```
#include "ftpwrap.h"
#include "fwdui.h"

/*
 *  help:   Print help message
 */

void cmd_help( PREQUEST lpReq ) {

        ftp_msgf( lpReq, 214, "This is FTP wrapper version %s\r\n"
              "Commands:\r\n"
              "    HELP  Print this help message\r\n"
              "    NOOP  Do nothing\r\n"
              "    QUIT  Exit and close connection\r\n"
              "End of HELP info", VERSION );

        return;
}


/*
 *  noop:   Performing no operation
 */

void cmd_noop( PREQUEST lpReq ) {

        ftp_msgf( lpReq, 250, "Requested action okay, completed" );

        return;
}


/*
 *  quit:   Terminate a SQLld session
 */

void cmd_quit( PREQUEST lpReq ) {

        lpReq->state = ST_QUIT;
        ftp_msgf( lpReq, 221, "Service closing transmission channel" );

        return;
}


/*
 *  port:
 */

int cmd_port( PREQUEST lpReq ) {
        _RPTF0(_CRT_WARN, "received PORT cmd.\n");

        // parse PORT's parameter
        int val[6];
        int i = 0, j = 0, k; // i=val conuter j=numstr counter k=parm string counter
        int parm_len = strlen( lpReq->ftp_parm[1] );
        char numstr[4];
        char                *invalidchar;       /*  Detect strtoul() convert error *
/

        memset( numstr, '\0', sizeof( numstr ) );

        for( k = 0; k < parm_len; k++ ) {
                if( lpReq->ftp_parm[1][k] == ',' ) {
                        numstr[j] = '\0';
                        if( strlen( numstr ) == 0 )
```



```
                                return 0;
                        // convert
                        val[i] = strtoul( numstr, &invalidchar, 10 );
                        if( invalidchar != (char *)( numstr + strlen( numstr ) ) )
                                return 0;
                        i++;
                        j = 0;
                        memset( numstr, '\0', sizeof( numstr ) );
                } else {
                        numstr[j] = lpReq->ftp_parm[1][k];
                        j++;
                        if( j > 3 )
                                return 0;
                }
        }

        if( strlen( numstr ) == 0 )
                return 0;

        // do last conversion
        val[i] = strtoul( numstr, &invalidchar, 10 );
        if( invalidchar != (char *)( numstr + strlen( numstr ) ) )
                return 0;

        if( i != 5 )
                return 0;

        for( i = 0; i < 6; i++ ) {
                if( val[i] < 0 || val[i] > 255 )
                        return 0;
        }

        _RPTF0(_CRT_WARN, "PORT cmd parameter is valid.\n");

        // set port

        int n;
        int first_ch = -1;
        int FoundReq = 0;
        for( n = 0; n < lpReq->pvar->fwd_state.num_requests; n++ ) {
                if( (lpReq->pvar->fwd_state.requests[n].status & FWD_DELETED) == 0 &&
                        lpReq->pvar->fwd_state.requests[n].spec.type == FWD_REMOTE_TO
_LOCAL ) {
                        if( first_ch == -1 )
                                first_ch = n;
                        if( lpReq->pvar->fwd_state.requests[n].spec.to_port2 > lpReq-
>lastremoteport ) {
                                FoundReq++;
                                break;
                        }
                }
        }
        if( ! FoundReq ) {
                if( first_ch != -1 )
                        n = first_ch;
                else
                        return 0;
        }

        lpReq->lastremoteport = lpReq->pvar->fwd_state.requests[n].spec.to_port2;
        _snprintf( lpReq->pvar->fwd_state.requests[n].spec.to_host,
                sizeof( lpReq->pvar->fwd_state.requests[n].spec.to_host ),
                "%d.%d.%d.%d", val[0], val[1], val[2], val[3] );
        lpReq->pvar->fwd_state.requests[n].spec.to_port = val[4] * 256 + val[5];

        _RPTF2(_CRT_WARN, "remote PF is set to %s:%d\n", lpReq->pvar->fwd_state.reque
```

```
sts[n].spec.to_host,
                lpReq->pvar->fwd_state.requests[n].spec.to_port );

        char buf[1024];
        _snprintf( buf, sizeof( buf ), "PORT 127,0,0,1,%d,%d\r\n",
                lpReq->pvar->fwd_state.requests[n].spec.from_port / 256,
                lpReq->pvar->fwd_state.requests[n].spec.from_port % 256 );
        SSH_channel_send(lpReq->pvar, lpReq->pvar->fwd_state.channels[lpReq->localch]
.remote_num,
                (unsigned char *)buf, strlen( buf ) );
        _RPTF1(_CRT_WARN, "sent to server: %s", buf );

        return 1;
}


/*
 *  user:
 */

void cmd_user( PREQUEST lpReq ) {
        char *sep_ptr;

        sep_ptr = strchr(lpReq->ftp_parm[1], '@');
        if( sep_ptr == NULL ||
                *(sep_ptr + 1) == '\0' ) {
                ftp_msgf( lpReq, 501, "Syntax error in parameters or arguments." );
                return;
        }
        if( sep_ptr - lpReq->ftp_parm[1] > 0 ) {
                MyStrNCpy2( lpReq->UserName, lpReq->ftp_parm[1], sep_ptr - lpReq->ftp
_parm[1] );
        } else {
                lpReq->UserName[0] = '\0';
        }
        MyStrNCpy2( lpReq->HostName, sep_ptr + 1, 0 );

        _RPTF0(_CRT_WARN, "calling SCM_connect\n");
        SCM_connect(g_hWnd, lpReq);
        lpReq->state = ST_CONNECTING;

  return;
}

void cmd_user2( PREQUEST lpReq, int result ) {
        if( lpReq->state != ST_CONNECTING )
                return;

        _RPTF0(_CRT_WARN, "called cmd_user2()\n");

        if( lpReq->pvar->fwd_state.in_interactive_mode ) {
                _RPTF0(_CRT_WARN, "SSH interactive session entered.\n");

                //
                // add FTP cmd port forwarding
                //

                FWDRequestSpec *requests;
                int i;
                int j = lpReq->pvar->fwd_state.num_requests + 1;

            requests = (FWDRequestSpec *)malloc(sizeof(FWDRequestSpec) * j);
                for( i = 0; i < (j - 1); i++ ) {
                        memcpy( (void *)&requests[i], &(lpReq->pvar->fwd_state.reques
ts[i].spec), sizeof( FWDRequestSpec ) );
                }
```



```
                requests[j-1].from_port = 1;
                MyStrNCpy2(requests[j-1].to_host, "localhost", 0 );
                requests[j-1].to_port = 21;
                requests[j-1].type = FWD_LOCAL_TO_REMOTE;
                requests[j-1].type2 = 1;

            qsort(requests, j, sizeof(FWDRequestSpec), FWD_compare_specs);

                for (i = 0; i < j - 1; i++) {
                  if (FWD_compare_specs(requests + i, requests + i + 1) == 0) {
                        memmove(requests + i, requests + i + 1, sizeof(FWDRequestSpec
)*(j - 1 - i));
                        i--;
                        j--;
                  }
                }

                FWD_set_request_specs(lpReq->pvar, requests, j);
                FWDUI_save_settings(lpReq->pvar);

                free(requests);

                _RPTF0(_CRT_WARN, "local PF request for FTP cmd port was registered.\
n");

                // SSH_open_channel (accept)の処理

                int channel_num;
                int request_num;
                int foundRequestSpec = 0;
                FWDRequest * request = lpReq->pvar->fwd_state.requests;

                // remote PF: set  to_port2
                for( i = 0; i < lpReq->pvar->fwd_state.num_requests; i++ ) {
                        if( (request[i].status & FWD_DELETED) == 0 &&
                                request[i].spec.type == FWD_REMOTE_TO_LOCAL ) {
                                request[i].spec.to_port2 = request[i].spec.to_port;
                        }
