON\_MESSAGE(WM\_appdlgovertime, &CtewDlg::Onappdlgovertime)

ON\_COMMAND(ID\_dlgconfig, &CtewDlg::Ondlgconfig)

ON\_MESSAGE(WM\_appdlgconfig, &CtewDlg::Onappdlgconfig)

ON\_MESSAGE(WM\_appdlgmsg, &CtewDlg::Onappdlgmsg)

ON\_MESSAGE(WM\_appdlgtcpservermsg, &CtewDlg::Onappdlgtcpservermsg)

ON\_MESSAGE(WM\_appdlgctrlstop, &CtewDlg::Onappdlgctrlstop)

ON\_COMMAND(ID\_dlgopenpath, &CtewDlg::Ondlgopenpath)

ON\_COMMAND(IDR\_ACCELERATOR, &CtewDlg::OnIdrAccelerator)

END\_MESSAGE\_MAP()

// CtewDlg 消息处理程序

BOOL CtewDlg::OnInitDialog()

{

CDialogEx::OnInitDialog();

// 设置此对话框的图标.当应用程序主窗口不是对话框时,框架将自动

// 执行此操作

SetIcon(m\_hIcon, TRUE); // 设置大图标

SetIcon(m\_hIcon, FALSE); // 设置小图标

// TODO: 在此添加额外的初始化代码

// 初始化配置标志

m\_configflag.loop=0;

m\_configflag.ngstop=0;

// 初始化app环境,包含界面初始化

initall();

// menu deny

setallcmddeny();

// app ready

CString t\_cs=\_T("[tew]ready");

this->SendMessage(WM\_appdlgctrladdloglist,(WPARAM)t\_cs.GetBuffer(0),0);

t\_cs.ReleaseBuffer();

// 默认回调函数OnTimer

SetTimer(APPDLG1STIMER,1000,NULL);

SetTimer(APPDLG60STIMER,60000,NULL);

// menu ready

setcmdinready();

// start sound

PlaySound(MAKEINTRESOURCE(IDR\_waveerror),AfxGetApp()->m\_hInstance,SND\_RESOURCE|SND\_ASYNC|SND\_NODEFAULT);

//载入快捷键资源,其中IDR\_ACCELERATOR为快捷键表的ID

m\_hAccTable=LoadAccelerators(AfxGetInstanceHandle(),MAKEINTRESOURCE(IDR\_ACCELERATOR));

return TRUE; // 除非将焦点设置到控件,否则返回 TRUE

}

// config

afx\_msg LRESULT CtewDlg::Onappdlgconfig(WPARAM wParam, LPARAM lParam)

{

// 指针传递,无需参数赋值

return 0;

}

// ctrl log list 增加内容

afx\_msg LRESULT CtewDlg::Onappdlgctrladdloglist(WPARAM wParam, LPARAM lParam)

{

CString t\_csdate;

// 获取时间

t\_csdate=getsystemdatetime();

CString t\_linecount;

t\_linecount.Format( \_T("%d"),m\_loglist.GetItemCount()+1);

m\_loglist.InsertItem(0,t\_linecount);

m\_loglist.SetItemText(0,1,t\_csdate);

m\_loglist.SetItemText(0,2,(LPCTSTR)wParam);

// dlg改变后重绘状态栏,包含分隔比例

if(IsWindow(m\_statusbarctrl.m\_hWnd))

{

// 添加状态栏,v\_i为1创建,0为重绘

initdlgviewstatusbar(FALSE);

}

// dlg改变后重绘log list,包含分隔比例

if(IsWindow(m\_loglist.m\_hWnd))

{

// 重绘log list

repaintloglist();

}

CString t\_cslog=(LPCTSTR)wParam;

t\_csdate+=\_T("\t")+t\_cslog+\_T("\n");

m\_csfapplogfile.WriteString(t\_csdate);

return 0;

}

// log list clear

void CtewDlg::Onctrlloglistclear()

{

m\_loglist.DeleteAllItems();

// log list界面向下扩展覆盖部分status bar空间,因此重绘dlg

CRect t\_rect;

GetClientRect(&t\_rect);

repaintctrl(t\_rect.Width(),t\_rect.Height());

}

// set ctrl statusbar

// 第npane个分栏加入pcstring

// wParam:字符串指针;lParam:状态栏编号

afx\_msg LRESULT CtewDlg::Onappdlgctrlsetstatusbar(WPARAM wParam, LPARAM lParam)

{

assert(lParam<STATUSBARSEPARATES);

// 第npane分栏加入pstr

m\_statusbarctrl.SetText((LPCTSTR)wParam,lParam,0);

return 0;

}

// trm msg

afx\_msg LRESULT CtewDlg::Onappdlgmsg(WPARAM wParam, LPARAM lParam)

{

struct treeitemdata \*t\_itemdata=reinterpret\_cast<treeitemdata\*>(wParam);

int t\_i=FALSE;

assert(t\_itemdata);

switch(lParam)

{

case 1:

// add

m\_trmmsg+=t\_itemdata->information;

t\_i=TRUE;

break;

case 2:

// check

if(m\_trmmsg.Find(t\_itemdata->information)!=-1)

{

t\_i=TRUE;

}

else

{

t\_i=FALSE;

}

break;

case 3:

// chkclr

if(m\_trmmsg.Find(t\_itemdata->information)!=-1)

{

m\_trmmsg.Replace(t\_itemdata->information,\_T(""));

t\_i=TRUE;

}

else

{

t\_i=FALSE;

}

break;

case 4:

// clear

m\_trmmsg.Empty();

t\_i=TRUE;

break;

default:

break;

}

if(!t\_i)

{

this->SendMessage(WM\_appdlgnodepropertymodify,wParam,1);

}

else

{

this->SendMessage(WM\_appdlgnodepropertymodify,wParam,0);

}

t\_itemdata->msgoperate=t\_i;

return 0;

}

// module begin or end

afx\_msg LRESULT CtewDlg::Onappdlgmodule(WPARAM wParam, LPARAM lParam)

{

struct treeitemdata \*t\_itemdata=reinterpret\_cast<treeitemdata\*>(wParam);

struct treeitemdata \*t\_rootid=reinterpret\_cast<treeitemdata\*>(m\_cfgtree.GetItemData(m\_cfgtree.GetRootItem()));

assert(t\_itemdata);

CString t\_result;

if(lParam==0)

{

// 模块开始 初值

t\_result.Format(\_T("%d"),t\_rootid->faultcounter);

}

else

{

// 模块结束 差值

int t\_i;

if(m\_mapmodulefield.find(t\_itemdata->information)!=m\_mapmodulefield.end())

{

t\_i=\_ttoi(m\_mapmodulefield[t\_itemdata->information]);

}

else

{

t\_i=0;

}

t\_i=t\_rootid->faultcounter-t\_i;

t\_result.Format(\_T("%d"),t\_i);

}

m\_mapmodulefield[t\_itemdata->information]=t\_result;

return 0;

}

// modify node property

afx\_msg LRESULT CtewDlg::Onappdlgnodepropertymodify(WPARAM wParam, LPARAM lParam)

{

CString t\_cs;

struct treeitemdata \*t\_itemdata=reinterpret\_cast<treeitemdata\*>(wParam);

assert(t\_itemdata);

// 属性修改dlg节点使能/去使能check为只读

m\_cfgtree.SetItemImage(t\_itemdata->node,t\_itemdata->runstatus,t\_itemdata->runstatus);

if((lParam==1)&&(m\_cfgtree.GetItemText(t\_itemdata->node)[0]==\_T('<')))

{

return 0;

}

if(t\_itemdata->faultcounter>0)

{

t\_cs.Format(\_T("[NG:%d]"),t\_itemdata->faultcounter);

t\_cs+=t\_itemdata->comment;

m\_cfgtree.SetItemText(t\_itemdata->node,t\_cs);

}

else if(t\_itemdata->faultcounter==0)

{

t\_cs=t\_itemdata->comment;

m\_cfgtree.SetItemText(t\_itemdata->node,t\_cs);

}

// lParam为1表示有msg check动作,修改相应节点显示

if(lParam==1)

{

if(m\_cfgtree.GetItemText(t\_itemdata->node)[0]!=\_T('<'))

{

int t\_i=6;

CString t\_cst,t\_csdisp;

t\_cst=t\_itemdata->information;

t\_cst=t\_cst.Left(t\_cst.GetLength()-1);

if(t\_cst.GetLength()>t\_i)

{

t\_cst=t\_cst.Left(t\_i);

}

t\_csdisp=\_T("<")+t\_cst+\_T(">")+t\_cs;

m\_cfgtree.SetItemText(t\_itemdata->node,t\_csdisp);

}

}

return 0;

}

// get timer

afx\_msg LRESULT CtewDlg::Onappdlgovertime(WPARAM wParam, LPARAM lParam)

{

struct treeitemdata \*t\_itemdata=reinterpret\_cast<treeitemdata\*>(wParam);

assert(t\_itemdata);

t\_itemdata->overtime=this->m\_timer;

return 0;

}

// replace label

afx\_msg LRESULT CtewDlg::Onappdlgreplacelabel(WPARAM wParam, LPARAM lParam)

{

struct treeitemdata \*t\_itemdata=reinterpret\_cast<treeitemdata\*>(wParam);

assert(t\_itemdata);

if(t\_itemdata->fieldname[0]!=\_T('\_'))

{

// 获取message字符串

if(!(m\_mapdutmessage.find(t\_itemdata->fieldname)!=m\_mapdutmessage.end()))

{

CString t\_cs;

t\_cs.Format(\_T("配置文件存在未定义的替换标志(%s),请检查"),t\_itemdata->fieldname);

AfxMessageBox(t\_cs);

t\_itemdata->fieldvalue.Empty();

}

else

{

t\_itemdata->fieldvalue=m\_mapdutmessage[t\_itemdata->fieldname];

}

}

else

{

// 获取save变量字符串

int t\_i;

t\_i=\_ttoi(t\_itemdata->fieldname.Right(t\_itemdata->fieldname.GetLength()-1));

if((t\_i>=0)&&(t\_i<=127))

{

t\_itemdata->fieldvalue=m\_savevalue[t\_i];

}

else

{

AfxMessageBox(\_T("获取save变量参数异常"));

}

}

return 0;

}

// 出现错误

afx\_msg LRESULT CtewDlg::Onappdlgtrmdlgtrmfaultcounter(WPARAM wParam, LPARAM lParam)

{

struct treeitemdata \*t\_itemdata=reinterpret\_cast<treeitemdata\*>(wParam);

assert(t\_itemdata);

HTREEITEM t\_node=t\_itemdata->node;

for(;t\_node!=NULL;)

{

t\_itemdata=reinterpret\_cast<treeitemdata\*>(m\_cfgtree.GetItemData(t\_node));

t\_itemdata->faultcounter++;

t\_itemdata->runstatus=iconrwf;

this->SendMessage(WM\_appdlgnodepropertymodify,(WPARAM)t\_itemdata,NULL);

t\_node=m\_cfgtree.GetParentItem(t\_node);

}

// 出现错误,启动声音提示

PlaySound(MAKEINTRESOURCE(IDR\_waveerror),AfxGetApp()->m\_hInstance,SND\_RESOURCE|SND\_ASYNC|SND\_NODEFAULT);

// ng 触发stop

if(m\_configflag.ngstop)

{

// 禁用循环执行标志

m\_configflag.loop=FALSE;

// 获取ID\_dlgstart菜单状态

UINT v\_i=m\_pmenu->GetSubMenu(1)->GetMenuState(ID\_dlgstart,MF\_BYCOMMAND);

// 判断菜单状态是否灰化

if(v\_i&MF\_GRAYED)

{

// stop

this->PostMessage(WM\_appdlgctrlstop,NULL,NULL);

}

}

return 0;

}

// 设置存储值

afx\_msg LRESULT CtewDlg::Onappdlgsetsavevalue(WPARAM wParam, LPARAM lParam)

{

struct treeitemdata \*t\_itemdata=reinterpret\_cast<treeitemdata\*>(wParam);

assert(t\_itemdata);

assert(lParam<SAVEVALUECOUNTER);

assert(lParam>=0);

m\_savevalue[lParam]=t\_itemdata->information;

return 0;

}

// ctrl stop

afx\_msg LRESULT CtewDlg::Onappdlgctrlstop(WPARAM wParam, LPARAM lParam)

{

this->OnDlgstop();

return 0;

}

// 读取存储值

afx\_msg LRESULT CtewDlg::Onappdlggetsavevalue(WPARAM wParam, LPARAM lParam)

{

struct treeitemdata \*t\_itemdata=reinterpret\_cast<treeitemdata\*>(wParam);

assert(t\_itemdata);

t\_itemdata->information=m\_savevalue[lParam];

return 0;

}

// setcsv

afx\_msg LRESULT CtewDlg::Onappdlgsetcsv(WPARAM wParam, LPARAM lParam)

{

if(wParam==NULL)

{

m\_csfcsv.WriteString(\_T("\n"));

}

else

{

CString t\_cs=m\_savevalue[lParam]+\_T(",");

m\_csfcsv.WriteString(t\_cs);

}

return 0;

}

// tcp server msg

afx\_msg LRESULT CtewDlg::Onappdlgtcpservermsg(WPARAM wParam, LPARAM lParam)

{

CString t\_cs,t\_cstemp;

t\_cs.Empty();

switch(lParam)

{

case 1:

// msg

t\_cs=this->m\_trmmsg;

break;

case 2:

// curtrm

for(int t\_i=0;t\_i<TABCTRLMAXPAGENUMBER;t\_i++)

{

if(m\_ptrmtabpageitemdataarray[t\_i]==NULL)

{

t\_cs+=\_T("");

break;

}

else

{

t\_cstemp.Format(\_T("[%s]\r\n\tlevel:\t\t%d\r\n\tfile name:\t%s\r\n\tline number:\t%d\r\n"),m\_ptrmtabpageitemdataarray[t\_i]->comment,m\_ptrmtabpageitemdataarray[t\_i]->level,m\_ptrmtabpageitemdataarray[t\_i]->filename,m\_ptrmtabpageitemdataarray[t\_i]->lineno);

t\_cs+=t\_cstemp;

}

}

break;

case 3:

// value

for(int t\_i=0;t\_i<SAVEVALUECOUNTER;t\_i++)

{

t\_cstemp.Format(\_T("[%d]\t"),t\_i);

t\_cs+=t\_cstemp+m\_savevalue[t\_i]+\_T("\r\n");

}

break;

default:

break;

}

\*((CString\*)wParam)=t\_cs;

return 0;

}

// trm dialog close message

afx\_msg LRESULT CtewDlg::Onappdlgtrmdlgclose(WPARAM wParam, LPARAM lParam)

{

// 使用信号量进行PV操作

WaitForSingleObject(m\_hsemaphore,INFINITE);

struct treeitemdata \*t\_itemdata=reinterpret\_cast<treeitemdata\*>(wParam);

assert(t\_itemdata->exitflag==0);

t\_itemdata->exitflag=1;

if(t\_itemdata->faultcounter==0)

{

t\_itemdata->runstatus=iconcwn;

}

else if(t\_itemdata->faultcounter>0)

{

t\_itemdata->runstatus=iconcwf;

}

else

{

assert(0);

}

this->SendMessage(WM\_appdlgnodepropertymodify,(WPARAM)t\_itemdata,NULL);

// 检查节点完成状态,延伸至父节点

setuserdefinenodeexitflag(t\_itemdata);

volatile int t\_ic;

m\_trmtab.DeleteItem(t\_itemdata->trmtabctrlnumber);

t\_ic=m\_trmtab.GetItemCount();

volatile int t\_i;

for(t\_i=t\_itemdata->trmtabctrlnumber;m\_ptrmtabpageitemdataarray[t\_i+1]!=NULL;t\_i++)

{

m\_ptrmtabpageitemdataarray[t\_i]=m\_ptrmtabpageitemdataarray[t\_i+1];

m\_ptrmtabpageitemdataarray[t\_i]->trmtabctrlnumber-=1;