                }

                for( i = 0; i < lpReq->pvar->fwd_state.num_requests; i++ ) {
                        if( (request[i].status & FWD_DELETED) == 0 &&
                                request[i].spec.from_port == 1 &&
                                strcmp(request[i].spec.to_host, "localhost") == 0 &&
                                request[i].spec.to_port == 21 &&
                                request[i].spec.type == FWD_LOCAL_TO_REMOTE &&
                                request[i].spec.type2 == 1 ) {
                                foundRequestSpec++;
                                break;
                        }
                }
                if( ! foundRequestSpec ) {
                        ftp_msgf( lpReq, 530, "Proxy failed to connect. (SSH cannot r
egister local PF request)" );
                        return;
                }

                request_num = i;
                request = &lpReq->pvar->fwd_state.requests[request_num];
                FWDChannel * channel;

                channel_num = alloc_channel(lpReq->pvar, FWD_LOCAL_CONNECTED, request
_num);
                channel = lpReq->pvar->fwd_state.channels + channel_num;
                lpReq->localch = channel_num;
```

```
                channel->local_socket = INVALID_SOCKET;
                channel->filter_closure = NULL;
                channel->filter = NULL;

                SSH_open_channel(lpReq->pvar, channel_num, request->spec.to_host,
                        request->spec.to_port, "127.0.0.1");

                _RPTF0(_CRT_WARN, "local PF channel for FTP cmd port was opened.\n");

        } else {
                lpReq->state = ST_SEND_RESP;
                switch( result ) {
                case 2:
                        ftp_msgf( lpReq, 530, "Proxy failed to connect. (SSH invalid
host)" );
                        break;
                case 3:
                        ftp_msgf( lpReq, 530, "Proxy failed to connect. (SSH cannot c
onnect)" );
                        break;
                case 4:
                        ftp_msgf( lpReq, 530, "Proxy failed to connect. (SSH connecti
on closed)" );
                        break;
                default:
                        break;
                }
        }

        return;
}

void cmd_user3( PREQUEST lpReq, int result ) {
        if( lpReq->state != ST_CONNECTING )
                return;

        _RPTF0(_CRT_WARN, "called cmd_user3()\n");
        _RPTF0(_CRT_WARN, "openning local PF channel for FTP cmd port was confirmed.\
n");

        lpReq->state = ST_CONNECTING2;
}
```

```
#include <stdio.h>
#include <sys/types.h>
#include <string.h>
#include <ctype.h>
#include "ftpwrap.h"


int ftp_parse( PREQUEST );
void ftp_lower( char * );


/*
 *  get_command:    Extract the command from an input line from the client
 */


int ftp_get_command( PREQUEST lpReq ) {
        int command_found = 0, arg_valid_flag = 1;
        int ret_flag = 1;

    /*  Parse the message into the parameter array */
        lpReq->parm_count = ftp_parse( lpReq );

        if( lpReq->state == ST_SEND_RESP ) {
            if( lpReq->parm_count < 0 ) {
                        ftp_msgf( lpReq, 500, "Syntax error, command unrecognized." )
;
                } else {
                        if( strcmp(lpReq->ftp_command, "user") == 0 ) {
                                command_found++;
                                if( lpReq->parm_count < 1 ) {
                                        arg_valid_flag = 0;
                                } else if( lpReq->parm_count > 1 ) {
                                        arg_valid_flag = 2;
                                } else {
                                        cmd_user( lpReq );
                                }
                        } else if( strcmp(lpReq->ftp_command, "help") == 0 ) {
                                command_found++;
                                cmd_help( lpReq );
                        } else if( strcmp(lpReq->ftp_command, "quit") == 0 ) {
                                command_found++;
                                cmd_quit( lpReq );
                                ret_flag = 0;
                        } else if( strcmp(lpReq->ftp_command, "noop") == 0 ) {
                                command_found++;
                                if( lpReq->parm_count > 0 ) {
                                        arg_valid_flag = 2;
                                } else {
                                        cmd_noop( lpReq );
                                }
                        }

                        /*  Were too few parameters passed to the command? */
                        if( arg_valid_flag == 0 ) {
                                ftp_msgf(lpReq, 501,
                                        "Too few arguments for the %.128s command.",l
pReq->ftp_command);
                        } else if( arg_valid_flag == 2 ) {
                                /*  Were too many parameters passed to the command? *
/
                                ftp_msgf(lpReq, 501,
                                        "Too many arguments for the %.128s command.",
lpReq->ftp_command);
                        }
```



```
                        if( ! command_found ) {
                                /*  The client command was not located in the command
/state table */
                                ftp_msgf( lpReq, 500, "Syntax error, command unrecogn
ized: \"%.128s\".",
                                        lpReq->ftp_command );
                        }
                }
        } else { // ST_CONNECT
            if( lpReq->parm_count == 1 && strcmp(lpReq->ftp_command, "port") == 0 ) {
                        if( cmd_port( lpReq ) )
                                ret_flag++;
                }
        }

        return( ret_flag );
}




/*
 *  lower:  Convert a string to lowercase
 */

void ftp_lower( char *buf ) {
  char *mp;

  for( mp = buf; *mp; mp++ )
    if( isupper( (int)*mp ) && isupper( (int)*mp ) )
      *mp = tolower( (int)*mp );
  return;
}


/*
 *  parse:  Parse a raw input line from the client
 *  into null-delimited tokens
 */

int ftp_parse( PREQUEST lpReq ) {
  char *mp = lpReq->ftp_cmdline_buf;
  int    i;

  /*  Loop through the command array */
  for( i = 0; ; i++ ) {

    /*  Skip leading spaces and tabs in the message */
    while( isspace( *mp ) )
      mp++;

    /*  Are we at the end of the message? */
    if( *mp == '\0' )
      break;

    /*  Have we already obtained the maximum allowable parameters? */
    if( i > MAXPARMCOUNT ) {
                if( lpReq->state == ST_SEND_RESP )
                        ftp_msgf( lpReq, 501, "Too many arguments supplied." );
                return( -1 );
    }

    /*  Point to the start of the token */
    lpReq->ftp_parm[i] = mp;

    /*  Search for the first space character (end of the token) */
```

```
    while( ! isspace( *mp ) && *mp )
      mp++;

    /*  Delimit the token with a null */
    if( *mp )
      *mp++ = 0;

    if( i == 0 ) {
      /*  Convert the first token (command) to lower case */
      ftp_lower( lpReq->ftp_command );
      /*
       * This is kinda gross.  Passwords have to be parsed diffrently
       * as they may contain spaces.  If you think of a cleaner way,
       * do it.  The "p->sqlld_command[0] == 'p'" is so save a call to
       * strcmp() on ever call to sqlld_parse();  This parsing keeps
       * leading and trailing speces behind for the password command.