}

m\_ptrmtabpageitemdataarray[t\_i]=NULL;

if(m\_pcuritemdata!=NULL)

{

if(m\_pcuritemdata->trmtabctrlnumber!=t\_itemdata->trmtabctrlnumber)

{

}

else

{

if(t\_ic>0)

{

setcurtreenode(m\_ptrmtabpageitemdataarray[t\_ic-1]->node);

settrmtabpage(t\_ic-1);

}

else if(t\_ic==0)

{

m\_pcuritemdata=NULL;

}

else

{

assert(0);

}

}

}

// 使用信号量进行PV操作

ReleaseSemaphore(m\_hsemaphore,1,NULL);

runtewdlgstart(m\_cfgtree.GetRootItem());

return 0;

}

// 如果向对话框添加最小化按钮,则需要下面的代码

// 来绘制该图标.对于使用文档/视图模型的 MFC 应用程序,

// 这将由框架自动完成.

void CtewDlg::OnPaint()

{

if (IsIconic())

{

CPaintDC dc(this); // 用于绘制的设备上下文

SendMessage(WM\_ICONERASEBKGND, reinterpret\_cast<WPARAM>(dc.GetSafeHdc()), 0);

// 使图标在工作区矩形中居中

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;

// 绘制图标

dc.DrawIcon(x, y, m\_hIcon);

}

else

{

CDialogEx::OnPaint();

}

}

// 初始化app\_cfg.dat&mode\_comm.dat

int CtewDlg::initfile(CString v\_filename,CString v\_cs)

{

CStdioFile t\_csf;

if(t\_csf.Open(getappdir()+v\_filename,CFile::modeRead|CFile::shareDenyNone))

{

t\_csf.Close();

}

else

{

if(t\_csf.Open(getappdir()+v\_filename,CFile::modeWrite|CFile::modeCreate|CFile::shareDenyNone))

{

t\_csf.WriteString(v\_cs);

t\_csf.Close();

}

else

{

AfxMessageBox(\_T("配置文件创建失败.请检查目录访问权限或空间,尝试以管理员身份运行."));

assert(0);

}

}

return 0;

}

// 初始化log目录

int CtewDlg::pathinitlogpath(CString v\_logpath)

{

m\_cslogpath=m\_mapappcfg[\_T("lastpath")]+LOG\_PATH;

m\_cscsvpath=m\_cslogpath+m\_csstarttime+\_T("\\");

struct treeitemdata \*t\_pitemdata=reinterpret\_cast<treeitemdata\*>(m\_cfgtree.GetItemData(m\_cfgtree.GetRootItem()));

t\_pitemdata->logpath=m\_cscsvpath;

pathmkdir(m\_cscsvpath);

return 0;

}

// 初始化csv文件

int CtewDlg::pathinitcsvfile()

{

if(m\_csfcsv.Open(m\_cscsvpath+CSV\_FILE,CFile::modeWrite|CFile::modeCreate|CFile::shareDenyNone))

{

}

else

{

assert(0);

}

return 0;

}

// 判断目录是否存在

bool CtewDlg::pathgetfolderexists(CString v\_cspath)

{

DWORD t\_attr;

t\_attr=GetFileAttributes(v\_cspath);

return (t\_attr!=(DWORD)(-1))&&(t\_attr&FILE\_ATTRIBUTE\_DIRECTORY);

}

// 递归创建目录， 如果目录已经存在或者创建成功返回TRUE

bool CtewDlg::pathmkdir(CString v\_cspath)

{

int t\_len=v\_cspath.GetLength();

if(t\_len<2)

{

return false;

}

if(\_T('\\')==v\_cspath[t\_len-1])

{

v\_cspath=v\_cspath.Left(t\_len-1);

t\_len=v\_cspath.GetLength();

}

if(t\_len<=0)

{

return false;

}

if(t\_len <=3)

{

if(pathgetfolderexists(v\_cspath))

{

return true;

}

else

{

return false;

}

}

if(pathgetfolderexists(v\_cspath))

{

return true;

}

CString t\_Parent;

t\_Parent=v\_cspath.Left(v\_cspath.ReverseFind(\_T('\\')));

if(t\_Parent.GetLength()<=0)

{

return false;

}

bool t\_Ret=pathmkdir(t\_Parent);

if(t\_Ret)

{

SECURITY\_ATTRIBUTES t\_sa;

t\_sa.nLength=sizeof(SECURITY\_ATTRIBUTES);

t\_sa.lpSecurityDescriptor=NULL;

t\_sa.bInheritHandle=0;

t\_Ret=(CreateDirectory(v\_cspath,&t\_sa)==TRUE);

return t\_Ret;

}

else

{

return false;

}

}

// 初始化app配置map

int CtewDlg::initmapappcfg()

{

m\_mapappcfg.clear();

CStdioFile t\_csf;

if(t\_csf.Open(getappdir()+APP\_CFG\_DAT,CFile::modeRead|CFile::shareDenyNone))

{

// 由文件得到app配置map

getfile2map(&t\_csf,&m\_mapappcfg);

m\_mapappcfg[\_T("apppath")]=getappdir();

t\_csf.Close();

}

else

{

AfxMessageBox(\_T("配置文件读取失败"));

assert(0);

}

m\_csmodecomm=m\_mapappcfg[\_T("apppath")]+MODE\_COMM\_DAT;

m\_cslogpath=m\_mapappcfg[\_T("apppath")]+LOG\_PATH;

return 0;

}

// 获取应用程序路径,并转换成目录

CString CtewDlg::getappdir()

{

CString t\_apppath;

GetModuleFileName(NULL,t\_apppath.GetBuffer(\_MAX\_PATH),\_MAX\_PATH);

t\_apppath.ReleaseBuffer();

return getdirfrompath(t\_apppath);

}

// 获取mapdutmessage

int CtewDlg::getdatafile2mapdutmessage()

{

m\_mapdutmessage.clear();

CString t\_csfile=m\_mapappcfg[\_T("apppath")]+DUT\_MSG\_FILE;

CStdioFile t\_csf;

if(t\_csf.Open(t\_csfile,CFile::modeRead|CFile::shareDenyNone))

{

// 由文件得到app配置map

getfile2map(&t\_csf,&m\_mapdutmessage);

m\_mapdutmessage[\_T("000Date")]=m\_csstarttime;

t\_csf.Close();

}

else

{

AfxMessageBox(\_T("生产测试模板读取失败"));

assert(0);

return TRUE;

}

return 0;

}

// 从路径获取目录

CString CtewDlg::getdirfrompath(CString v\_cspath)

{

return v\_cspath.Left(v\_cspath.ReverseFind(\_T('\\'))+1);

}

// 从路径获取文件

CString CtewDlg::getfilefrompath(CString v\_cspath)

{

return v\_cspath.Right(v\_cspath.GetLength()-v\_cspath.ReverseFind(\_T('\\'))-1);

}

// get dlg 界面控件间隙

int CtewDlg::getdlgctrlscale()

{

m\_pointns1.SetPoint(m\_ctrlpositionarray[0][2]-EDGEWIDTH,m\_ctrlpositionarray[0][0]+TOOLBARWIDTH+EDGEWIDTH);

m\_pointns2.SetPoint(m\_ctrlpositionarray[2][0]+EDGEWIDTH,m\_ctrlpositionarray[2][3]-STATUSBARWIDTH-EDGEWIDTH);

m\_pointwe1.SetPoint(m\_ctrlpositionarray[1][0]+EDGEWIDTH,m\_ctrlpositionarray[1][3]-EDGEWIDTH);

m\_pointwe2.SetPoint(m\_ctrlpositionarray[2][2]-EDGEWIDTH,m\_ctrlpositionarray[2][1]+EDGEWIDTH);

return 0;

}

// resize后记录size to map

int CtewDlg::getsize2mapappcfg()

{

CRect t\_rect;

GetWindowRect(&t\_rect);

m\_mapappcfg[\_T("lastx")].Empty();

m\_mapappcfg[\_T("lastx")].Format(\_T("%d"),t\_rect.Width());

m\_mapappcfg[\_T("lasty")].Empty();

m\_mapappcfg[\_T("lasty")].Format(\_T("%d"),t\_rect.Height());

return 0;

}

// 替换字符串中变量符号,"$\_"开头

// return TRUE:exist

// return FALSE:none

int CtewDlg::getstring4replacelabel(CString \*v\_line)

{

int t\_i;

CString t\_cslabel,t\_cslabelid;

t\_i=v\_line->Find(REPLACE\_LABEL);

if(t\_i!=-1)

{

int t\_j;

// 发现替换符号

t\_cslabelid=REPLACE\_LABEL;

t\_j=t\_cslabelid.GetLength();

t\_cslabel=v\_line->Right(v\_line->GetLength()-t\_i-t\_j);

t\_j=t\_cslabel.GetLength();

for(t\_i=0;t\_i<t\_j;t\_i++)

{

if(!(((t\_cslabel[t\_i]>=\_T('0'))&&(t\_cslabel[t\_i]<=\_T('9')))||\

((t\_cslabel[t\_i]>=\_T('a'))&&(t\_cslabel[t\_i]<=\_T('z')))||\

((t\_cslabel[t\_i]>=\_T('A'))&&(t\_cslabel[t\_i]<=\_T('Z')))))

{

break;

}

}

t\_cslabel=t\_cslabel.Left(t\_i);

if(!(m\_mapdutmessage.find(t\_cslabel)!=m\_mapdutmessage.end()))

{

CString t\_cs;

t\_cs.Format(\_T("配置文件存在未定义的替换标志(%s),请检查"),t\_cslabel);

AfxMessageBox(t\_cs);

}

else

{

v\_line->Replace(REPLACE\_LABEL+t\_cslabel,m\_mapdutmessage[t\_cslabel]);

}

return TRUE;

}

return FALSE;

}

// 获取时间 v\_type:0-format;1-dir used

CString CtewDlg::getsystemdatetime(int v\_type)

{

CString t\_cstime;

SYSTEMTIME t\_sysTime;

GetLocalTime(&t\_sysTime);

if(v\_type==0)

{

t\_cstime.Format( \_T("%04ld-%02ld-%02ld %02ld:%02ld:%02ld.%03ld"),t\_sysTime.wYear,t\_sysTime.wMonth,t\_sysTime.wDay,t\_sysTime.wHour,t\_sysTime.wMinute,t\_sysTime.wSecond,t\_sysTime.wMilliseconds);

}

else if(v\_type==1)

{

t\_cstime.Format( \_T("%04ld%02ld%02ld%02ld%02ld%02ld%03ld"),t\_sysTime.wYear,t\_sysTime.wMonth,t\_sysTime.wDay,t\_sysTime.wHour,t\_sysTime.wMinute,t\_sysTime.wSecond,t\_sysTime.wMilliseconds);

}

else

{

assert(0);

}

return t\_cstime;

}

// 分析目录是否存在userdefine file,return 1 means yes,0 means no

int CtewDlg::getuserdefinefilefromdir(CString v\_csdir)

{

CFileFind t\_f;

CString t\_cs;

BOOL t\_bcfg=t\_f.FindFile(v\_csdir+\_T("\\\*.cfg"));

int t\_i=0;

while(t\_bcfg)

{

t\_bcfg=t\_f.FindNextFile();

t\_cs=t\_f.GetFilePath();

// the first cfg file

if(t\_i==0)

{

m\_mapappcfg[\_T("lastfile")]=getfilefrompath(t\_cs);

m\_mapappcfg[\_T("lastpath")]=getdirfrompath(t\_cs);

}

t\_i++;

}

t\_f.Close();

return t\_i;

}

// 解析用户define目录

int CtewDlg::getuserdefinefromdir(CString v\_csdir)

{

CString t\_cs=\_T("open ")+v\_csdir;

this->SendMessage(WM\_appdlgctrladdloglist,(WPARAM)t\_cs.GetBuffer(0),0);

t\_cs.ReleaseBuffer();

// 分析目录是否存在userdefine file,return 1 means yes,0 means no

if(getuserdefinefilefromdir(v\_csdir))

{

// 生成cfg tree

getuserdefinecfgtree();

}

else

{

// clear lastfile item

m\_mapappcfg[\_T("lastfile")]=\_T("");

HTREEITEM t\_node;

// set用户define root node

struct treeitemdata \*t\_pid=reinterpret\_cast<treeitemdata\*>(getuserdefinedefaultitemdata());

t\_pid->path+=\_T("\\");

t\_pid->comment=t\_pid->path;

// 创建根节点

t\_node=m\_cfgtree.InsertItem(t\_pid->comment,t\_pid->runstatus,t\_pid->runstatus);

// 配置节点携带信息,指针引用

t\_pid->node=t\_node;

m\_cfgtree.SetItemData(t\_node,(DWORD)t\_pid);

// 递归建立目录配置tree

setuserdefinetree4path(v\_csdir,t\_pid);

// 展开树控件,选择root node

setuserdefinetreeitem(m\_cfgtree.GetRootItem(),1);

m\_cfgtree.SelectItem(m\_cfgtree.GetRootItem());

}

// 清空cfgtree和tabctrl

this->SendMessage(ID\_dlgstop,NULL,NULL);

// menu open

setcmdinopen();

return 0;

}

// 递归建立目录配置tree

int CtewDlg::setuserdefinetree4path(CString v\_cspath,treeitemdata \*v\_pid)

{

CFileFind t\_f;

CString t\_cs;

BOOL t\_bcfg=t\_f.FindFile(v\_cspath+\_T("\\\*.cfg"));

int t\_i=0;

while(t\_bcfg)

{

t\_bcfg=t\_f.FindNextFile();

t\_cs=t\_f.GetFilePath();

t\_i++;

t\_cs.MakeLower();

// 存在cfg文件,取第一个cfg文件

if(t\_i==1)

{

CStdioFile t\_csf;

t\_csf.Open(t\_cs,CFile::modeRead|CFile::shareDenyNone);

t\_csf.SeekToBegin();

getuserdefinebyparseoldfile(&t\_csf,v\_pid);

t\_csf.Close();

}

}

if(t\_i>1)

{

AfxMessageBox(v\_cspath+\_T(" 有多个cfg文件"));

}

// 未发现cfg文件,进入子目录

if(t\_i==0)

{

// 清理,准备遍历目录

t\_f.Close();

t\_bcfg=t\_f.FindFile(v\_cspath+\_T("\\\*.\*"));

// 遍历目录

while(t\_bcfg)

{

t\_bcfg=t\_f.FindNextFile();

t\_cs=t\_f.GetFilePath();

// 判断目录非'.'以及是目录和非zlog目录

if((!t\_f.IsDots())&&(t\_f.IsDirectory())&&(t\_cs.Find(\_T("zlog"))==-1))

{

struct treeitemdata \*t\_pid=new treeitemdata;

\*t\_pid=\*v\_pid;

t\_pid->level=v\_pid->level+1;

t\_pid->comment=t\_cs;

// 创建节点

HTREEITEM t\_node;

t\_node=m\_cfgtree.InsertItem(t\_pid->comment,t\_pid->runstatus,t\_pid->runstatus,v\_pid->node,TVI\_LAST);

t\_pid->path=t\_cs+\_T("\\");

// 配置节点携带信息,指针引用

t\_pid->node=t\_node;

m\_cfgtree.SetItemData(t\_node,(DWORD)t\_pid);

// 递归遍历

setuserdefinetree4path(t\_cs,t\_pid);

}

}

t\_f.Close();

}

return 0;

}

// parse userdefine with old file

int CtewDlg::getuserdefinebyparseoldfile(CStdioFile \*v\_csf,treeitemdata \*v\_pid)

{

CString t\_line,t\_cs;

for(;(v\_csf->ReadString(t\_line))!=NULL;)

{

// 忽略无用信息

t\_line.TrimLeft();

t\_line.TrimRight();

if(getuserdefinelinebydelunuseinfo(t\_line)==1)

{

continue;

}

// t\_line末尾补';'

if(t\_line[t\_line.GetLength()]!=\_T(';'))

{

t\_line+=\_T(";");

}

// 创建节点

HTREEITEM t\_node;

struct treeitemdata \*t\_pid=new treeitemdata;