       */
      if( lpReq->ftp_command[0] == 'p' &&
          strcmp( lpReq->ftp_command, "pass" ) == 0 ) {
        if( *mp != 0 ) {
          lpReq->ftp_parm[1] = mp;
          if( strlen( mp ) > 0 ) {
            mp = mp + strlen( mp ) - 1;
            while( *mp == '\r' || *mp == '\n' )
              *mp-- = 0;
          }
          return( 1 );
        } else
          return( -1 );
      }
    }
  }

  /*  Were any parameters passed at all? */
  if( i == 0 )
    return( -1 );

  /*  Return the number of tokens extracted minus the command itself */
  return( i - 1 );

}



int ftp_get_resp( PREQUEST lpReq ) {
        unsigned int strofs;
        char statcodestr[4];
        char                *invalidchar;        /*  Detect strtoul() convert error *
/
        unsigned int curlen = strlen( lpReq->ftp_resp_str );

        _RPTF1(_CRT_WARN, "ftp_get_resp() entered, current resp_str len = %d\n", curl
en );

        if( lpReq->SSHRdDataSize < 6 )
                return 0;

        statcodestr[0] = lpReq->SSHRdBuff[0];
        statcodestr[1] = lpReq->SSHRdBuff[1];
        statcodestr[2] = lpReq->SSHRdBuff[2];
        statcodestr[3] = '\0';
        lpReq->ftp_resp_code = strtoul( statcodestr, &invalidchar, 10 );
        if( invalidchar == (char *)( statcodestr + 3 ) ) {
                // response code found

                int FoundLF = 1;
```



```
                unsigned int i;
                int FoundResponseEnd = 0;

                for( strofs = 0; strofs < lpReq->SSHRdDataSize; strofs++ ) {
                        if( FoundLF && strofs + 5 < lpReq->SSHRdDataSize &&
                                lpReq->SSHRdBuff[strofs+0] >= '0' && lpReq->SSHRdBuff
[strofs+0] <= '9' &&
                                lpReq->SSHRdBuff[strofs+1] >= '0' && lpReq->SSHRdBuff
[strofs+1] <= '9' &&
                                lpReq->SSHRdBuff[strofs+2] >= '0' && lpReq->SSHRdBuff
[strofs+2] <= '9' &&
                                lpReq->SSHRdBuff[strofs+3] == ' ' ) {
                                for( i = strofs + 4; i < lpReq->SSHRdDataSize; i++ )
{
                                        if( lpReq->SSHRdBuff[i] == '\n' ) {
                                                FoundResponseEnd++;
                                                break;
                                        }
                                }
                                break;
                        }
                        if( lpReq->SSHRdBuff[strofs] == '\n' )
                                FoundLF++;
                        else
                                FoundLF = 0;
                }

                if( FoundResponseEnd ) {
                        memcpy( lpReq->ftp_resp_str + curlen, lpReq->SSHRdBuff,
                                min( i + 1, sizeof( lpReq->ftp_resp_str ) - curlen -
1 ) );
                        lpReq->ftp_resp_str[curlen + min( i + 1, sizeof( lpReq->ftp_r
esp_str ) - curlen - 1 )] = '\0';
                        lpReq->SSHRdDataSize -= i + 1;
                        memmove( lpReq->SSHRdBuff, lpReq->SSHRdBuff + i + 1, lpReq->S
SHRdDataSize );
                        return 1;
                } else {
                        // buffer is not ended
                        return 0;
                }
        } else {
                // response code not found
                memcpy( lpReq->ftp_resp_str + curlen, lpReq->SSHRdBuff,
                        min( lpReq->SSHRdDataSize, sizeof( lpReq->ftp_resp_str ) - cu
rlen - 1 ) );
                lpReq->ftp_resp_str[min( lpReq->SSHRdDataSize, sizeof( lpReq->ftp_res
p_str ) - curlen - 1 )] = '\0';
                lpReq->SSHRdDataSize = 0;
                return 2;
        }
}
```

```
#include <stdio.h>
#include <sys/types.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>
#include <stdarg.h>
#include "ftpwrap.h"

#define BUFSIZE 2048


/*
 *  msgf:    Send formatted lines to the client
 */

void ftp_msgf( PREQUEST lpReq, int stat, const char *format,... ) {
  va_list ap;
  char message[BUFSIZE];
  char message2[BUFSIZE+8];
  char *mp = message;
  char statuscode[4];
  char *mp2;

  /*  Append the message (formatted, if necessary) */
  va_start( ap, format );
  if( format ) {
    _vsnprintf( message, sizeof( message ), format, ap );
  }
  va_end( ap );

  /*  Format the status code at the beginning of the message */
  if( stat > 999 )
    stat = 999;
  else if( stat < 0 )
    stat = 0;
  _snprintf( statuscode, sizeof( statuscode ), "%03d", stat );

  /* message display loop */
  do {
        MyStrNCpy2(message2, statuscode, 0);

    mp2 = strchr( mp, '\n' );

    if( mp2 == NULL ) { /* this is last line of message */
                MyStrNCat2(message2, " ", 0);
    } else { /* multi line message */
                MyStrNCat2(message2, "-", 0);
                *mp2 = '\0';
                if( mp2 > mp && *( mp2 - 1 ) == '\r' )
                        *( mp2 - 1 ) = '\0';
    }

    /* send a line of message */
        MyStrNCat2(message2, mp, 0);
    /* send <CR><LF> */
        MyStrNCat2(message2, "\r\n", 0);

        strncpy( (char *)(lpReq->WrBuff + lpReq->WrDataSize), message2,
                FTPWRBUFFSIZE - lpReq->WrDataSize - 1 );
    *(lpReq->WrBuff + FTPWRBUFFSIZE - 1) = '\0';

        lpReq->WrDataSize += strlen( message2 );

    mp = mp2 + 1;

  } while( mp2 != NULL );
```



```
  WriteData(lpReq);
  return;
}
```

```
// ftpwrap.cpp : アプリケーション用のエントリ ポイントの定義
//

#include "stdafx.h"
#include <winsock2.h>
#include <shellapi.h>
#include <commctrl.h>
#include <shlwapi.h>
#include <richedit.h>
#include "ftpwrap.h"

THREADDATA ThreadData = { NULL, NULL, NULL, 0, NULL, 0, NULL, NULL,
        { 0, 0, FALSE, 0, "\0", FALSE, FALSE, FALSE, FALSE, 0 }, NULL, { 0, 0, 0, 0},
        FALSE, NULL, NULL, NULL, { 0, 0, 0, 0 }, { RGB( 0, 0, 255 ), RGB( 0, 128, 0 )
, RGB( 0, 0, 0 ), RGB( 255, 0, 0 ) },
        NULL, FALSE, NULL, NULL, NULL, NULL, NULL };


HWND g_hWnd;

int APIENTRY WinMain( HINSTANCE hInstance,
                      HINSTANCE hPrevInstance,
                      LPSTR     lpCmdLine,
                      INT       nCmdShow ) {

//      SECURITY_ATTRIBUTES SecAttr;
        MSG msg;
        WNDCLASSEX wc;
        HWND hWnd;
        DWORD dwExitCode;
        INITCOMMONCONTROLSEX Icc;
        static TCHAR szAppName[256];
        PTHREADDATA pThreadData;
        DWORD dwRegDataSz;
        DWORD dwRegData;
        DWORD dwCreateKeyResult;
        BOOL bFirstExecute = FALSE;
        RECT RectMain, RectStatus;
        HINSTANCE hInstRichEdit;
        INT nX = CW_USEDEFAULT, nY = CW_USEDEFAULT, nWidth = 500, nHeight = 400;
        TCHAR Str[256];
        HKEY hRegKey;
        HMENU hMenu;

#ifdef _DEBUG
        _CrtSetReportMode(_CRT_WARN, _CRTDBG_MODE_DEBUG);
#endif

        pThreadData = &ThreadData;
        ThreadData.hInstance = hInstance;
        ThreadData.hRsrcInstance = hInstance;

        LOADSTR( IDS_APPNAME, szAppName );
        if ( ! hPrevInstance ) {
                wc.cbSize = sizeof( WNDCLASSEX );
                wc.style = CS_HREDRAW | CS_VREDRAW | CS_GLOBALCLASS;
                wc.lpfnWndProc = (WNDPROC)WndProc;
                wc.cbClsExtra = 0;
                wc.cbWndExtra = 0;
                wc.hInstance = hInstance;
                wc.hIcon = LoadIcon( hInstance, MAKEINTRESOURCE( IDI_ICON1 ) );
                wc.hCursor = LoadCursor( NULL, IDC_ARROW );
                wc.hbrBackground = (HBRUSH)GetStockObject( WHITE_BRUSH );
//              wc.lpszMenuName = MAKEINTRESOURCE( IDR_MENU1 );
                wc.lpszMenuName = NULL;
                wc.lpszClassName = szAppName;
```



```
                wc.hIconSm = (HICON)LoadImage( hInstance, MAKEINTRESOURCE( IDI_ICON1
), IMAGE_ICON, 16, 16, 0 );
                RegisterClassEx( &wc );
        }

        Icc.dwSize = sizeof( INITCOMMONCONTROLSEX );
        Icc.dwICC = ICC_HOTKEY_CLASS | ICC_INTERNET_CLASSES | ICC_PAGESCROLLER_CLASS
| ICC_BAR_CLASSES;
        InitCommonControlsEx( &Icc );

        if( RegOpenKeyEx( HKEY_CURRENT_USER, FTPWRAPREGKEY, 0, KEY_READ | KEY_WRITE,
&( pThreadData->hRegKey ) )
                == ERROR_SUCCESS ) {
                dwRegDataSz = sizeof( DWORD );
                // リモートのポートを取得
                if( RegQueryValueEx( pThreadData->hRegKey, FTPWRAPRemotePortREG, NULL
, NULL, (LPBYTE)&dwRegData, &dwRegDataSz )
                        == ERROR_SUCCESS )
                        pThreadData->Cfg.uRemotePort = (USHORT)dwRegData;
                // ローカルのポートを取得
                if( RegQueryValueEx( pThreadData->hRegKey, FTPWRAPLocalPortREG, NULL,
 NULL, (LPBYTE)&dwRegData, &dwRegDataSz )
                        == ERROR_SUCCESS )
                        pThreadData->Cfg.uLocalPort = (USHORT)dwRegData;
                // リモートからの接続を拒否するかの設定
                if( RegQueryValueEx( pThreadData->hRegKey, FTPWRAPRejectRemoteREG, NU
LL, NULL, (LPBYTE)&dwRegData, &dwRegDataSz )
                        == ERROR_SUCCESS )
                        pThreadData->Cfg.bRejectRemote = (BOOL)dwRegData;
                // リモートホスト指定の方法
                if( RegQueryValueEx( pThreadData->hRegKey, FTPWRAPRemoteAddrTypeREG,
NULL, NULL, (LPBYTE)&dwRegData, &dwRegDataSz )
                        == ERROR_SUCCESS )
                        pThreadData->Cfg.bRemoteAddrType = (BOOL)dwRegData;
                // リモート IP アドレス
                if( RegQueryValueEx( pThreadData->hRegKey, FTPWRAPRemoteAddrREG, NULL
, NULL, (LPBYTE)&dwRegData, &dwRegDataSz )
                        == ERROR_SUCCESS )
                        pThreadData->Cfg.uRemoteIP = (UINT)dwRegData;
                // ケロちゃんにおまかせ
                if( RegQueryValueEx( pThreadData->hRegKey, FTPWRAPKerberosREG, NULL,
NULL, (LPBYTE)&dwRegData, &dwRegDataSz )
                        == ERROR_SUCCESS )
                        pThreadData->Cfg.bKerberos = (BOOL)dwRegData;
                // 実行時に自動的にサービスを開始
                if( RegQueryValueEx( pThreadData->hRegKey, FTPWRAPStartServiceREG, NU
LL, NULL, (LPBYTE)&dwRegData, &dwRegDataSz )
                        == ERROR_SUCCESS )
                        pThreadData->Cfg.bStartService = (BOOL)dwRegData;
// TODO: 認証方式のプラグインからの読み込み等
                // 認証方式の種類
                if( RegQueryValueEx( pThreadData->hRegKey, FTPWRAPAuthMethodREG, NULL
, NULL, (LPBYTE)&dwRegData, &dwRegDataSz )