\*t\_pid=\*v\_pid;

t\_pid->level=v\_pid->level+1;

// 分析行参数

// filename

t\_cs=t\_line.Left(t\_line.Find(\_T(";")));

t\_pid->filename=t\_cs;

t\_pid->comment=t\_cs;

t\_line=t\_line.Right(t\_line.GetLength()-1-t\_line.Find(\_T(";")));

// host

t\_cs=t\_line.Left(t\_line.Find(\_T(";")));

t\_pid->host=t\_cs;

t\_line=t\_line.Right(t\_line.GetLength()-1-t\_line.Find(\_T(";")));

// port

t\_cs=t\_line.Left(t\_line.Find(\_T(";")));

if(t\_cs.Find(SSHV2DEF)!=-1)

{

t\_pid->sshv2=TRUE;

t\_cs.Replace(SSHV2DEF,\_T(""));

}

else

{

t\_pid->sshv2=FALSE;

}

t\_pid->port=\_ttoi(t\_cs);

assert(t\_pid->port);

t\_node=m\_cfgtree.InsertItem(t\_pid->comment,t\_pid->runstatus,t\_pid->runstatus,v\_pid->node,TVI\_LAST);

// 配置节点携带信息,指针引用

t\_pid->node=t\_node;

m\_cfgtree.SetItemData(t\_node,(DWORD)t\_pid);

}

return 0;

}

// 解析用户define文件

int CtewDlg::getuserdefinefromfile(CString v\_csdir,CString v\_csfile)

{

CString t\_cs=\_T("open ")+v\_csdir+v\_csfile;

this->SendMessage(WM\_appdlgctrladdloglist,(WPARAM)t\_cs.GetBuffer(0),0);

t\_cs.ReleaseBuffer();

t\_cs=v\_csdir+v\_csfile;

// SetWindowText:设置标题

SetWindowText(\_T("tew---")+t\_cs);

CStdioFile t\_csf;

if(t\_csf.Open(t\_cs,CFile::modeRead|CFile::shareDenyNone))

{

// parse userdefine

getuserdefinebyparsefile(&t\_csf,0);

t\_csf.Close();

}

else

{

AfxMessageBox(\_T("sorry,no find this file"));

}

return 0;

}

// parse userdefine with file

int CtewDlg::getuserdefinebyparsefile(CStdioFile \*v\_pcsf,HTREEITEM v\_fathernode)

{

CString t\_line;

HTREEITEM t\_node;

v\_pcsf->SeekToBegin();

t\_node=v\_fathernode;

for(;(v\_pcsf->ReadString(t\_line))!=NULL;)

{

// 忽略无用信息

t\_line.TrimLeft();

t\_line.TrimRight();

if(getuserdefinelinebydelunuseinfo(t\_line)==1)

{

continue;

}

// 替换字符串中变量符号,"$\_"开头

getstring4replacelabel(&t\_line);

t\_node=getuserdefinebyparseline(t\_line,t\_node);

}

// 展开树控件,选择root node

setuserdefinetreeitem(m\_cfgtree.GetRootItem(),1);

m\_cfgtree.SelectItem(m\_cfgtree.GetRootItem());

// 清空cfgtree和tabctrl

this->SendMessage(ID\_dlgstop,NULL,NULL);

return 0;

}

// parse userdefine with line

HTREEITEM CtewDlg::getuserdefinebyparseline(CString v\_csline,HTREEITEM v\_fathernode)

{

struct treeitemdata \*t\_ptreeitemdata;

// 分析携带信息

if(v\_fathernode==0)

{

t\_ptreeitemdata=reinterpret\_cast<treeitemdata\*>(getuserdefinedefaultitemdata());

t\_ptreeitemdata->level+=1;

}

else

{

struct treeitemdata \*t\_ptemp;

t\_ptreeitemdata=new treeitemdata;

t\_ptemp=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(v\_fathernode));

\*t\_ptreeitemdata=\*t\_ptemp;

// 判断当前节点信息是否包含文件名

if(t\_ptemp->filename.GetLength()==0)

{

// 确定下一节点level+1

t\_ptreeitemdata->level+=1;

}

else

{

// 判断是否需要更新终端节点携带信息

if(getuserdefinenodetype(v\_csline))

{

HTREEITEM t\_node;

t\_node=m\_cfgtree.GetParentItem(v\_fathernode);

assert(t\_node);

t\_node=m\_cfgtree.GetParentItem(t\_node);

if(t\_node!=NULL)

{

// 获取父2层节点携带信息

t\_ptemp=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(t\_node));

\*t\_ptreeitemdata=\*t\_ptemp;

// 确定下一节点level+2

t\_ptreeitemdata->level+=2;

}

}

}

}

// 根据配置行内容更新携带信息

getuserdefineitemdata(v\_csline,t\_ptreeitemdata);

return setuserdefinetreenode(v\_fathernode,t\_ptreeitemdata);

}

// get node type by parse line;1 to has child;0 not

int CtewDlg::getuserdefinenodetype(CString v\_csline)

{

if(v\_csline.Find(\_T('{'))!=0)

{

// not has child

return 0;

}

else

{

// has child

return 1;

}

}

// 从path获取id list

LPITEMIDLIST CtewDlg::GetItemIDListFromFilePath(CString v\_strFilePath)

{

CString t\_strFilePath=v\_strFilePath;

if(t\_strFilePath.GetLength()==0)

{

return NULL;

}

// 得到桌面的目录

LPSHELLFOLDER t\_pDesktopFolder=NULL;

HRESULT t\_hr=SHGetDesktopFolder(&t\_pDesktopFolder);

if(FAILED(t\_hr))

{

return NULL;

}

LPITEMIDLIST t\_pItemIDList=NULL;

t\_hr=t\_pDesktopFolder->ParseDisplayName(NULL,NULL,t\_strFilePath.GetBuffer(0),NULL,&t\_pItemIDList,NULL);

t\_strFilePath.ReleaseBuffer();

t\_pDesktopFolder->Release();

if(FAILED(t\_hr))

{

return NULL;

}

return t\_pItemIDList;

}

// parse itemdata to get user define line

CString CtewDlg::getuserdefineline(HTREEITEM v\_node)

{

CString t\_csline,t\_cstemp;

struct treeitemdata \*t\_itemdata;

assert(v\_node);

t\_itemdata=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(v\_node));

assert(t\_itemdata);

if(t\_itemdata->filename.GetLength()==0)

{

// has child

// {level;path;comment;runmtmode;enableflag;message}

t\_csline=\_T("{");

// level

t\_cstemp.Format(\_T("%d"),t\_itemdata->level);

t\_csline+=t\_cstemp+CONFIGDELIM;

// path

if(m\_cfgtree.GetParentItem(v\_node)==0)

{

// root node

t\_csline+=t\_itemdata->path+CONFIGDELIM;

}

else

{

// not root node

t\_cstemp=t\_itemdata->path.Right(t\_itemdata->path.GetLength()-(reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(m\_cfgtree.GetParentItem(v\_node))))->path.GetLength());

t\_csline+=t\_cstemp+CONFIGDELIM;

}

// comment

t\_csline+=t\_itemdata->comment+CONFIGDELIM;

// runmtmode

t\_cstemp.Format(\_T("%d"),t\_itemdata->runmtmode);

t\_csline+=t\_cstemp+CONFIGDELIM;

// enableflag

t\_cstemp.Format(\_T("%d"),t\_itemdata->enableflag);

t\_csline+=t\_cstemp+CONFIGDELIM;

// message

t\_csline+=t\_itemdata->message+CONFIGDELIM;

t\_csline+=\_T("}\n");

}

else

{

// not has child

// filename;host;[sshv2:]port;comment;runmtmode;enableflag;message;

t\_csline=\_T("\t");

// filename

t\_csline+=t\_itemdata->filename+CONFIGDELIM;

// host

t\_csline+=t\_itemdata->host+CONFIGDELIM;

// port

if(t\_itemdata->sshv2)

{

t\_cstemp.Format(\_T("sshv2:%d"),t\_itemdata->port);

}

else

{

t\_cstemp.Format(\_T("%d"),t\_itemdata->port);

}

t\_csline+=t\_cstemp+CONFIGDELIM;

// comment

t\_csline+=t\_itemdata->comment+CONFIGDELIM;

// runmtmode

t\_cstemp.Format(\_T("%d"),t\_itemdata->runmtmode);

t\_csline+=t\_cstemp+CONFIGDELIM;

// enableflag

t\_cstemp.Format(\_T("%d"),t\_itemdata->enableflag);

t\_csline+=t\_cstemp+CONFIGDELIM;

// message

t\_csline+=t\_itemdata->message+\_T(";\n");

}

return t\_csline;

}

// parse line to get user define itemdata

int CtewDlg::getuserdefineitemdata(CString v\_csline,treeitemdata \*v\_ptreeitemdata)

{

if(v\_csline.Find(\_T('{'))!=0)

{

// 获取文件名,';'结束

v\_ptreeitemdata->filename=v\_csline.Left(v\_csline.Find(CONFIGDELIM));

v\_csline=v\_csline.Right(v\_csline.GetLength()-1-v\_csline.Find(CONFIGDELIM));

// 获取主机名,';'结束

v\_ptreeitemdata->host=v\_csline.Left(v\_csline.Find(CONFIGDELIM));

v\_csline=v\_csline.Right(v\_csline.GetLength()-1-v\_csline.Find(CONFIGDELIM));

// 获取端口,';'结束

CString t\_csport=v\_csline.Left(v\_csline.Find(CONFIGDELIM));

if(t\_csport.Find(SSHV2DEF)!=-1)

{

v\_ptreeitemdata->sshv2=TRUE;

t\_csport.Replace(SSHV2DEF,\_T(""));

}

else

{

v\_ptreeitemdata->sshv2=FALSE;

}

v\_ptreeitemdata->port=\_ttoi(t\_csport);

// 获取标题,';'结束

v\_csline=v\_csline.Right(v\_csline.GetLength()-1-v\_csline.Find(CONFIGDELIM));

if(v\_csline.Find(CONFIGDELIM)>0)

{

v\_ptreeitemdata->comment=v\_csline.Left(v\_csline.Find(CONFIGDELIM));

}

else

{

v\_ptreeitemdata->comment=v\_ptreeitemdata->filename;

}

// 获取runmtmode,';'结束

if(v\_csline.Find(CONFIGDELIM)!=-1)

{

v\_csline=v\_csline.Right(v\_csline.GetLength()-1-v\_csline.Find(CONFIGDELIM));

if(((v\_csline.Left(v\_csline.Find(CONFIGDELIM))).GetLength())!=0)

{

v\_ptreeitemdata->runmtmode=\_ttoi(v\_csline.Left(v\_csline.Find(CONFIGDELIM)));

}

}

// 获取enableflag,';'结束

if(v\_csline.Find(CONFIGDELIM)!=-1)

{

v\_csline=v\_csline.Right(v\_csline.GetLength()-1-v\_csline.Find(CONFIGDELIM));

if(((v\_csline.Left(v\_csline.Find(CONFIGDELIM))).GetLength())!=0)

{

v\_ptreeitemdata->enableflag=\_ttoi(v\_csline.Left(v\_csline.Find(CONFIGDELIM)));

}

}

if(v\_ptreeitemdata->enableflag==0)

{

v\_ptreeitemdata->runstatus=icondisable;

}

else

{

v\_ptreeitemdata->runstatus=iconready;

}

// 获取信息,';'结束

v\_csline=v\_csline.Right(v\_csline.GetLength()-1-v\_csline.Find(CONFIGDELIM));

if(v\_csline.Find(CONFIGDELIM)>=0)

{

v\_ptreeitemdata->message=v\_csline.Left(v\_csline.Find(CONFIGDELIM));

}

else

{

v\_ptreeitemdata->message=\_T('');

}

}

else

{

// 去除左边'{'

v\_csline=v\_csline.Right(v\_csline.GetLength()-1);

// 获取层次,';'结束

v\_ptreeitemdata->level=\_ttoi(v\_csline.Left(v\_csline.Find(CONFIGDELIM)));

// 获取路径,';'结束

v\_csline=v\_csline.Right(v\_csline.GetLength()-1-v\_csline.Find(CONFIGDELIM));

// level=0,取打开路径值

if(v\_ptreeitemdata->level!=0)

{

v\_ptreeitemdata->path+=(v\_csline.Left(v\_csline.Find(CONFIGDELIM)));

}

else

{

v\_ptreeitemdata->path=m\_mapappcfg[\_T("lastpath")];

}

// 获取comment,';'结束

if(v\_csline.Find(CONFIGDELIM)!=-1)

{

v\_csline=v\_csline.Right(v\_csline.GetLength()-1-v\_csline.Find(CONFIGDELIM));

if(((v\_csline.Left(v\_csline.Find(CONFIGDELIM))).GetLength())!=0)

{

v\_ptreeitemdata->comment=(v\_csline.Left(v\_csline.Find(CONFIGDELIM)));

if(v\_ptreeitemdata->comment.GetLength()==0)

{

v\_ptreeitemdata->comment=v\_ptreeitemdata->path;

}

}

}

// 获取runmtmode,';'结束

if(v\_csline.Find(CONFIGDELIM)!=-1)

{

v\_csline=v\_csline.Right(v\_csline.GetLength()-1-v\_csline.Find(CONFIGDELIM));

if(((v\_csline.Left(v\_csline.Find(CONFIGDELIM))).GetLength())!=0)

{

v\_ptreeitemdata->runmtmode=\_ttoi(v\_csline.Left(v\_csline.Find(CONFIGDELIM)));

}

}

// 获取enableflag,';'结束

if(v\_csline.Find(CONFIGDELIM)!=-1)

{

v\_csline=v\_csline.Right(v\_csline.GetLength()-1-v\_csline.Find(CONFIGDELIM));

if(((v\_csline.Left(v\_csline.Find(CONFIGDELIM))).GetLength())!=0)

{

v\_ptreeitemdata->enableflag=\_ttoi(v\_csline.Left(v\_csline.Find(CONFIGDELIM)));

}

}

if(v\_ptreeitemdata->enableflag==0)

{

v\_ptreeitemdata->runstatus=icondisable;

}

else

{

v\_ptreeitemdata->runstatus=iconready;

}

// 获取message,';'结束

if(v\_csline.Find(CONFIGDELIM)!=-1)

{

v\_csline=v\_csline.Right(v\_csline.GetLength()-1-v\_csline.Find(CONFIGDELIM));

if(((v\_csline.Left(v\_csline.Find(CONFIGDELIM))).GetLength())!=0)

{

v\_ptreeitemdata->message=(v\_csline.Left(v\_csline.Find(CONFIGDELIM)));

}

}

// clear file/host/port

v\_ptreeitemdata->filename=v\_ptreeitemdata->host=\_T("");

v\_ptreeitemdata->port=0;

v\_ptreeitemdata->sshv2=FALSE;

}

return 0;

}

// 文件内容忽略无用信息

int CtewDlg::getuserdefinelinebydelunuseinfo(CString v\_line)

{

// 忽略空行和注释行

if((v\_line.GetLength()==0)||(v\_line.Find(DEF\_COMMENT)==0))

{

return 1;

}

return 0;

}

// 生成cfg tree

int CtewDlg::getuserdefinecfgtree()

{

// 初始化dut message

if(getproducttest())

{

m\_mapdutmessage.clear();

// 获取mapdutmessage

if(getdatafile2mapdutmessage())

{

assert(0);

}

// 准备生产测试场景

cdlgdutmsg t\_dlgdutmsg;

t\_dlgdutmsg.m\_ptewdlg=this;

int t\_nResponse=t\_dlgdutmsg.DoModal();

if(t\_nResponse==IDOK)

{

CString t\_csfile=m\_mapappcfg[\_T("apppath")]+DUT\_MSG\_FILE;

CStdioFile t\_csf;

// 防止日期表项被修改

m\_mapdutmessage[\_T("000Date")]=m\_csstarttime;

if(t\_csf.Open(t\_csfile,CFile::modeWrite|CFile::shareDenyNone|CFile::modeCreate))