                        == ERROR_SUCCESS )
                        pThreadData->Cfg.uAuthMethod = (INT)dwRegData;
                // ウィンドウの位置(X 座標)
                if( RegQueryValueEx( pThreadData->hRegKey, FTPWRAPWindowXREG, NULL, N
ULL, (LPBYTE)&dwRegData, &dwRegDataSz )
                        == ERROR_SUCCESS ) {
                        if( (INT)dwRegData >= 0 && (INT)dwRegData < GetSystemMetrics(
 SM_CXSCREEN ) )
                                nX = (INT)dwRegData;
                        else
                                nX = CW_USEDEFAULT;             // 画面の外の場合はデ
フォルト値を使う
                }
```

```
                // ウィンドウの位置(Y 座標)
                if( RegQueryValueEx( pThreadData->hRegKey, FTPWRAPWindowYREG, NULL, N
ULL, (LPBYTE)&dwRegData, &dwRegDataSz )
                        == ERROR_SUCCESS ) {
                        if( (INT)dwRegData >= 0 && (INT)dwRegData < GetSystemMetrics(
 SM_CYSCREEN ) && nX != CW_USEDEFAULT )
                                nY = (UINT)dwRegData;
                        else
                                nX = nY = CW_USEDEFAULT;                // 画面の外の
場合はデフォルト値を使う
                }
                // ウィンドウの幅
                if( RegQueryValueEx( pThreadData->hRegKey, FTPWRAPWindowWidthREG, NUL
L, NULL, (LPBYTE)&dwRegData, &dwRegDataSz )
                        == ERROR_SUCCESS ) {
                        if( (INT)dwRegData < GetSystemMetrics( SM_CXSCREEN ) )
                                nWidth = (INT)dwRegData;
                        else
                                nWidth = GetSystemMetrics( SM_CXSCREEN );
                }
                // ウィンドウの高さ
                if( RegQueryValueEx( pThreadData->hRegKey, FTPWRAPWindowHeightREG, NU
LL, NULL, (LPBYTE)&dwRegData, &dwRegDataSz )
                        == ERROR_SUCCESS ) {
                        if( (INT)dwRegData < GetSystemMetrics( SM_CYSCREEN ) )
                                nHeight = (INT)dwRegData;
                        else
                                nHeight = GetSystemMetrics( SM_CYSCREEN );
                }
                // リモートホスト名
                dwRegDataSz = sizeof( pThreadData->Cfg.cRemoteName );
                RegQueryValueEx( pThreadData->hRegKey, FTPWRAPRemoteNameREG, NULL, NU
LL, (LPBYTE)pThreadData->Cfg.cRemoteName, &dwRegDataSz );
        } else {
                // 初回起動時
                ThreadData.bFirstExecute = TRUE;
                RegCreateKeyEx( HKEY_CURRENT_USER, FTPWRAPREGKEY, 0, NULL,
                        REG_OPTION_NON_VOLATILE, KEY_READ | KEY_WRITE, NULL, &( pThre
adData->hRegKey ), &dwCreateKeyResult );
        }

        if( RegOpenKeyEx( HKEY_CURRENT_USER, RunREGKEY, 0, KEY_READ, &hRegKey )
                == ERROR_SUCCESS ) {
                dwRegDataSz = sizeof( Str );
                // Windows 起動時に自動的に実行
                if( RegQueryValueEx( hRegKey, RunFTPWRAPREG, NULL, NULL, (LPBYTE)Str,
 &dwRegDataSz )
                        == ERROR_SUCCESS ) {
                        pThreadData->Cfg.bAutoRun = TRUE;
                        HOEHOEOS hoehoeOS;
                        HoehoeGetOSType( NULL, &hoehoeOS, NULL );
                        if( hoehoeOS.OSType == Win95 ) {
                                ThreadData.lpCmdLine = (LPTSTR)GlobalAlloc( GPTR, siz
eof( Str ) );
                                lstrcpy( ThreadData.lpCmdLine, Str );
                        }
                }
                RegCloseKey( hRegKey );
        }

        // メインウィンドウの作成
        hMenu = LoadMenu( ThreadData.hRsrcInstance, MAKEINTRESOURCE( IDR_MENU1 ) );
        hWnd = CreateWindowEx( 0, szAppName, szAppName, WS_OVERLAPPEDWINDOW | WS_CLIP
CHILDREN,
                nX, nY, nWidth, nHeight, NULL, hMenu, hInstance, NULL );
        ThreadData.hMainWnd = hWnd;
```



```
        g_hWnd = hWnd;
        ThreadData.hMainMenu = hMenu;

        // メニュー項目
        if( ThreadData.Cfg.bAutoRun )
                CheckMenuItem( ThreadData.hMainMenu, ID_MENUITEM_AUTORUN, MF_BYCOMMAN
D | MFS_CHECKED );
        if( ThreadData.Cfg.bStartService )
                CheckMenuItem( ThreadData.hMainMenu, ID_MENUITEM_AUTOSTARTSVC, MF_BYC
OMMAND | MFS_CHECKED );

        // ステータス行を追加
        LOADSTR( IDS_STATBAR_STOP, Str );
        ThreadData.hStatusWnd = CreateStatusWindow( WS_CHILD | WS_VISIBLE, Str, hWnd,
 WM_STATUSBAR );

        GetClientRect( hWnd, &RectMain );
        GetWindowRect( ThreadData.hStatusWnd, &RectStatus );

        // リッチエディットボックスを追加
        hInstRichEdit = LoadLibrary( RichEditDLL );
        if( ( ThreadData.hEditWnd = CreateWindowEx( 0, "RichEdit", NULL, WS_CHILD | W
S_VISIBLE | WS_BORDER | ES_READONLY | ES_MULTILINE | ES_AUTOVSCROLL | WS_VSCROLL | WS
_EX_OVERLAPPEDWINDOW,
                0, 0, RectMain.right, RectMain.bottom - ( RectStatus.bottom - RectSta
tus.top ),
                hWnd, NULL, hInstance, NULL ) ) == NULL ) {
                DispErrMsg( hWnd, "CreateWindowEx" );
        }
        // 背景色を設定
        SendMessage( ThreadData.hEditWnd, EM_SETBKGNDCOLOR, 0, RGB( 255, 255, 255 ) )
;
//      // IME 操作を無効にする
//      PostMessage( ThreadData.hEditWnd, EM_SETIMEOPTIONS, 0, IMF_FORCENONE );

        if( ThreadData.bFirstExecute )