{

getmap2file(&t\_csf,&m\_mapdutmessage);

t\_csf.Close();

}

else

{

AfxMessageBox(\_T("生产测试模板写入失败"));

assert(0);

}

}

}

// 清空树控件node

setuserdefinetreeempty();

// create tree

getuserdefinefromfile(m\_mapappcfg[\_T("lastpath")],m\_mapappcfg[\_T("lastfile")]);

return 0;

}

// get default itemdata

int CtewDlg::getuserdefinedefaultitemdata()

{

treeitemdata \*t\_pitemdata=new treeitemdata;

t\_pitemdata->level=0;

t\_pitemdata->path=m\_mapappcfg[\_T("lastpath")];

if((m\_mapappcfg[\_T("lastfile")]).GetLength()==0)

{

t\_pitemdata->comment=m\_mapappcfg[\_T("lastpath")];

}

else

{

t\_pitemdata->comment=m\_mapappcfg[\_T("lastfile")];

}

t\_pitemdata->runmtmode=0;

t\_pitemdata->enableflag=1;

t\_pitemdata->exitflag=0;

t\_pitemdata->runstatus=0;

t\_pitemdata->filename=\_T("");

t\_pitemdata->host=\_T("");

t\_pitemdata->port=0;

t\_pitemdata->sshv2=FALSE;

t\_pitemdata->message=\_T("");

t\_pitemdata->ptrdlg=0;

t\_pitemdata->trmtabctrlnumber=-1;

t\_pitemdata->loopcounter=0;

t\_pitemdata->faultcounter=0;

t\_pitemdata->logpath=m\_cslogpath;

return (int)t\_pitemdata;

}

// get file version

CString CtewDlg::getfileversion()

{

CString t\_csversion;

CString t\_apppath;

DWORD t\_dwHandle;

VS\_FIXEDFILEINFO \*t\_pVi;

GetModuleFileName(NULL,t\_apppath.GetBuffer(\_MAX\_PATH),\_MAX\_PATH);

t\_apppath.ReleaseBuffer();

int t\_size=GetFileVersionInfoSize(t\_apppath,&t\_dwHandle);

if(t\_size>0)

{

wchar\_t \*t\_pbuffer=new wchar\_t[t\_size];

if(GetFileVersionInfo(t\_apppath,t\_dwHandle,t\_size,t\_pbuffer))

{

if(VerQueryValue(t\_pbuffer,\_T("\\"),(LPVOID\*)&t\_pVi,(PUINT)&t\_size))

{

int t\_ivmsh,t\_ivmsl,t\_ivlsh,t\_ivlsl;

m\_dlocalver.vmah=t\_ivmsh=HIWORD(t\_pVi->dwFileVersionMS);

m\_dlocalver.vmal=t\_ivmsl=LOWORD(t\_pVi->dwFileVersionMS);

m\_dlocalver.vlah=t\_ivlsh=HIWORD(t\_pVi->dwFileVersionLS);

m\_dlocalver.vlal=t\_ivlsl=LOWORD(t\_pVi->dwFileVersionLS);

t\_csversion.Format(\_T("%d.%d.%d.%d"),t\_ivmsh,t\_ivmsl,t\_ivlsh,t\_ivlsl);

}

}

delete[] t\_pbuffer;

t\_pbuffer=NULL;

}

return t\_csversion;

}

// 由配置map得到文件

int CtewDlg::getmap2file(CStdioFile \*v\_pcsf,map<CString,CString> \*v\_map)

{

map<CString,CString>::iterator t\_it;

for(t\_it=(\*v\_map).begin();t\_it!=(\*v\_map).end();++t\_it)

{

v\_pcsf->WriteString(t\_it->first+\_T("=")+t\_it->second+\_T("\n"));

}

return 0;

}

// 由文件得到app配置map

int CtewDlg::getfile2map(CStdioFile \*v\_pcsf,map<CString,CString> \*v\_map)

{

CString t\_line,t\_cs1,t\_cs2;

v\_pcsf->SeekToBegin();

for(;(v\_pcsf->ReadString(t\_line)!=NULL);)

{

int t\_i=0;

//删除行末注释部分

t\_i=t\_line.Find(DEF\_COMMENT);

if(t\_i>0)

{

t\_line.Left(t\_i);

}

//跳过注释和空行

if((t\_i==0)||(t\_line.GetLength()==0))

{

continue;

}

t\_i=t\_line.Find(\_T("="));

if(t\_i==-1)

{

AfxMessageBox(\_T("app\_cfg.dat\n格式错误:")+t\_line);

continue;

}

//去除空格/换行/回车/tab,""之间空格不去除

t\_cs1=t\_line.Left(t\_i);

t\_cs1.TrimLeft();

t\_cs1.TrimRight();

t\_cs2=t\_line.Right(t\_line.GetLength()-t\_i-1);

t\_cs2.TrimLeft();

t\_cs2.TrimRight();

t\_cs2.Replace(CONFIGDELIM,\_T(""));

(\*v\_map)[t\_cs1]=t\_cs2;

}

return 0;

}

// app配置map保存到文件

int CtewDlg::getcfgmapappcfg2file()

{

CString t\_csarray[]=MAPAPPCFG;

CStdioFile t\_csf;

if(t\_csf.Open(getappdir()+APP\_CFG\_DAT,CFile::modeWrite|CFile::modeCreate|CFile::shareDenyNone))

{

int t\_i;

for(t\_i=sizeof(t\_csarray)/sizeof(\*t\_csarray);t\_i>0;)

{

t\_i--;

t\_csf.WriteString(t\_csarray[t\_i]+\_T("=")+m\_mapappcfg[t\_csarray[t\_i]]+\_T(";\n"));

}

t\_csf.Close();

}

else

{

AfxMessageBox(\_T("配置文件写入失败"));

assert(0);

}

return 0;

}

// 初始化对话框界面

int CtewDlg::initdlgview()

{

// 初始化cfg trm log 控件布局

float t\_ctrlscalearray[3][4]=DLGCTRLSIZEARRAY;

int t\_x=\_ttoi(m\_mapappcfg[\_T("lastx")]);

int t\_y=\_ttoi(m\_mapappcfg[\_T("lasty")]);

for(int t\_i=0;t\_i<3;t\_i++)

{

m\_ctrlpositionarray[t\_i][0]=(int)(t\_ctrlscalearray[t\_i][0]\*(t\_x-8));

m\_ctrlpositionarray[t\_i][1]=(int)(t\_ctrlscalearray[t\_i][1]\*(t\_y-46));

m\_ctrlpositionarray[t\_i][2]=(int)(t\_ctrlscalearray[t\_i][2]\*(t\_x-8));

m\_ctrlpositionarray[t\_i][3]=(int)(t\_ctrlscalearray[t\_i][3]\*(t\_y-46));

}

//设置初始化窗口size为上次

MoveWindow(CRect(0,0,t\_x,t\_y));

// 增加菜单栏

initdlgviewmenu();

// 增加工具栏

initdlgviewtoolsbar();

// 添加状态栏,V\_i为1创建,0为重绘

initdlgviewstatusbar(TRUE);

// 初始化cfgtree ctrl

initctrlcfgtree();

// 初始化trm tab ctrl

initctrltrmtab();

// 初始化log list ctrl

initctrlloglist();

return 0;

}

// 增加菜单栏

int CtewDlg::initdlgviewmenu()

{

// dlg菜单句柄

HMENU t\_hMenu;

// 导入资源,创建菜单

t\_hMenu=LoadMenu(AfxGetInstanceHandle(),MAKEINTRESOURCE(IDR\_dlgmenu));

// 添加到对话框

::SetMenu(this->GetSafeHwnd(),t\_hMenu);

// get菜单栏的指针

m\_pmenu=CMenu::FromHandle(t\_hMenu);

return 0;

}

// 增加工具栏

int CtewDlg::initdlgviewtoolsbar()

{

// imagelist大小为32\*32,颜色为32位

m\_toolbarimagelist.Create(32,32,ILC\_COLOR32|ILC\_MASK,TOOLBARBUTTONCOUNTER,TOOLBARBUTTONCOUNTER);

CBitmap t\_bmp;

UINT t\_bmparray[TOOLBARBUTTONCOUNTER]=TOOLBARICONARRAY;

UINT t\_idarray[TOOLBARBUTTONCOUNTER]=TOOLBARIDARRAY;

int t\_i;

for(t\_i=0;t\_i<TOOLBARBUTTONCOUNTER;t\_i++)

{

// 使用现有位图为bmp赋值

t\_bmp.LoadBitmap(t\_bmparray[t\_i]);

// 结合ILC\_MASK,将RGB(255,255,255)也就是白色透明处理

m\_toolbarimagelist.Add(&t\_bmp,RGB(255,255,255));

t\_bmp.DeleteObject();

}

// 创建工具栏控件,CreateEx函数创建的工具栏,默认有浮动按钮属性

m\_toolbar.CreateEx(this);

// 按钮设置ID

m\_toolbar.SetButtons(t\_idarray,TOOLBARBUTTONCOUNTER);

// 函数第一个是按钮大小,第二个是图像大小,按钮必须比图像要大,具体是按钮的要比图像的宽大7或者以上,高6

m\_toolbar.SetSizes(CSize(40,40),CSize(32,32));

// 获得工具条控制指针

m\_ptoolbarctrl=&(m\_toolbar.GetToolBarCtrl());

// 设置图像

m\_ptoolbarctrl->SetImageList(&m\_toolbarimagelist);

// 设置工具栏tip

if(m\_tooltipctrl.Create(this))

{

CRect t\_toolbuttonrect;

for(t\_i=0;t\_i<TOOLBARBUTTONCOUNTER;t\_i++)

{

TBBUTTON t\_btn;

if((m\_ptoolbarctrl->GetButton(t\_i,&t\_btn))&&(t\_btn.fsStyle==TBSTYLE\_BUTTON))

{

m\_ptoolbarctrl->GetItemRect(t\_i,&t\_toolbuttonrect);

m\_tooltipctrl.AddTool(m\_ptoolbarctrl,t\_btn.idCommand,t\_toolbuttonrect,t\_btn.idCommand);

}

}

m\_ptoolbarctrl->SetToolTips(&m\_tooltipctrl);

}

if(NULL!=m\_ptoolbarctrl->GetToolTips())

{

m\_ptoolbarctrl->GetToolTips()->UpdateTipText(\_T("打开文件"),m\_ptoolbarctrl,t\_idarray[0]);

m\_ptoolbarctrl->GetToolTips()->UpdateTipText(\_T("打开目录"),m\_ptoolbarctrl,t\_idarray[1]);

m\_ptoolbarctrl->GetToolTips()->UpdateTipText(\_T("开始执行"),m\_ptoolbarctrl,t\_idarray[2]);

m\_ptoolbarctrl->GetToolTips()->UpdateTipText(\_T("暂停执行"),m\_ptoolbarctrl,t\_idarray[3]);

m\_ptoolbarctrl->GetToolTips()->UpdateTipText(\_T("停止执行"),m\_ptoolbarctrl,t\_idarray[4]);

m\_ptoolbarctrl->GetToolTips()->UpdateTipText(\_T("帮助文档"),m\_ptoolbarctrl,t\_idarray[5]);

}

// 显示工具栏

repainttoolstatus(0);

return 0;

}

// 添加状态栏,v\_i为1创建,0为重绘

int CtewDlg::initdlgviewstatusbar(int v\_i)

{

// 状态栏分割数量和比例

CRect rect;

GetClientRect(&rect);

int t\_iarray[STATUSBARSEPARATES];

for(int t\_i=0;t\_i<STATUSBARSEPARATES;t\_i++)

{

t\_iarray[t\_i]=rect.Width()\*(t\_i+1)/STATUSBARSEPARATES;

}

t\_iarray[STATUSBARSEPARATES-1]=-1;

if(v\_i)

{

m\_statusbarctrl.Create(WS\_CHILD|WS\_VISIBLE|SBARS\_TOOLTIPS|CCS\_BOTTOM,rect,this,ID\_dlgstatusbarctrl);

}

m\_statusbarctrl.SetParts(STATUSBARSEPARATES,t\_iarray);

//重绘状态栏

repainttoolstatus(ID\_dlgstatusbarctrl);

return 0;

}

// 重绘控件

// 重绘ctrl log list

int CtewDlg::repaintloglist()

{

CRect t\_rect;

// 获取列表视图控件的位置和大小

m\_loglist.GetClientRect(&t\_rect);

// 为列表视图控件添加3列,首项默认左对齐不可修改,因此不配置0列,应用仍按此顺序,but重绘时按0列开始计算

m\_loglist.SetColumnWidth(0,t\_rect.Width()/8);

m\_loglist.SetColumnWidth(1,t\_rect.Width()\*3/8);

m\_loglist.SetColumnWidth(2,t\_rect.Width()/2);

return 0;

}

// v\_id:0表示工具栏,ID\_dlgstatusbarctrl表示状态栏

int CtewDlg::repainttoolstatus(int v\_id)

{

RepositionBars(AFX\_IDW\_CONTROLBAR\_FIRST,AFX\_IDW\_CONTROLBAR\_LAST,v\_id);

return 0;

}

// 重绘各控件

int CtewDlg::repaintctrl(int v\_x,int v\_y)

{

// 忽略最小化

if((v\_x!=0)&&(v\_y!=0))

{

// 窗口改变后重绘控件

int t\_ctrlarray[]=DLGCTRLARRAY;

int t\_i;

CRect t\_rect;

CRect t\_windowsrect;

GetWindowRect(&t\_windowsrect);

float t\_scalex=(float)t\_windowsrect.Width()/(float)(\_ttoi(m\_mapappcfg[\_T("lastx")]));

float t\_scaley=(float)t\_windowsrect.Height()/(float)(\_ttoi(m\_mapappcfg[\_T("lasty")]));

for(t\_i=0;t\_i<3;t\_i++)

{

CWnd \*t\_pWnd;

// 获取控件句柄

t\_pWnd=GetDlgItem(t\_ctrlarray[t\_i]);

// 判断是否为空，因为对话框创建时会调用此函数，而当时控件还未创建

if(t\_pWnd!=NULL)

{

assert((t\_i>=0)&&(t\_i<=3));

m\_ctrlpositionarray[t\_i][0]=(int)((float)m\_ctrlpositionarray[t\_i][0]\*t\_scalex);

m\_ctrlpositionarray[t\_i][1]=(int)((float)m\_ctrlpositionarray[t\_i][1]\*t\_scaley);

m\_ctrlpositionarray[t\_i][2]=(int)((float)m\_ctrlpositionarray[t\_i][2]\*t\_scalex);

m\_ctrlpositionarray[t\_i][3]=(int)((float)m\_ctrlpositionarray[t\_i][3]\*t\_scaley);

switch(t\_i)

{

case 0:

t\_rect.left=EDGEWIDTH;

t\_rect.top=TOOLBARWIDTH+EDGEWIDTH;

t\_rect.right=m\_ctrlpositionarray[t\_i][2]-EDGEWIDTH;

t\_rect.bottom=v\_y-STATUSBARWIDTH-EDGEWIDTH;

break;

case 1:

t\_rect.left=m\_ctrlpositionarray[t\_i][0]+EDGEWIDTH;

t\_rect.top=m\_ctrlpositionarray[t\_i][1]+TOOLBARWIDTH+EDGEWIDTH;

t\_rect.right=v\_x-EDGEWIDTH;

t\_rect.bottom=m\_ctrlpositionarray[t\_i][3]-EDGEWIDTH;

break;

case 2:

t\_rect.left=m\_ctrlpositionarray[t\_i][0]+EDGEWIDTH;

t\_rect.top=m\_ctrlpositionarray[t\_i][1]+EDGEWIDTH;

t\_rect.right=v\_x-EDGEWIDTH;

t\_rect.bottom=v\_y-STATUSBARWIDTH-EDGEWIDTH;

break;

}

// 设置控件大小

t\_pWnd->MoveWindow(&t\_rect);

}

}

// tabctrl关联dlg重绘/repaint ctrl trmdlg

if(m\_pcuritemdata!=NULL)