                ShowWindow( hWnd, SW_SHOWNOACTIVATE );
        else {
                SendMessage( hWnd, WM_SYSCOMMAND, SC_MINIMIZE, 0 );
                ShowWindow( hWnd, SW_HIDE );
        }
        UpdateWindow( hWnd );


        // 実行時にサービスを開始
        if( ThreadData.Cfg.bStartService )
                SendMessage( hWnd, WM_COMMAND, ID_MENUITEM_STARTSVC, 0 );

        while ( GetMessage( &msg, NULL, 0, 0L ) ) {
                if( ( ThreadData.hMainWnd == NULL || ! IsDialogMessage( ThreadData.hM
ainWnd, &msg ) ) &&
                        ( ThreadData.hEditWnd == NULL || ! IsDialogMessage( ThreadDat
a.hEditWnd, &msg ) ) &&
                        ( ThreadData.hStatusWnd == NULL || ! IsDialogMessage( ThreadD
ata.hStatusWnd, &msg ) ) &&
                        ( ThreadData.hDlgWnd == NULL || ! IsDialogMessage( ThreadData
.hDlgWnd, &msg ) )
                        ) {
                        TranslateMessage( &msg );
                        DispatchMessage( &msg );
                }
        }

        FreeLibrary( hInstRichEdit );

        // ウィンドウの位置を保存
```

```
        dwRegDataSz = sizeof( DWORD );
        // X 座標
        if( pThreadData->RectMain.left >= 0 && pThreadData->RectMain.left < GetSystem
Metrics( SM_CXSCREEN ) ) {
                dwRegData = pThreadData->RectMain.left;
                RegSetValueEx( pThreadData->hRegKey, FTPWRAPWindowXREG, 0, REG_DWORD,
 (LPBYTE)&dwRegData, dwRegDataSz );
                // Y 座標
                if( pThreadData->RectMain.top >= 0 && pThreadData->RectMain.top < Get
SystemMetrics( SM_CYSCREEN ) ) {
                        dwRegData = pThreadData->RectMain.top;
                        RegSetValueEx( pThreadData->hRegKey, FTPWRAPWindowYREG, 0, RE
G_DWORD, (LPBYTE)&dwRegData, dwRegDataSz );
                        // 幅
                        dwRegData = pThreadData->RectMain.right - pThreadData->RectMa
in.left;
                        RegSetValueEx( pThreadData->hRegKey, FTPWRAPWindowWidthREG, 0
, REG_DWORD, (LPBYTE)&dwRegData, dwRegDataSz );
                        // 高さ
                        dwRegData = pThreadData->RectMain.bottom - pThreadData->RectM
ain.top;
                        RegSetValueEx( pThreadData->hRegKey, FTPWRAPWindowHeightREG,
0, REG_DWORD, (LPBYTE)&dwRegData, dwRegDataSz );
                }
        }

        RegCloseKey( pThreadData->hRegKey );

        if( GetExitCodeThread( ThreadData.hThread, &dwExitCode ) != STILL_ACTIVE ) {
                CloseHandle( ThreadData.hThread );
        } else {
                TerminateThread( ThreadData.hThread, -1 );
                CloseHandle( ThreadData.hThread );
        }

        return ThreadData.iReturnValue;
}


HMENU CreateContextMenu( PTHREADDATA pThreadData ) {
        HMENU hMenu;
        static MENUITEMINFO mii[7];
        static CHAR PopMenuString[7][64];
        INT n;

        hMenu = CreatePopupMenu();

        LOADSTR( IDS_POPMENU_RUN, PopMenuString[0] );
        mii[0].wID = ID_MENUITEM_STARTSVC;
        LOADSTR( IDS_POPMENU_STOP, PopMenuString[1] );
        mii[1].wID = ID_MENUITEM_ENDSVC;
        LOADSTR( IDS_POPMENU_CFG, PopMenuString[3] );
        mii[3].wID = ID_MENUITEM_SETUP;
        LOADSTR( IDS_POPMENU_SHOW, PopMenuString[4] );
        mii[4].wID = IDS_POPMENU_SHOW;
        LOADSTR( IDS_POPMENU_EXIT, PopMenuString[6] );
        mii[6].wID = ID_MENUITEM_EXIT;
        for( n = 0; n < 7; n++ ) {
                if( n == 2 || n == 5 ) {
                        mii[n].fMask = MIIM_TYPE;
                        mii[n].fType = MFT_SEPARATOR;
                        continue;
                }
                mii[n].fMask = MIIM_TYPE | MIIM_ID | MIIM_STATE;
                mii[n].fType = MFT_STRING;
                mii[n].fState = MFS_ENABLED;
```



```
                mii[n].dwTypeData = PopMenuString[n];
                mii[n].cch = lstrlen( mii[n].dwTypeData );
        }

        // サービスの状態によってメニューを無効化
        if( pThreadData->sListen == NULL ) {
                mii[1].fState = MFS_GRAYED;
                mii[0].fState = MFS_ENABLED | MFS_DEFAULT;
//              SetMenuDefaultItem( hMenu, 0, TRUE );
        } else {
                mii[0].fState = MFS_GRAYED;
                mii[1].fState = MFS_ENABLED | MFS_DEFAULT;
//              SetMenuDefaultItem( hMenu, 1, TRUE );
        }

        // 設定ダイアログが開いている時は [ 設定 ] を無効化
        if( pThreadData->hDlgWnd != NULL && IsWindow( pThreadData->hDlgWnd ) )
                mii[3].fState = MFS_GRAYED;

        // メインウィンドウが見えている時は [ ウィンドウを開く ] を無効化
        if( ! IsIconic( pThreadData->hMainWnd ) )
                mii[4].fState = MFS_GRAYED;

        for( n = 0; n < 7; n++ ) {
                mii[n].cbSize = sizeof( MENUITEMINFO );
                InsertMenuItem( hMenu, n, TRUE, &mii[n] );
        }

// サービスの開始(R)
// サービスの停止(S)
// ---------------
// 設定(C)
// ウィンドウを開く(O)
// ---------------
// プログラムの終了(X)

        return hMenu;
}


VOID ShowMainWindow( PTHREADDATA pThreadData ) {
        DWORD dwActiveTID;
        HWND hForeWnd;
        HWND hWnd = pThreadData->hMainWnd;

        ShowWindow( hWnd, SW_SHOW );
        UpdateWindow( hWnd );

// TODO: Maximized の時は発行しない -> OK?