{

repaintctrltrmdlg(m\_pcuritemdata->node);

}

// dlg改变后重绘log list,包含分隔比例

if(IsWindow(m\_loglist.m\_hWnd))

{

// 重绘log list

repaintloglist();

}

// dlg改变后重绘状态栏,包含分隔比例

if(IsWindow(m\_statusbarctrl.m\_hWnd))

{

// 添加状态栏,v\_i为1创建,0为重绘

initdlgviewstatusbar(FALSE);

}

// get dlg 界面控件间隙

getdlgctrlscale();

}

return 0;

}

// tabctrl关联dlg重绘/repaint ctrl trmdlg

int CtewDlg::repaintctrltrmdlg(HTREEITEM v\_node)

{

struct treeitemdata \*t\_itemdata=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(v\_node));

assert(t\_itemdata);

if(t\_itemdata->ptrdlg!=0)

{

CRect t\_tabdlgrect;

// 终端tab control指针

m\_trmtab.GetClientRect(&t\_tabdlgrect);

t\_tabdlgrect.top+=2;

t\_tabdlgrect.left+=2;

t\_tabdlgrect.bottom-=25;

t\_tabdlgrect.right-=2;

(reinterpret\_cast<ctrmdlg \*>(t\_itemdata->ptrdlg))->m\_rc=t\_tabdlgrect;

(reinterpret\_cast<ctrmdlg \*>(t\_itemdata->ptrdlg))->SendMessage(WM\_trmdlgresize,NULL,NULL);

}

repainttoolstatus(ID\_dlgstatusbarctrl);

return 0;

}

// 重载系统消息处理,拦截onclose消息

BOOL CtewDlg::OnWndMsg(UINT message, WPARAM wParam, LPARAM lParam, LRESULT\* pResult)

{

// TODO: 在此添加专用代码和/或调用基类

if(message==WM\_CLOSE)

{

// 询问是否关闭,发送关闭窗口消息,进入关闭对话框的代码

if(AfxMessageBox(\_T("确认关闭程序?"),MB\_YESNO)==IDYES)

{

// tcp server exit

m\_ptcpserver->PostThreadMessage(WM\_QUIT,0,0);

// udp exit

m\_pudp->PostThreadMessage(WM\_QUIT,0,0);

// app exit

CString t\_cs=\_T("[tew]close");

this->SendMessage(WM\_appdlgctrladdloglist,(WPARAM)t\_cs.GetBuffer(0),0);

t\_cs.ReleaseBuffer();

// 遍历树,dlgstop

setuserdefinetreeitem(m\_cfgtree.GetRootItem(),4);

// app配置map保存到文件

getcfgmapappcfg2file();

// close app log file

m\_csfapplogfile.Close();

// 等待清理完成

Sleep(APPENDDELAY);

return CDialogEx::OnWndMsg(message, wParam, lParam, pResult);

}

return TRUE;

}

else

{

return CDialogEx::OnWndMsg(message, wParam, lParam, pResult);

}

}

// dialog exit

void CtewDlg::Ondlgexit()

{

// TODO: ÔÚ´ËÌí¼ÓÃüÁî´¦Àí³ÌÐò´úÂë

SendMessage(WM\_CLOSE);

}

// dialog size

void CtewDlg::OnSize(UINT nType, int cx, int cy)

{

CDialogEx::OnSize(nType, cx, cy);

// TODO: ÔÚ´Ë´¦Ìí¼ÓÏûÏ¢´¦Àí³ÌÐò´úÂë

if((cx!=0)&&(cy!=0))

{

// dlg改变后重绘各控件,注意会调用mapappcfg

repaintctrl(cx,cy);

// resize后记录size to map

getsize2mapappcfg();

}

}

// dlg最小size X&Y

void CtewDlg::OnGetMinMaxInfo(MINMAXINFO\* lpMMI)

{

// TODO: ÔÚ´ËÌí¼ÓÏûÏ¢´¦Àí³ÌÐò´úÂëºÍ/»òµ÷ÓÃÄ¬ÈÏÖµ

// set dlg size min

lpMMI->ptMinTrackSize.x=DLGMINX;

lpMMI->ptMinTrackSize.y=DLGMINY;

CDialogEx::OnGetMinMaxInfo(lpMMI);

}

// log list menu popup

void CtewDlg::OnNMRClickloglist(NMHDR \*pNMHDR, LRESULT \*pResult)

{

LPNMITEMACTIVATE pNMItemActivate=reinterpret\_cast<LPNMITEMACTIVATE>(pNMHDR);

// TODO: ÔÚ´ËÌí¼Ó¿Ø¼þÍ¨Öª´¦Àí³ÌÐò´úÂë

CMenu menu;

POINT pt={0};

GetCursorPos(&pt);// 得到鼠标点击位置

menu.LoadMenu(IDR\_loglistmenu);// 菜单资源ID

// m\_list是CListCtrl对象,这里的消息发到this

menu.GetSubMenu(0)->TrackPopupMenu(0,pt.x,pt.y,this);

\*pResult=0;

}

// 遍历,exit

int CtewDlg::runtewdlgexit(HTREEITEM v\_node)

{

Sleep(100);

runtewdlgclear(v\_node);

// menu open

setcmdinopen();

return 0;

}

// 遍历,clear

int CtewDlg::runtewdlgclear(HTREEITEM v\_node)

{

// close file

m\_csfcsv.Close();

// 删除标签控件中所有的项

if(m\_trmtab.DeleteAllItems()==0)

{

assert(0);

}

m\_pcuritemdata=0;

return 0;

}

// 准备开始

int CtewDlg::runtewdlgperstart()

{

// 清理报告字段

m\_mapmodulefield.clear();

// 初始化log目录

pathinitlogpath(m\_cslogpath);

// 初始化csv文件

pathinitcsvfile();

// 初始化终端页面数组

inittrmtabpageitemdataarray();

HTREEITEM t\_node;

t\_node=m\_cfgtree.GetRootItem();

// 3-scan script filename

setuserdefinetreeitem(t\_node,3);

// 5-dlgstart icon&text初始化

setuserdefinetreeitem(t\_node,5);

// 6-trm log path init

setuserdefinetreeitem(t\_node,6);

// 遍历,并发

runtewdlgstart(t\_node);

return 0;

}

// 遍历,并行和顺序化兼容,节点信息runmtflag表示本层次串并执行方式

int CtewDlg::runtewdlgstart(HTREEITEM v\_node)

{

int t\_ireturn=0;

if(v\_node!=NULL)

{

struct treeitemdata \*t\_pitemdata;

HTREEITEM t\_hChildItem;

t\_pitemdata=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(v\_node));

if(t\_pitemdata!=NULL)

{

if((t\_pitemdata->enableflag==0)||(t\_pitemdata->exitflag==1))

{

if((t\_pitemdata->enableflag==0)&&(xavietex))

{

CString t\_cs=\_T("disable ")+t\_pitemdata->comment;

this->SendMessage(WM\_appdlgctrladdloglist,(WPARAM)t\_cs.GetBuffer(0),0);

t\_cs.ReleaseBuffer();

}

t\_pitemdata->exitflag=1;

// 访问兄弟节点

t\_hChildItem=m\_cfgtree.GetNextSiblingItem(v\_node);

t\_ireturn=runtewdlgstart(t\_hChildItem);

return t\_ireturn;

}

else

{

CString t\_cs=\_T("start ")+t\_pitemdata->comment;

this->SendMessage(WM\_appdlgctrladdloglist,(WPARAM)t\_cs.GetBuffer(0),0);

t\_cs.ReleaseBuffer();

if((t\_pitemdata->filename.GetLength()!=0)&&(t\_pitemdata->runstatus==iconready))

{

// 使用信号量进行PV操作

WaitForSingleObject(m\_hsemaphore,INFINITE);

// 创建trmdlg

ctrmdlg \*t\_trmdlg=new ctrmdlg;

t\_pitemdata->ptrdlg=(int)t\_trmdlg;

(reinterpret\_cast<ctrmdlg \*>(t\_pitemdata->ptrdlg))->m\_pitemdata=t\_pitemdata;

t\_pitemdata->trmtabctrlnumber=m\_trmtab.GetItemCount();

m\_ptrmtabpageitemdataarray[t\_pitemdata->trmtabctrlnumber]=t\_pitemdata;

m\_trmtab.InsertItem(t\_pitemdata->trmtabctrlnumber,t\_pitemdata->comment);

t\_trmdlg->m\_ptewdlg=this;

// 创建一个非模态对话框

t\_trmdlg->Create(IDD\_trmdlg,&m\_trmtab);

// 刷新当前节点

setcurtreenode(v\_node);

// 终端实例界面重绘

if(m\_pcuritemdata!=NULL)

{

repaintctrltrmdlg(m\_pcuritemdata->node);

}

t\_ireturn=1;

// 修改当前和父节点图标

HTREEITEM t\_node;

t\_node=m\_cfgtree.GetParentItem(v\_node);

for(;t\_node!=NULL;)

{

struct treeitemdata \*t\_pid;

t\_pid=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(t\_node));

if(t\_pid->runstatus==iconready)

{

t\_pid->runstatus=iconrwn;

this->SendMessage(WM\_appdlgnodepropertymodify,(WPARAM)t\_pid,NULL);

}

t\_node=m\_cfgtree.GetParentItem(t\_node);

}

// 使用信号量进行PV操作

ReleaseSemaphore(m\_hsemaphore,1,NULL);

}

}

if(t\_pitemdata->runmtmode==0)

{

// 顺序化方式

// 注意禁止空节点，既非脚本节点不存在脚本节点的子节点

t\_hChildItem=m\_cfgtree.GetChildItem(v\_node);

t\_ireturn=runtewdlgstart(t\_hChildItem);

}

else if(t\_pitemdata->runmtmode==1)

{

// 并行

t\_hChildItem=m\_cfgtree.GetChildItem(v\_node);

runtewdlgstart(t\_hChildItem);

t\_hChildItem=m\_cfgtree.GetNextSiblingItem(v\_node);

t\_ireturn=runtewdlgstart(t\_hChildItem);

}

else

{

assert(0);

}

}

}

// run over

if((NULL==v\_node)&&(m\_trmtab.GetItemCount()==0))

{

// 保存测试结果记录,包含ng行描述

setnginformation();

runtewdlgexit(v\_node);

if(getproducttest())

{

// 生产测试结果确认

getproducttestresultmsg();

}

if(this->m\_configflag.loop==1)

{

// start

OnDlgstart();

}

}

return t\_ireturn;

}

// 生产测试结果确认

int CtewDlg::getproducttestresultmsg()

{

// 准备生产测试结果确认

cdlgproductresult t\_dlgproductresult;

t\_dlgproductresult.m\_ptewdlg=this;

CString t\_csdirname;

int t\_nResponse=t\_dlgproductresult.DoModal();

if(t\_nResponse==IDOK)

{

// 创建测试报告

if((m\_mapdutmessage[\_T("002ProductSN")].GetLength()!=0)||(m\_mapdutmessage[\_T("003SN1")].GetLength()!=0))

{

// 报告文件路径

t\_csdirname=m\_mapdutmessage[\_T("002ProductSN")];

if(t\_csdirname.GetLength()==0)

{

t\_csdirname=m\_mapdutmessage[\_T("003SN1")];

}

// 002ProductSN or 003SN1 生成生产测试报告

setproducttestresultmsg(t\_csdirname);

// 009SN2 生成生产测试报告

if(((m\_mapdutmessage[\_T("002ProductSN")].GetLength()==0)&&(m\_mapdutmessage[\_T("009SN2")].GetLength()!=0)))

{

// 报告文件路径

t\_csdirname=m\_mapdutmessage[\_T("009SN2")];

// 009SN2 生成生产测试报告

setproducttestresultmsg(t\_csdirname);

}

}

else

{

AfxMessageBox(\_T("ProductSN or SN1 error"));

}

}

else if(t\_nResponse==IDCANCEL)

{

}

return 0;

}

// 保存测试结果记录,包含ng行描述

int CtewDlg::setnginformation()

{

CStdioFile t\_csf;

if(t\_csf.Open(m\_cscsvpath+RESULT\_FILE,CFile::modeWrite|CFile::modeCreate|CFile::shareDenyNone))

{

setuserdefinetreeitem(m\_cfgtree.GetRootItem(),8,&t\_csf);

t\_csf.Close();

}

else

{

assert(0);

}

return 0;

}

// 生成生产测试报告

int CtewDlg::setproducttestresultmsg(CString v\_csdirname)

{

CString t\_csresultpath;

t\_csresultpath=PRODUCT\_RESULT\_PATH+v\_csdirname+\_T("\\");

// 创建目录

pathmkdir(t\_csresultpath);

// 计算文件数量t\_i

CFileFind t\_f;

CString t\_cs;

BOOL t\_bcfg=t\_f.FindFile(t\_csresultpath+\_T("\*.txt"));

int t\_i=0;

while(t\_bcfg)

{

t\_bcfg=t\_f.FindNextFile();

t\_i++;

}

t\_f.Close();

// make file name

CString t\_csresultfile;

t\_csresultfile.Format(\_T("%d"),t\_i);

t\_csresultfile=t\_csresultpath+v\_csdirname+\_T("\_")+t\_csresultfile+\_T(".txt");

CStdioFile t\_csf;

// file operate

t\_csf.Open(t\_csresultfile,CFile::modeWrite|CFile::shareDenyNone|CFile::modeCreate);

// write

map<CString,CString>::iterator t\_it;

CString t\_csvalue;

for(t\_it=m\_mapdutmessage.begin();t\_it!=m\_mapdutmessage.end();++t\_it)

{

if(t\_it->second.GetLength()==0)

{

t\_csvalue=\_T("[N/A]");

}

else

{

t\_csvalue=t\_it->second;

}

t\_csf.WriteString(t\_it->first+\_T("=")+t\_csvalue+\_T("\n"));

}

for(t\_it=m\_mapmodulefield.begin();t\_it!=m\_mapmodulefield.end();++t\_it)

{

if(t\_it->second==\_T("0"))

{

t\_csvalue=\_T("OK");

}

else

{

t\_csvalue=\_T("NG[")+t\_it->second+\_T("]");

}

t\_csf.WriteString(t\_it->first+\_T("=")+t\_csvalue+\_T("\n"));

t\_csf.Flush();

}

t\_csf.Close();

return 0;

}

// 获取生产测试场景

int CtewDlg::getproducttest()

{

if(m\_mapappcfg[\_T("producttest")]==\_T("1"))

{

return TRUE;

}

else

{

return FALSE;

}

}

// 获取database场景

int CtewDlg::getdatabase()

{

if(m\_mapappcfg[\_T("database")]==\_T("1"))

{

return TRUE;

}

else

{

return FALSE;

}

}

// 刷新当前节点

int CtewDlg::setcurtreenode(HTREEITEM v\_node)

{

assert(v\_node);

volatile struct treeitemdata \*t\_pitemdataold;

struct treeitemdata \*t\_pitemdatanew;

if(m\_pcuritemdata==NULL)

{

m\_pcuritemdata=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(v\_node));

}

t\_pitemdataold=m\_pcuritemdata;

t\_pitemdatanew=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(v\_node));

assert(t\_pitemdataold);

assert(t\_pitemdatanew);

if(t\_pitemdatanew->ptrdlg!=NULL)

{

if(t\_pitemdataold->ptrdlg!=NULL)

{

(reinterpret\_cast<ctrmdlg \*>(t\_pitemdataold->ptrdlg))->ShowWindow(SW\_HIDE);

}

repaintctrltrmdlg(t\_pitemdatanew->node);

(reinterpret\_cast<ctrmdlg \*>(t\_pitemdatanew->ptrdlg))->ShowWindow(SW\_SHOW);

m\_pcuritemdata=t\_pitemdatanew;

m\_trmtab.SetCurSel(m\_pcuritemdata->trmtabctrlnumber);

}

return 0;

}

// 使能或禁止菜单

// menu表示从0开始的主菜单序号

// submenu表示子菜单ID号,为0表示操作对象为主菜单

// flag为MF\_GRAYED或MF\_ENABLED

// 返回TRUE正常或FALSE异常

int CtewDlg::setmenuaction(int v\_menu, int v\_submenu, int v\_flag)

{

assert(v\_menu>=0);

if(v\_submenu==0)

{

m\_pmenu->EnableMenuItem(v\_menu,MF\_BYPOSITION|v\_flag); //禁用主菜单

return TRUE;

}

else

{

m\_pmenu->GetSubMenu(v\_menu)->EnableMenuItem(v\_submenu,v\_flag); //menu表示从0开始的主菜单序号,submenu表示子菜单ID号,MF\_GRAYED表示禁用并且为灰色

return TRUE;

}

return FALSE;

}

// 工具栏使能或禁止

// id表示按钮ID号

// flag为false或ture

// 返回TRUE正常或FALSE异常

int CtewDlg::settoolbaraction(int v\_id,BOOL v\_flag)

{

return m\_ptoolbarctrl->EnableButton(v\_id,v\_flag);

}

// deny all command

int CtewDlg::setallcmddeny()

{

setmenuaction(0,ID\_dlgopenfile,MF\_GRAYED);

setmenuaction(0,ID\_dlgopendir,MF\_GRAYED);

setmenuaction(0,ID\_dlgsave,MF\_GRAYED);

setmenuaction(0,ID\_dlgsaveas,MF\_GRAYED);

setmenuaction(1,ID\_dlgstart,MF\_GRAYED);

setmenuaction(1,ID\_dlgpause,MF\_GRAYED);

setmenuaction(1,ID\_dlgstop,MF\_GRAYED);

setmenuaction(2,ID\_dlgenablenode,MF\_GRAYED);

setmenuaction(2,ID\_dlgdisablenode,MF\_GRAYED);

setmenuaction(2,ID\_dlgquarynode,MF\_GRAYED);

settoolbaraction(ID\_dlgopenfile,false);

settoolbaraction(ID\_dlgopendir,false);

settoolbaraction(ID\_dlgstart,false);

settoolbaraction(ID\_dlgpause,false);

settoolbaraction(ID\_dlgstop,false);

return 0;

}

// ready

int CtewDlg::setcmdinready()

{

setmenuaction(0,ID\_dlgopenfile,MF\_ENABLED);

setmenuaction(0,ID\_dlgopendir,MF\_ENABLED);

setmenuaction(0,ID\_dlgsave,MF\_GRAYED);

setmenuaction(0,ID\_dlgsaveas,MF\_GRAYED);

setmenuaction(1,ID\_dlgstart,MF\_GRAYED);

setmenuaction(1,ID\_dlgpause,MF\_GRAYED);

setmenuaction(1,ID\_dlgstop,MF\_GRAYED);

setmenuaction(2,ID\_dlgenablenode,MF\_GRAYED);

setmenuaction(2,ID\_dlgdisablenode,MF\_GRAYED);

setmenuaction(2,ID\_dlgquarynode,MF\_GRAYED);

settoolbaraction(ID\_dlgopenfile,true);

settoolbaraction(ID\_dlgopendir,true);

settoolbaraction(ID\_dlgstart,false);

settoolbaraction(ID\_dlgpause,false);

settoolbaraction(ID\_dlgstop,false);

return 0;

}

// open

int CtewDlg::setcmdinopen()

{

setmenuaction(0,ID\_dlgopenfile,MF\_ENABLED);

setmenuaction(0,ID\_dlgopendir,MF\_ENABLED);

setmenuaction(0,ID\_dlgsave,MF\_ENABLED);

setmenuaction(0,ID\_dlgsaveas,MF\_ENABLED);

setmenuaction(1,ID\_dlgstart,MF\_ENABLED);

setmenuaction(1,ID\_dlgpause,MF\_GRAYED);

setmenuaction(1,ID\_dlgstop,MF\_GRAYED);

setmenuaction(2,ID\_dlgenablenode,MF\_ENABLED);

setmenuaction(2,ID\_dlgdisablenode,MF\_ENABLED);

setmenuaction(2,ID\_dlgquarynode,MF\_ENABLED);

settoolbaraction(ID\_dlgopenfile,true);

settoolbaraction(ID\_dlgopendir,true);

settoolbaraction(ID\_dlgstart,true);

settoolbaraction(ID\_dlgpause,false);

settoolbaraction(ID\_dlgstop,false);

return 0;

}

// start

int CtewDlg::setcmdinstart()

{

setmenuaction(0,ID\_dlgopenfile,MF\_GRAYED);

setmenuaction(0,ID\_dlgopendir,MF\_GRAYED);

setmenuaction(0,ID\_dlgsave,MF\_GRAYED);

setmenuaction(0,ID\_dlgsaveas,MF\_GRAYED);

setmenuaction(1,ID\_dlgstart,MF\_GRAYED);

setmenuaction(1,ID\_dlgpause,MF\_ENABLED);

setmenuaction(1,ID\_dlgstop,MF\_ENABLED);

setmenuaction(2,ID\_dlgenablenode,MF\_GRAYED);

setmenuaction(2,ID\_dlgdisablenode,MF\_GRAYED);

setmenuaction(2,ID\_dlgquarynode,MF\_GRAYED);

settoolbaraction(ID\_dlgopenfile,false);

settoolbaraction(ID\_dlgopendir,false);

settoolbaraction(ID\_dlgstart,false);

settoolbaraction(ID\_dlgpause,true);

settoolbaraction(ID\_dlgstop,true);

return 0;

}

// set用户define node

HTREEITEM CtewDlg::setuserdefinetreenode(HTREEITEM v\_fathernode,treeitemdata \*v\_ptreeitemdata)

{

// 节点句柄

assert(v\_ptreeitemdata);

HTREEITEM t\_hnode;

struct treeitemdata \*t\_pitemdata;

if(v\_fathernode)

{

// 非root节点

t\_pitemdata=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(v\_fathernode));

// cmp node level

assert((t\_pitemdata->level>=0)&&(v\_ptreeitemdata->level>=0));

// target node level match

if((t\_pitemdata->level+1)==v\_ptreeitemdata->level)

{

// 根据当前节点信息是否包含文件名确定下一节点level

if(t\_pitemdata->filename.GetLength()!=0)

{

t\_hnode=m\_cfgtree.GetParentItem(m\_cfgtree.GetParentItem(v\_fathernode));

t\_hnode=setuserdefinetreenode(t\_hnode,v\_ptreeitemdata);

}

else

{

// 创建非根节点

t\_hnode=m\_cfgtree.InsertItem(v\_ptreeitemdata->comment,v\_ptreeitemdata->runstatus,v\_ptreeitemdata->runstatus,v\_fathernode,TVI\_LAST);

// 配置节点携带信息,指针引用

v\_ptreeitemdata->node=t\_hnode;

m\_cfgtree.SetItemData(t\_hnode,(DWORD)v\_ptreeitemdata);

}

}

// target node level big

else if((t\_pitemdata->level+1)<v\_ptreeitemdata->level)

{

// 根据当前节点信息是否包含文件名确定下一节点level

if(t\_pitemdata->filename.GetLength()!=0)

{

t\_hnode=m\_cfgtree.GetParentItem(m\_cfgtree.GetParentItem(v\_fathernode));

}

else

{

struct treeitemdata \*t\_ptemp;

t\_ptemp=new treeitemdata;

\*t\_ptemp=\*t\_pitemdata;

t\_ptemp->level+=1;

t\_hnode=m\_cfgtree.InsertItem(t\_ptemp->comment,t\_ptemp->runstatus,t\_ptemp->runstatus,v\_fathernode,TVI\_LAST);

t\_ptemp->node=t\_hnode;

m\_cfgtree.SetItemData(t\_hnode,(DWORD)t\_ptemp);

}

t\_hnode=setuserdefinetreenode(t\_hnode,v\_ptreeitemdata);

}

// target node level small or equal

else

{

t\_hnode=v\_fathernode;

t\_pitemdata=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(t\_hnode));

if(t\_pitemdata->filename.GetLength()!=0)

{

t\_hnode=m\_cfgtree.GetParentItem(t\_hnode);

t\_pitemdata=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(t\_hnode));

if(v\_ptreeitemdata->filename.GetLength()==0)

{

t\_hnode=m\_cfgtree.GetParentItem(t\_hnode);

t\_pitemdata=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(t\_hnode));

}

}

for(;(t\_pitemdata->level)>=(v\_ptreeitemdata->level);)

{

t\_hnode=m\_cfgtree.GetParentItem(t\_hnode);

t\_pitemdata=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(t\_hnode));

}

t\_hnode=setuserdefinetreenode(t\_hnode,v\_ptreeitemdata);

}

}

else

{

// 配置根节点携带信息,指针引用

if(v\_ptreeitemdata->level!=0)

{

t\_pitemdata=reinterpret\_cast<treeitemdata \*>(getuserdefinedefaultitemdata());

// 创建根节点

t\_hnode=m\_cfgtree.InsertItem(t\_pitemdata->comment,t\_pitemdata->runstatus,t\_pitemdata->runstatus);

t\_pitemdata->node=t\_hnode;

m\_cfgtree.SetItemData(t\_hnode,(DWORD)t\_pitemdata);

t\_hnode=setuserdefinetreenode(t\_hnode,v\_ptreeitemdata);

}

else

{

// 创建根节点

t\_hnode=m\_cfgtree.InsertItem(v\_ptreeitemdata->comment,v\_ptreeitemdata->runstatus,v\_ptreeitemdata->runstatus);

v\_ptreeitemdata->node=t\_hnode;

t\_pitemdata=v\_ptreeitemdata;

m\_cfgtree.SetItemData(t\_hnode,(DWORD)t\_pitemdata);

}

}

return t\_hnode;

}

// 清空树控件node

int CtewDlg::setuserdefinetreeempty()

{

// clear tree from root node

m\_pcuritemdata=0;

setuserdefinetreeitem(m\_cfgtree.GetRootItem(),0);

if(m\_cfgtree.DeleteAllItems()==0)

{

assert(0);

}

return 0;

}

// 遍历树控件节点

// 0-删除携带信息指针，清空节点携带信息引用

// 1-展开节点

// 2-save

// 3-scan script filename

// 4-dlgstop

// 5-dlgstart icon&text初始化

// 6-trm log path init

// 7-replace $\_

// 8-保存测试结果记录,包含ng行描述

int CtewDlg::setuserdefinetreeitem(HTREEITEM v\_nodea,int v\_i,CStdioFile \*v\_pcsf)

{

struct treeitemdata \*t\_pid;

CString t\_cs,t\_cstemp;

if(v\_nodea!=NULL)

{

t\_pid=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(v\_nodea));

if(t\_pid!=NULL)

{

switch(v\_i)

{

case 0:

// 0-删除携带信息指针，清空节点携带信息引用

delete t\_pid;

m\_cfgtree.SetItemData(v\_nodea,NULL);

break;

case 1:

// 1-展开节点

m\_cfgtree.Expand(v\_nodea,TVE\_EXPAND);

break;

case 2:

// 2-save

assert(v\_pcsf);

v\_pcsf->WriteString(getuserdefineline(v\_nodea));

break;

case 3:

// 3-scan script filename

if(t\_pid->filename.GetLength()!=0)

{

CFileFind t\_fFind;

CString t\_cspath=t\_pid->path+t\_pid->filename;

if(t\_fFind.FindFile(t\_cspath))

{

}

else

{

t\_pid->enableflag=0;

CString t\_csmsg;

t\_csmsg.Format(\_T("%s is not exist,\ndisable node %s!"),t\_cspath,t\_pid->comment);

AfxMessageBox(t\_csmsg);

}

t\_fFind.Close();

}

break;

case 4:

// 4-dlgstop

if((t\_pid->ptrdlg!=0)&&(t\_pid->exitflag!=1))

{

(reinterpret\_cast<ctrmdlg \*>(t\_pid->ptrdlg))->SendMessage(WM\_trmdlgstop,0,0);

t\_pid->exitflag=1;

t\_pid->trmtabctrlnumber=-1;

}

break;

case 5:

// 5-dlgstart

t\_pid->exitflag=0;

t\_pid->faultcounter=0;

if(t\_pid->enableflag==0)

{

t\_pid->runstatus=icondisable;

}

else

{

t\_pid->runstatus=iconready;

}

this->SendMessage(WM\_appdlgnodepropertymodify,(WPARAM)t\_pid,NULL);

break;

case 6:

// 6-trm log path init

if(t\_pid->node!=m\_cfgtree.GetRootItem())

{

struct treeitemdata \*t\_pfatherid;

CString t\_csdiff;

t\_pfatherid=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(m\_cfgtree.GetParentItem(t\_pid->node)));

t\_csdiff=t\_pid->path.Right(t\_pid->path.GetLength()-t\_pfatherid->path.GetLength());

t\_pid->logpath=t\_pfatherid->logpath+t\_csdiff;

}

pathmkdir(t\_pid->logpath);

break;

case 7:

// 7-replace $\_

if(t\_pid->filename.GetLength()!=0)

{

getstring4replacelabel(&(t\_pid->host));

}

break;

case 8:

// 8-保存测试结果记录,包含ng行描述,忽略disable节点

if(t\_pid->enableflag==1)

{

t\_cstemp.Format(\_T("%d"),t\_pid->level);

t\_cs=\_T("{")+t\_cstemp+\_T(";")+t\_pid->comment+\_T(";");

t\_cstemp.Format(\_T("NG(%d)"),t\_pid->faultcounter);

t\_cs+=t\_cstemp+\_T(";")+t\_pid->filename+\_T(";}\n");

t\_cstemp.Empty();

for(int t\_i=0;t\_i<t\_pid->level;t\_i++)

{

t\_cstemp+=\_T(" ");

}

t\_cs=t\_cstemp+t\_cs;

if(t\_pid->nginformation.GetLength()>0)

{

t\_cs+=t\_pid->nginformation;

}

v\_pcsf->WriteString(t\_cs);

v\_pcsf->Flush();

}

break;

default:

break;

}

}

if(m\_cfgtree.ItemHasChildren(v\_nodea)!=NULL)

{

HTREEITEM t\_hChildItem=m\_cfgtree.GetChildItem(v\_nodea);

while(t\_hChildItem!=NULL)

{

setuserdefinetreeitem(t\_hChildItem,v\_i,v\_pcsf); //递归遍历子节点

t\_hChildItem=m\_cfgtree.GetNextSiblingItem(t\_hChildItem);

}

}

}

return 0;

}

// 模拟鼠标点击

int CtewDlg::setmouseclick(CPoint v\_pt)

{

CPoint t\_opt;

GetCursorPos(&t\_opt);

SetCursorPos(v\_pt.x,v\_pt.y);

// mouse\_event 发送模拟鼠标消息到该矩形内一点。

mouse\_event(MOUSEEVENTF\_LEFTDOWN,0,0,0,0);

mouse\_event(MOUSEEVENTF\_LEFTUP,0,0,0,0);

SetCursorPos(t\_opt.x,t\_opt.y);

return 0;

}

// 选中tabpage，关联trmdlg和cfgtree

int CtewDlg::settrmtabpage(int v\_i)

{

if(m\_pcuritemdata!=NULL)

{

if(v\_i!=m\_pcuritemdata->trmtabctrlnumber)

{

setcurtreenode(m\_ptrmtabpageitemdataarray[v\_i]->node);

CRect t\_rcItem;

// 取得该ITEM的坐标。CTreeCtrl::GetItemRect

m\_cfgtree.GetItemRect(m\_pcuritemdata->node,&t\_rcItem,TRUE);

// 判断坐标在窗口范围

CRect t\_rcdlg;

m\_cfgtree.GetClientRect(&t\_rcdlg);

if(t\_rcItem.bottom<t\_rcdlg.bottom)

{

// 转换到屏幕坐标 CTreeCtrl::ClientToScreen

CPoint t\_pt;

t\_pt.x=t\_rcItem.left+t\_rcItem.Width()/2;

t\_pt.y=t\_rcItem.top+t\_rcItem.Height()/2;