        if( ! IsZoomed( hWnd ) )
                SendMessage( hWnd, WM_SYSCOMMAND, SC_RESTORE, 0 );

        // アクティブスレッドの ThreadID を取得する
        if( ( hForeWnd = GetForegroundWindow() ) != NULL ) {
                dwActiveTID = GetWindowThreadProcessId( hForeWnd, NULL );
                if( dwActiveTID == 0 ) {
                        DispErrMsg( pThreadData->hMainWnd, "GetWindowThreadProcessId"
 );
                }

                // 自分がアクティブスレッドなら
                if ( GetCurrentThreadId() == dwActiveTID ) {
                        // 普通に前面に持ってくる
                        SetForegroundWindow( hWnd );
        //              BringWindowToTop( hFocusWnd );
                } else {
```

```
                        // 自分がアクティブスレッドでないなら
                        // アクティブスレッドの入力処理をアタッチする
                        if( ! AttachThreadInput( GetCurrentThreadId(), dwActiveTID, T
RUE ) ) {
                                DispErrMsg( pThreadData->hMainWnd, "AttachThreadInput
" );
                        }
                        __try {
        //              try {
                                // アクティブスレッドの入力処理をアタッチしているので
                                // 自分はアクティブスレッド扱いになり受け付けてくれる
                                SetForegroundWindow( hWnd );
                        } __finally {
                //              } catch(...) {
                                // アクティブスレッドの入力処理をデタッチする
                                if( ! AttachThreadInput( GetCurrentThreadId(), dwActi
veTID, FALSE ) ) {
                                        DispErrMsg( pThreadData->hMainWnd, "AttachThr
eadInput" );
                                }
        //                      throw;
                        }
                }
        } else
                SetForegroundWindow( hWnd );
        BringWindowToTop( hWnd );
        return;
}


LRESULT CALLBACK WndProc( HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam ) {
        static PTHREADDATA pThreadData;
        NOTIFYICONDATA SysTrayData;
        static LPTSTR pWndTitle;
        HICON hIcon;
//      DWORD dwThreadId;
        INT iReturn;
//      HANDLE hSync1Event;
        RECT RectStatus;
        SECURITY_ATTRIBUTES SecAttr;
        HANDLE hMutex;
        TCHAR Str[256], Str2[64];
        DWORD dwRegData, dwRegDataSz;
        DWORD dwCreateKeyResult;
        HKEY hRegKey;
        LPTSTR pCmdLine, pBaseName;
        INT n;
        UINT uStrOffset;
        ULONG uIOParam = 0;
        BOOL bSocketErr = FALSE;
        static LPARAM buttondown = 0;

        switch( uMsg ) {
        case WM_CREATE:
//              pThreadData = (PTHREADDATA)lParam;
                pThreadData = &ThreadData;

//              hIcon = LoadIcon( AfxGetInstanceHandle(), MAKEINTRESOURCE( IDI_ICON3
) );
//              PostMessage( hWnd, WM_SETICON, TRUE, (LPARAM)hIcon );
//              DestroyIcon( hIcon );

                // 二重起動チェーック
                SecAttr.nLength = sizeof( SecAttr );
                SecAttr.lpSecurityDescriptor = NULL;
                SecAttr.bInheritHandle = FALSE;
```



```
                hMutex = CreateMutex( &SecAttr, FALSE, AppID );
                if( GetLastError() == ERROR_ALREADY_EXISTS ) {
                        LOADSTR( IDS_ERR_EXECMULTIPLE, Str );
                        LOADSTR( IDS_CAPTION_ERR, Str2 );
                        MessageBox( hWnd, Str, Str2, MB_OK | MB_ICONERROR | MB_SETFOR
EGROUND );
                        SendMessage( hWnd, WM_DESTROY, 0, 0 );
                        break;
                }

                // システムトレイにアイコンを追加
                SysTrayData.cbSize = sizeof( SysTrayData );
                SysTrayData.hWnd = hWnd;
                LOADSTR( IDS_APPNAME, SysTrayData.szTip );
                SysTrayData.uFlags = NIF_ICON | NIF_TIP | NIF_MESSAGE;
                SysTrayData.uCallbackMessage = WM_SYSTRAYICON;
                SysTrayData.uID = ID_SYSTRAYICON1;
                hIcon = (HICON)LoadImage( pThreadData->hInstance, MAKEINTRESOURCE( ID
I_ICON3 ), IMAGE_ICON, 16, 16, 0 );
//              hIcon = LoadIcon( pThreadData->hInstance, MAKEINTRESOURCE( IDI_ICON1
) );
                SysTrayData.hIcon = hIcon;
                Shell_NotifyIcon( NIM_ADD, &SysTrayData );
                DestroyIcon( hIcon );
                pThreadData->bSysTrayIcon = TRUE;

                // 初回起動時はセットアップダイアログを開く
                if( pThreadData->bFirstExecute ) {
                        if( ( iReturn = DialogBoxParam( pThreadData->hInstance, MAKEI
NTRESOURCE( IDD_SETUPDLG ),
                                hWnd, (DLGPROC)SetupDlgProc, (LPARAM)pThreadData ) )
== -1 ) {
                                DispErrMsg( hWnd, "DialogBoxParam" );
                        }
                }
                break;

        case WM_COMMAND:
                switch( LOWORD( wParam ) ) {
                case ID_MENUITEM_EXIT:
                        DestroyWindow( pThreadData->hMainWnd );
                        break;

                case ID_MENUITEM_SETUP:
                        if( ( iReturn = DialogBoxParam( pThreadData->hInstance, MAKEI
NTRESOURCE( IDD_SETUPDLG ),
                                hWnd, (DLGPROC)SetupDlgProc, (LPARAM)pThreadData ) )
== -1 ) {
                                DispErrMsg( hWnd, "DialogBoxParam" );
                        }
                        break;

                case ID_MENUITEM_VERSION:
                        if( ( iReturn = DialogBoxParam( pThreadData->hInstance, MAKEI
NTRESOURCE( IDD_VERSIONDLG ),
                                hWnd, (DLGPROC)VersionDlgProc, (LPARAM)pThreadData )
) == -1 ) {
                                DispErrMsg( hWnd, "DialogBoxParam" );
                        }