::ClientToScreen(m\_cfgtree.m\_hWnd,&t\_pt);

// 模拟鼠标点击

setmouseclick(t\_pt);

}

}

}

return 0;

}

// save user define file

int CtewDlg::setuserdefinefile(CString v\_filename)

{

// 如果点击了文件对话框上的“打开”按钮，则将选择的文件路径显示到编辑框里

m\_mapappcfg[\_T("lastfile")]=getfilefrompath(v\_filename);

// path包含文件名

m\_mapappcfg[\_T("lastpath")]=getdirfrompath(v\_filename);

CStdioFile t\_csf;

if(t\_csf.Open(v\_filename,CFile::modeWrite|CFile::modeCreate|CFile::shareDenyNone))

{

// 遍历树控件节点,0-删除携带信息指针,1-展开节点,2-save

setuserdefinetreeitem(m\_cfgtree.GetRootItem(),2,&t\_csf);

t\_csf.Close();

}

else

{

assert(0);

}

return 0;

}

// 检查节点完成状态,延伸至父节点

int CtewDlg::setuserdefinenodeexitflag(treeitemdata \*v\_ptreeitemdata)

{

HTREEITEM t\_node;

// int t\_runstatus=iconrwn;

// 获取第一个兄弟节点

t\_node=m\_cfgtree.GetParentItem(v\_ptreeitemdata->node);

if(t\_node!=NULL)

{

int t\_exitflag=1;

treeitemdata \*t\_pid;

t\_node=m\_cfgtree.GetChildItem(t\_node);

assert(t\_node);

for(;t\_node!=NULL;)

{

t\_pid=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(t\_node));

if((t\_pid->exitflag!=1)&&(t\_pid->enableflag==1))

{

t\_exitflag=0;

}

t\_node=m\_cfgtree.GetNextSiblingItem(t\_node);

}

t\_node=m\_cfgtree.GetParentItem(v\_ptreeitemdata->node);

t\_pid=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(t\_node));

assert(t\_pid);

if(t\_exitflag==1)

{

t\_pid->exitflag=1;

setuserdefinenodeexitflag(t\_pid);

// 刷新节点图标

if(t\_pid->faultcounter>0)

{

t\_pid->runstatus=iconcwf;

}

else

{

t\_pid->runstatus=iconcwn;

}

}

else if(t\_exitflag==0)

{

t\_pid->exitflag=0;

// 刷新节点图标

if(t\_pid->faultcounter>0)

{

t\_pid->runstatus=iconrwf;

}

else

{

t\_pid->runstatus=iconrwn;

}

}

else

{

assert(0);

}

this->SendMessage(WM\_appdlgnodepropertymodify,(WPARAM)t\_pid);

}

return 0;

}

// 树控件携带信息enableflag

// v\_flag:1-enable;0-disable

int CtewDlg::modifytreeallitemdatarunmode(HTREEITEM v\_node,int v\_flag)

{

if(v\_node!=NULL)

{

if((m\_cfgtree.GetItemData(v\_node))!=NULL)

{

// 修改节点信息中runmode

if(v\_flag==TRUE)

{

HTREEITEM t\_node=v\_node;

for(;t\_node!=NULL;)

{

(reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(t\_node)))->enableflag=TRUE;

m\_cfgtree.SetItemImage(t\_node,iconready,iconready);

t\_node=m\_cfgtree.GetParentItem(t\_node);

}

}

else if(v\_flag==FALSE)

{

(reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(v\_node)))->enableflag=FALSE;

m\_cfgtree.SetItemImage(v\_node,icondisable,icondisable);

}

else

{

assert(0);

}

}

if(m\_cfgtree.ItemHasChildren(v\_node)!=NULL)

{

HTREEITEM t\_hChildItem=m\_cfgtree.GetChildItem(v\_node);

while(t\_hChildItem!=NULL)

{

modifytreeallitemdatarunmode(t\_hChildItem,v\_flag); //递归遍历子节点

t\_hChildItem=m\_cfgtree.GetNextSiblingItem(t\_hChildItem);

}

}

}

return 0;

}

// set cfg tree property

int CtewDlg::setuserdefinenodeproperty(HTREEITEM v\_node)

{

// 获取当前选中节点的句柄

HTREEITEM t\_hItem;

if(v\_node==0)

{

t\_hItem=m\_cfgtree.GetSelectedItem();

}

else

{

t\_hItem=v\_node;

}

assert(t\_hItem);

// 检查节点有效,可以进行节点属性编辑

if(t\_hItem!=0)

{

// 模态创建节点属性对话框

cnodepropertydlg t\_nodepropertydlg;

// 获取选中节点信息赋值给节点属性对话框实例的公共变量m\_pitemdata

t\_nodepropertydlg.m\_pitemdata=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(t\_hItem));

t\_nodepropertydlg.m\_pappdlg=this;

if(t\_nodepropertydlg.DoModal()==IDOK)

{

}

}

return 0;

}

// 文件/目录支持拖拽

void CtewDlg::OnDropFiles(HDROP hDropInfo)

{

// TODO: ÔÚ´ËÌí¼ÓÏûÏ¢´¦Àí³ÌÐò´úÂëºÍ/»òµ÷ÓÃÄ¬ÈÏÖµ

CString t\_filename;

int t\_DropCount=DragQueryFile(hDropInfo,-1,NULL,0);

if(t\_DropCount==1)

{

DragQueryFile(hDropInfo,0,t\_filename.GetBuffer(\_MAX\_PATH),\_MAX\_PATH);

t\_filename.ReleaseBuffer();

if(PathIsDirectory(t\_filename)||PathIsRoot(t\_filename))

{

getuserdefinefromdir(t\_filename);

}

else

{

// 如果点击了文件对话框上的“打开”按钮，则将选择的文件路径显示到编辑框里

m\_mapappcfg[\_T("lastfile")]=getfilefrompath(t\_filename);

// path包含文件名

m\_mapappcfg[\_T("lastpath")]=getdirfrompath(t\_filename);

// 清空树控件node

setuserdefinetreeempty();

getuserdefinefromfile(m\_mapappcfg[\_T("lastpath")],m\_mapappcfg[\_T("lastfile")]);

}

}

CDialogEx::OnDropFiles(hDropInfo);

}

// 鼠标处于控件间隙时可拖动修改scale

void CtewDlg::OnMouseMove(UINT nFlags, CPoint point)

{

// TODO: ÔÚ´ËÌí¼ÓÏûÏ¢´¦Àí³ÌÐò´úÂëºÍ/»òµ÷ÓÃÄ¬ÈÏÖµ

if(m\_modctrlscaleflag)

{

if(((point.x<=m\_ctrlpositionarray[0][0]+BORDERJUDGE)||(point.x>=m\_ctrlpositionarray[2][2]-BORDERJUDGE))||((point.y<=m\_ctrlpositionarray[0][1]+BORDERJUDGE)||(point.y>=m\_ctrlpositionarray[2][3]-BORDERJUDGE)))

{

SendMessage(WM\_LBUTTONUP,0,0);

}

}

else

{

// 光标类型

LPWSTR t\_cursortype;

// 鼠标处于控件间隙时可拖动修改scale

// 列

if((point.x>=m\_pointns1.x)&&(point.x<=m\_pointns2.x)&&(point.y>=m\_pointns1.y)&&(point.y<=m\_pointns2.y))

{

t\_cursortype=IDC\_SIZEWE;

}

// 行

else if((point.x>=m\_pointwe1.x)&&(point.x<=m\_pointwe2.x)&&(point.y>=m\_pointwe1.y)&&(point.y<=m\_pointwe2.y))

{

t\_cursortype=IDC\_SIZENS;

}

else

{

t\_cursortype=IDC\_ARROW;

}

SetCursor(LoadCursor(NULL,t\_cursortype));

ShowCursor(TRUE);

}

CDialogEx::OnMouseMove(nFlags, point);

}

// press mouse lbutton set scale行为flag

void CtewDlg::OnLButtonDown(UINT nFlags, CPoint point)

{

// TODO: ÔÚ´ËÌí¼ÓÏûÏ¢´¦Àí³ÌÐò´úÂëºÍ/»òµ÷ÓÃÄ¬ÈÏÖµ

// 光标类型

LPWSTR t\_cursortype;

// 鼠标处于控件间隙时可拖动修改scale,修改时隐藏主要控件

// 修改scale行为flag,1indicate x,2 indicate y,0 indicate none

// 修改列宽

if((point.x>=m\_pointns1.x)&&(point.x<=m\_pointns2.x)&&(point.y>=m\_pointns1.y)&&(point.y<=m\_pointns2.y))

{

t\_cursortype=IDC\_SIZEWE;

m\_modctrlscaleflag=1;

GetDlgItem(IDC\_cfgtree)->ShowWindow(SW\_HIDE);

GetDlgItem(IDC\_trmtab)->ShowWindow(SW\_HIDE);

GetDlgItem(IDC\_loglist)->ShowWindow(SW\_HIDE);

UpdateWindow();

}

// 修改行高

else if((point.x>=m\_pointwe1.x)&&(point.x<=m\_pointwe2.x)&&(point.y>=m\_pointwe1.y)&&(point.y<=m\_pointwe2.y))

{

t\_cursortype=IDC\_SIZENS;

m\_modctrlscaleflag=2;

GetDlgItem(IDC\_cfgtree)->ShowWindow(SW\_HIDE);

GetDlgItem(IDC\_trmtab)->ShowWindow(SW\_HIDE);

GetDlgItem(IDC\_loglist)->ShowWindow(SW\_HIDE);

UpdateWindow();

}

else

{

t\_cursortype=IDC\_ARROW;

m\_modctrlscaleflag=0;

}

SetCursor(LoadCursor(NULL,t\_cursortype));

ShowCursor(TRUE);

CDialogEx::OnLButtonDown(nFlags, point);

}

// press mouse lbutton unset scale行为flag

void CtewDlg::OnLButtonUp(UINT nFlags, CPoint point)

{

// TODO: ÔÚ´ËÌí¼ÓÏûÏ¢´¦Àí³ÌÐò´úÂëºÍ/»òµ÷ÓÃÄ¬ÈÏÖµ

CRect t\_rect;

GetClientRect(&t\_rect);

// 修改scale行为flag,1indicate x,2 indicate y,0 indicate none

// 修改完成后显示主要控件

if(m\_modctrlscaleflag)

{

if(m\_modctrlscaleflag==1)

{

if((point.x>=m\_ctrlpositionarray[0][0]+BORDERJUDGE)&&(point.x<=m\_ctrlpositionarray[2][2]-BORDERJUDGE))

{

m\_ctrlpositionarray[1][0]=m\_ctrlpositionarray[2][0]=m\_ctrlpositionarray[0][2]=point.x;

}

}

else if(m\_modctrlscaleflag==2)

{

if((point.y>=m\_ctrlpositionarray[0][1]+BORDERJUDGE)&&(point.y<=m\_ctrlpositionarray[2][3]-BORDERJUDGE))

{

m\_ctrlpositionarray[1][3]=m\_ctrlpositionarray[2][1]=point.y;

}

}

repaintctrl(t\_rect.Width(),t\_rect.Height());

GetDlgItem(IDC\_cfgtree)->ShowWindow(SW\_SHOW);

GetDlgItem(IDC\_trmtab)->ShowWindow(SW\_SHOW);

GetDlgItem(IDC\_loglist)->ShowWindow(SW\_SHOW);

UpdateWindow();

}

SetCursor(LoadCursor(NULL,IDC\_ARROW));

ShowCursor(TRUE);

m\_modctrlscaleflag=0;

CDialogEx::OnLButtonUp(nFlags, point);

}

// 界面命令

void CtewDlg::OnDlgopenfile()

{

// TODO: ÔÚ´ËÌí¼ÓÃüÁî´¦Àí³ÌÐò´úÂë

// 设置过滤器

CString t\_lastcfgpath;

TCHAR t\_szFilter[]=\_T("配置文件(\*.cfg)|\*.cfg|所有文件(\*.\*)|\*.\*||");

// 构造打开文件对话框

CFileDialog t\_fileDlg(TRUE,NULL,NULL,0,t\_szFilter);

//设置执行配置文件初始路径,如果上次路径为空,使用应用程序路径

t\_lastcfgpath=m\_mapappcfg[\_T("lastpath")];

if(t\_lastcfgpath.GetLength()==0)

{

t\_lastcfgpath=m\_mapappcfg[\_T("apppath")];

}

// default path

t\_fileDlg.m\_ofn.lpstrInitialDir=t\_lastcfgpath;

// dlg title

t\_fileDlg.m\_ofn.lpstrTitle= \_T("打开...");

// 显示打开文件对话框

if(IDOK == t\_fileDlg.DoModal())

{

CStdioFile t\_csf;

// 判断所打开文件存在

if(t\_csf.Open(t\_fileDlg.GetPathName(),CFile::modeRead|CFile::shareDenyNone))

{

t\_csf.Close();

// 如果点击了文件对话框上的“打开”按钮，则将选择的文件路径显示到编辑框里

m\_mapappcfg[\_T("lastfile")]=t\_fileDlg.GetFileName();

// path包含文件名

m\_mapappcfg[\_T("lastpath")]=getdirfrompath(t\_fileDlg.GetPathName());

// 生成cfg tree

getuserdefinecfgtree();

}

else

{

AfxMessageBox(\_T("sorry,no find this file"));

}

}

// menu open

setcmdinopen();

}

void CtewDlg::OnDlgopendir()

{

// TODO: ÔÚ´ËÌí¼ÓÃüÁî´¦Àí³ÌÐò´úÂë

// 存放选择的目录路径

TCHAR t\_wcpath[MAX\_PATH];

ZeroMemory(t\_wcpath, sizeof(t\_wcpath));

CString t\_path;

BROWSEINFO t\_bi;

t\_bi.hwndOwner=m\_hWnd;

// 打开上次目录

// t\_bi.pidlRoot=GetItemIDListFromFilePath(m\_mapappcfg[\_T("lastpath")]);

t\_bi.pidlRoot=NULL;

t\_bi.pszDisplayName=t\_wcpath;

t\_bi.lpszTitle=\_T("选择目录");

t\_bi.ulFlags=0;

t\_bi.lpfn=NULL;

t\_bi.lParam=0;

t\_bi.iImage=0;

// 弹出选择目录对话框

LPITEMIDLIST t\_lp=SHBrowseForFolder(&t\_bi);

int t\_i=SHGetPathFromIDList(t\_lp,t\_wcpath);

t\_path.Format(\_T("%s"),t\_wcpath );

if(t\_lp && t\_i)

{

m\_mapappcfg[\_T("lastpath")]=t\_path;

// SetWindowText:设置标题

SetWindowText(\_T("tew---")+t\_path);

// 清空树控件node

setuserdefinetreeempty();

getuserdefinefromdir(t\_path);

}

}

void CtewDlg::OnDlgpause()

{

// TODO: ÔÚ´ËÌí¼ÓÃüÁî´¦Àí³ÌÐò´úÂë

AfxMessageBox(\_T("OnDlgpause"));

}

void CtewDlg::OnDlgstart()

{

// TODO: ÔÚ´ËÌí¼ÓÃüÁî´¦Àí³ÌÐò´úÂë

HTREEITEM t\_node;

t\_node=m\_cfgtree.GetRootItem();

struct treeitemdata \*t\_id;

t\_id=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(t\_node));

// 检查root是否enable

if(t\_id->enableflag==0)

{

AfxMessageBox(\_T("root节点未激活"));

}

else

{

INT\_PTR t\_nResponse=0;

// 获取时间,关联测试时间和log目录

m\_csstarttime=this->getsystemdatetime(1);

// 清空终端消息

m\_trmmsg.Empty();

if(getproducttest())

{

m\_mapdutmessage.clear();

// 获取mapdutmessage

if(getdatafile2mapdutmessage())

{

assert(0);

}

// 准备生产测试场景

cdlgdutmsg t\_dlgdutmsg;

t\_dlgdutmsg.m\_ptewdlg=this;

t\_nResponse=t\_dlgdutmsg.DoModal();

if(t\_nResponse==IDOK)

{

CString t\_csfile=m\_mapappcfg[\_T("apppath")]+DUT\_MSG\_FILE;

CStdioFile t\_csf;

// 防止日期表项被修改

m\_mapdutmessage[\_T("000Date")]=m\_csstarttime;

if(t\_csf.Open(t\_csfile,CFile::modeWrite|CFile::shareDenyNone|CFile::modeCreate))

{

getmap2file(&t\_csf,&m\_mapdutmessage);

t\_csf.Close();

}

else

{

AfxMessageBox(\_T("生产测试模板写入失败"));

assert(0);

}

// 遍历树控件node,替换$\_

HTREEITEM t\_node;

t\_node=m\_cfgtree.GetRootItem();

// 7-replace $\_

setuserdefinetreeitem(t\_node,7);

}

else if(t\_nResponse==IDCANCEL)

{

}

}

// 准备开始

if((t\_nResponse==IDOK)||(!getproducttest()))

{

runtewdlgperstart();

// menu start

setcmdinstart();

}

}

}

void CtewDlg::OnDlgstop()

{

// TODO: ÔÚ´ËÌí¼ÓÃüÁî´¦Àí³ÌÐò´úÂë

if(this->m\_configflag.loop==FALSE)

{

// 遍历树,dlgstop

setuserdefinetreeitem(m\_cfgtree.GetRootItem(),4);

// menu open

setcmdinopen();

// 模态创建节点属性对话框

cdlgprocessing t\_dlgprocessing;

// 获取选中节点信息赋值给节点属性对话框实例的公共变量m\_pitemdata

t\_dlgprocessing.m\_endflag=NULL;

t\_dlgprocessing.m\_flag=&(m\_pcuritemdata->ptrdlg);

if(t\_dlgprocessing.DoModal()==IDOK)

{

}

if(getproducttest())

{

// 生产测试结果确认

getproducttestresultmsg();

}

// 保存测试结果记录,包含ng行描述

setnginformation();

// run clear

runtewdlgclear(m\_cfgtree.GetRootItem());

}

else

{

AfxMessageBox(\_T("终止前请取消环境配置菜单中循环执行"));

}

}

void CtewDlg::OnDlghelpdoc()

{

// TODO: ÔÚ´ËÌí¼ÓÃüÁî´¦Àí³ÌÐò´úÂë

AfxMessageBox(\_T("OnDlghelpdoc"));

}

void CtewDlg::OnAbout()

{

// TODO: ÔÚ´ËÌí¼ÓÃüÁî´¦Àí³ÌÐò´úÂë

caboutdlg t\_aboutdlg;

t\_aboutdlg.m\_csversion=this->getfileversion();

if(t\_aboutdlg.DoModal()==IDOK)

{

}

}

void CtewDlg::OnDlgexplorer()

{

// TODO: ÔÚ´ËÌí¼ÓÃüÁî´¦Àí³ÌÐò´úÂë

ShellExecute(NULL,\_T("open"),m\_mapappcfg[\_T("lastpath")],NULL,NULL,SW\_SHOW);

}

void CtewDlg::OnNMRClickcfgtree(NMHDR \*pNMHDR, LRESULT \*pResult)

{

// TODO: ÔÚ´ËÌí¼Ó¿Ø¼þÍ¨Öª´¦Àí³ÌÐò´úÂë

CMenu menu;

POINT t\_pt={0};

// 得到鼠标点击位置

GetCursorPos(&t\_pt);

m\_cfgtree.ScreenToClient(&t\_pt);

UINT t\_uFlags;

HTREEITEM hItem=m\_cfgtree.HitTest(t\_pt,&t\_uFlags);

if ((hItem != NULL)&&(TVHT\_ONITEM&t\_uFlags))

{

m\_cfgtree.Select(hItem,TVGN\_CARET);

m\_cfgtree.ClientToScreen(&t\_pt);

menu.LoadMenu(IDR\_cfgtreemenu);//菜单资源ID

//m\_list是CListCtrl对象,这里的消息发到this

menu.GetSubMenu(0)->TrackPopupMenu(0,t\_pt.x,t\_pt.y,this);

}

\*pResult=0;

}

void CtewDlg::Ondlgenablenode()

{

// TODO: ÔÚ´ËÌí¼ÓÃüÁî´¦Àí³ÌÐò´úÂë

HTREEITEM t\_node=m\_cfgtree.GetSelectedItem();

if(t\_node)

{

struct treeitemdata \*t\_pitemdata=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(t\_node));

if(t\_node!=m\_cfgtree.GetRootItem())

{

// non root node展开节点

m\_cfgtree.Expand(t\_node,TVE\_EXPAND);

}

t\_pitemdata->enableflag=1;

t\_pitemdata->runstatus=0;

modifytreeallitemdatarunmode(t\_pitemdata->node,t\_pitemdata->enableflag);

}

}

void CtewDlg::Ondlgdisablenode()

{

// TODO: ÔÚ´ËÌí¼ÓÃüÁî´¦Àí³ÌÐò´úÂë

HTREEITEM t\_node=m\_cfgtree.GetSelectedItem();

if(t\_node)

{

if(t\_node!=m\_cfgtree.GetRootItem())

{

// non root node收起节点

//m\_cfgtree.Expand(t\_node,TVE\_COLLAPSE);

}

struct treeitemdata \*t\_pitemdata=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(t\_node));

assert(t\_pitemdata);

t\_pitemdata->enableflag=0;

t\_pitemdata->runstatus=6;

modifytreeallitemdatarunmode(t\_pitemdata->node,t\_pitemdata->enableflag);

t\_node=m\_cfgtree.GetParentItem(t\_node);

// 向上计算disable状态

if(t\_node!=NULL)

{

t\_node=m\_cfgtree.GetChildItem(t\_node);

int t\_enflag=0;

assert(t\_pitemdata);

for(;(t\_node!=NULL)||(t\_enflag==0);)

{

t\_pitemdata=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(t\_node));

if(t\_pitemdata->enableflag==1)

{

// enable则跳出循环

t\_enflag=1;

break;

}

else

{

if(m\_cfgtree.GetNextSiblingItem(t\_node)==NULL)

{

// 无兄弟节点,disable父节点

t\_node=m\_cfgtree.GetParentItem(t\_node);

if(t\_node!=NULL)

{

t\_pitemdata=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(t\_node));

t\_pitemdata->enableflag=0;

t\_pitemdata->runstatus=6;

modifytreeallitemdatarunmode(t\_pitemdata->node,t\_pitemdata->enableflag);

// 循环变量设为第一个兄弟节点

for(;m\_cfgtree.GetPrevSiblingItem(t\_node)!=NULL;)

{

t\_node=m\_cfgtree.GetPrevSiblingItem(t\_node);

}

}

else

{

// 无父节点则跳出循环

break;

}

}

else

{

// // 循环变量设为下一个兄弟节点

t\_node=m\_cfgtree.GetNextSiblingItem(t\_node);

}

}

}

}

}

}

void CtewDlg::Onctrlcfgtreeproperty()

{

// TODO: ÔÚ´ËÌí¼ÓÃüÁî´¦Àí³ÌÐò´úÂë

// set cfg tree property

setuserdefinenodeproperty(0);

}

void CtewDlg::OnNMDblclkcfgtree(NMHDR \*pNMHDR, LRESULT \*pResult)

{

// TODO: ÔÚ´ËÌí¼Ó¿Ø¼þÍ¨Öª´¦Àí³ÌÐò´úÂë

// set cfg tree property

setuserdefinenodeproperty(0);

// TRUE is 双击不收起树节点

\*pResult=TRUE;

}

void CtewDlg::Ondlgsaveas()

{

// TODO: ÔÚ´ËÌí¼ÓÃüÁî´¦Àí³ÌÐò´úÂë

// 设置过滤器

CString t\_lastcfgpath;

TCHAR t\_szFilter[]=\_T("配置文件(\*.cfg)|\*.cfg|所有文件(\*.\*)|\*.\*||");

// 构造打开文件对话框

CFileDialog t\_fileDlg(FALSE, \_T("\*.cfg"),NULL,OFN\_HIDEREADONLY|OFN\_OVERWRITEPROMPT,t\_szFilter,NULL);

//设置执行配置文件初始路径,如果上次路径为空,使用应用程序路径

t\_lastcfgpath=m\_mapappcfg[\_T("lastpath")];

if(t\_lastcfgpath.GetLength()==0)

{

t\_lastcfgpath=m\_mapappcfg[\_T("apppath")];

}

// default path

t\_fileDlg.m\_ofn.lpstrInitialDir=t\_lastcfgpath;

// dlg title

t\_fileDlg.m\_ofn.lpstrTitle= \_T("另存为...");

// 显示打开文件对话框

if(IDOK == t\_fileDlg.DoModal())

{

CStdioFile t\_csf;

// save user define file

setuserdefinefile(t\_fileDlg.GetPathName());

// 生成cfg tree

getuserdefinecfgtree();

}

}

void CtewDlg::Ondlgsave()

{

// TODO: ÔÚ´ËÌí¼ÓÃüÁî´¦Àí³ÌÐò´úÂë

if(m\_mapappcfg[\_T("lastfile")].GetLength()>0)

{

CString t\_csfilename;

int t\_iflag=0;

t\_csfilename=m\_mapappcfg[\_T("lastpath")]+m\_mapappcfg[\_T("lastfile")];

CStdioFile t\_csf;

t\_csf.Open(t\_csfilename,CFile::modeRead|CFile::shareDenyNone);

CString t\_csline;

for(;t\_csf.ReadString(t\_csline)!=NULL;)

{

if(getstring4replacelabel(&t\_csline))

{

t\_iflag=1;

break;

}

}

t\_csf.Close();

if(t\_iflag)

{

AfxMessageBox(\_T("配置文件含有替换标识,禁止保存!"));

}

else

{

// save user define file

setuserdefinefile(t\_csfilename);

// 生成cfg tree

getuserdefinecfgtree();

}

}

else

{

this->Ondlgsaveas();

}

}

void CtewDlg::Ondlgedit()

{

// TODO: ÔÚ´ËÌí¼ÓÃüÁî´¦Àí³ÌÐò´úÂë

// 当前节点信息

HTREEITEM t\_hItem;

t\_hItem=m\_cfgtree.GetSelectedItem();

// 检查节点有效,可以进行节点属性编辑

if(t\_hItem)

{

// 获取选中节点信息赋值给节点属性对话框实例的公共变量m\_pitemdata

struct treeitemdata \*t\_pitemdata=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(t\_hItem));

if(t\_pitemdata->filename.GetLength()!=0)

{

// 初始化脚本编辑dlg

cscripteditdlg \*t\_pscripteditdlg=new cscripteditdlg;

t\_pscripteditdlg->m\_filename=t\_pitemdata->path+t\_pitemdata->filename;

t\_pscripteditdlg->m\_pappdlg=this;

// 创建一个非模态对话框

t\_pscripteditdlg->Create(IDD\_scripteditdlg);

// 显示非模态对话框

t\_pscripteditdlg->ShowWindow(SW\_SHOWNORMAL);

}

}

}

void CtewDlg::OnTvnSelchangedcfgtree(NMHDR \*pNMHDR, LRESULT \*pResult)

{

LPNMTREEVIEW pNMTreeView=reinterpret\_cast<LPNMTREEVIEW>(pNMHDR);

// TODO: ÔÚ´ËÌí¼Ó¿Ø¼þÍ¨Öª´¦Àí³ÌÐò´úÂë

HTREEITEM t\_hItem;

t\_hItem=m\_cfgtree.GetSelectedItem();

setcurtreenode(t\_hItem);

\*pResult=0;

}

void CtewDlg::OnTcnSelchangetrmtab(NMHDR \*pNMHDR, LRESULT \*pResult)

{

// TODO: ÔÚ´ËÌí¼Ó¿Ø¼þÍ¨Öª´¦Àí³ÌÐò´úÂë

int t\_sel;

t\_sel=m\_trmtab.GetCurSel();

// 选中tabpage，关联trmdlg和cfgtree

settrmtabpage(t\_sel);

\*pResult=0;

}

void CtewDlg::Ondlglog()

{

// TODO: ÔÚ´ËÌí¼ÓÃüÁî´¦Àí³ÌÐò´úÂë

HTREEITEM t\_hItem;

t\_hItem=m\_cfgtree.GetSelectedItem();

if(t\_hItem)

{

struct treeitemdata \*t\_pid=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(t\_hItem));

CString t\_cslogfile=t\_pid->logpath+t\_pid->filename+TRM\_LOG\_EXT;

CStdioFile t\_csdlogfile;

// 初始化脚本编辑dlg

cscripteditdlg \*t\_pscripteditdlg=new cscripteditdlg;

t\_pscripteditdlg->m\_filename=t\_cslogfile;

t\_pscripteditdlg->m\_pappdlg=this;

// edit ctrl readonly

t\_pscripteditdlg->m\_readonlyflag=TRUE;

// 创建一个非模态对话框

t\_pscripteditdlg->Create(IDD\_scripteditdlg);

// 显示非模态对话框

t\_pscripteditdlg->ShowWindow(SW\_SHOWNORMAL);

}

}

void CtewDlg::Ondlgcsv()

{

// TODO: ÔÚ´ËÌí¼ÓÃüÁî´¦Àí³ÌÐò´úÂë

// 取前三行

CString t\_cs,t\_csline;

if(m\_csfcsv.Open(m\_cscsvpath+CSV\_FILE,CFile::modeRead|CFile::shareDenyNone))

{

for(int t\_i=0;(t\_i<3)&&(m\_csfcsv.ReadString(t\_csline));t\_i++)

{

t\_cs+=t\_csline+\_T("\n");

}

m\_csfcsv.Close();

}

else

{

t\_cs=\_T("csv file fail!");

}

AfxMessageBox(t\_cs);

}

void CtewDlg::OnTimer(UINT\_PTR nIDEvent)

{

// TODO: ÔÚ´ËÌí¼ÓÏûÏ¢´¦Àí³ÌÐò´úÂëºÍ/»òµ÷ÓÃÄ¬ÈÏÖµ

switch(nIDEvent)

{

case APPDLG1STIMER:

m\_timer++;

break;

case APPDLG60STIMER:

break;

default:

break;

}

CDialogEx::OnTimer(nIDEvent);

}

// 重载OnOK,避免回车关闭对话框

void CtewDlg::OnOK()

{

// TODO: ÔÚ´ËÌí¼Ó×¨ÓÃ´úÂëºÍ/»òµ÷ÓÃ»ùÀà

//CDialogEx::OnOK();

}

void CtewDlg::Ondlgconfig()

{

// TODO: ÔÚ´ËÌí¼ÓÃüÁî´¦Àí³ÌÐò´úÂë

// 模态创建节点属性对话框

cdlgconfig t\_dlgconfig;

// 指针传递

t\_dlgconfig.m\_pconfigflag=&m\_configflag;

t\_dlgconfig.m\_pappdlg=this;

if(t\_dlgconfig.DoModal()==IDOK)

{

}

}

// 打开节点文件夹

void CtewDlg::Ondlgopenpath()

{

// TODO: ÔÚ´ËÌí¼ÓÃüÁî´¦Àí³ÌÐò´úÂë

HTREEITEM t\_node=m\_cfgtree.GetSelectedItem();

struct treeitemdata \*t\_pitemdata=reinterpret\_cast<treeitemdata \*>(m\_cfgtree.GetItemData(t\_node));

assert(t\_pitemdata);

ShellExecute(NULL,\_T("open"),t\_pitemdata->path,NULL,NULL,SW\_SHOW);

}

BOOL CtewDlg::PreTranslateMessage(MSG\* pMsg)

{

// TODO: ÔÚ´ËÌí¼Ó×¨ÓÃ´úÂëºÍ/»òµ÷ÓÃ»ùÀà

if(m\_hAccTable)

{

if(::TranslateAccelerator(m\_hWnd,m\_hAccTable,pMsg))

{

return true;

}

}

return CDialogEx::PreTranslateMessage(pMsg);

}