                        break;

                case ID_MENUITEM_AUTORUN:
                        // 起動時に自動的に実行
                        ThreadData.Cfg.bAutoRun = ! ThreadData.Cfg.bAutoRun;
                        if( RegCreateKeyEx( HKEY_CURRENT_USER, RunREGKEY, 0, NULL,
                                REG_OPTION_NON_VOLATILE, KEY_WRITE, NULL, &hRegKey, &
```

```
dwCreateKeyResult ) != ERROR_SUCCESS )
                                DispErrMsg( hWnd, "RegCreateKeyEx" );
                        if( ThreadData.Cfg.bAutoRun ) {
                                CheckMenuItem( ThreadData.hMainMenu, ID_MENUITEM_AUTO
RUN, MF_BYCOMMAND | MFS_CHECKED );

                                GetCurrentDirectory( sizeof( Str ), Str );
                                MakeFullPath( Str, Str, FTPWRAPPATH );
//                              MessageBox( hWnd, Str, "current", MB_OK );
                                if( ! PathFileExists( Str ) ) {
                                        // プログラムへのフルパスを作成する
                                        pCmdLine = GetCommandLine();
                                        // quote を取り除く
                                        Str[0] = '\0';
                                        for( n = 0; n < lstrlen( pCmdLine ); n++ ) {
                                                if( *(pCmdLine + n) == '"' ) {
                                                        uStrOffset = ++n;
                                                        for( ; n < lstrlen( pCmdLine
); n++ ) {
                                                                if( *(pCmdLine + n) =
= '"' ) {
                                                                        StrNCat( Str,
 pCmdLine + uStrOffset, n - uStrOffset + 1 );
                                                                        uStrOffset =
n + 1;
                                                                        break;
                                                                }
                                                        }
                                                }
                                        }
                                        lstrcat( Str, pCmdLine + uStrOffset );
                                        pBaseName = StrRChr( Str, Str + lstrlen( Str
), '\\' ) + 1;
                                        if( pBaseName == NULL )
                                                pBaseName = Str;
                                        lstrcpy( pBaseName, FTPWRAPPATH );
//                                      MessageBox( hWnd, Str, "cmdline", MB_OK );

                                        // Win95: Run では GetCommandLine が...
                                        if( ! PathFileExists( Str ) && pThreadData->l
pCmdLine != NULL ) {
                                                lstrcpy( Str, pThreadData->lpCmdLine
);
//                                              MessageBox( hWnd, Str, "registry", MB
_OK );
                                        }
                                }

                                dwRegDataSz = lstrlen( Str ) + 1;
                                if( RegSetValueEx( hRegKey, RunFTPWRAPREG, 0, REG_SZ,
 (LPBYTE)Str, dwRegDataSz ) != ERROR_SUCCESS )
                                        DispErrMsg( hWnd, "RegSetValueEx" );

                                // 有効にされたら，同時に「実行時に自動的にサービスを
開始」も有効にする
                                ThreadData.Cfg.bStartService = TRUE;
                                CheckMenuItem( ThreadData.hMainMenu, ID_MENUITEM_AUTO
STARTSVC, MF_BYCOMMAND | MFS_CHECKED );
                                dwRegData = (DWORD)TRUE;
                                dwRegDataSz = sizeof( DWORD );
                                RegSetValueEx( pThreadData->hRegKey, FTPWRAPStartServ
iceREG, 0, REG_DWORD, (LPBYTE)&dwRegData, dwRegDataSz );
                        } else {
                                CheckMenuItem( ThreadData.hMainMenu, ID_MENUITEM_AUTO
RUN, MF_BYCOMMAND | MFS_UNCHECKED );
                                RegDeleteValue( hRegKey, RunFTPWRAPREG );
```



```
                        }
                        RegCloseKey( hRegKey );
                        break;

                case ID_MENUITEM_AUTOSTARTSVC:
                        // 実行時に自動的にサービスを開始
                        ThreadData.Cfg.bStartService = ! ThreadData.Cfg.bStartService
;
                        CheckMenuItem( ThreadData.hMainMenu, ID_MENUITEM_AUTOSTARTSVC
, MF_BYCOMMAND | ( ThreadData.Cfg.bStartService ? MFS_CHECKED : MFS_UNCHECKED ) );
                        dwRegDataSz = sizeof( DWORD );
                        dwRegData = (DWORD)pThreadData->Cfg.bStartService;
                        RegSetValueEx( pThreadData->hRegKey, FTPWRAPStartServiceREG,
0, REG_DWORD, (LPBYTE)&dwRegData, dwRegDataSz );
                        break;

                case ID_MENUITEM_STARTSVC:
                        WORD wVersionRequested;
                        WSADATA wsaData;

                        wVersionRequested = MAKEWORD( 1, 1 );
                        if ( WSAStartup( wVersionRequested, &wsaData ) != 0 ) {
                                LOGNETERR( "WSAStartup:" );
                                return 1;
                        }

//                      wsprintf( Str, "Winsock のバージョンは %d.%d です．", HIBYTE(
 wsaData.wVersion ), LOBYTE( wsaData.wVersion ) );
//                      MessageBox( hWnd, Str, "winsock version", MB_OK );

                        if( HIBYTE( wsaData.wVersion ) < 1 ) {
                                WSACleanup();
                                return 1;
                        }

                        if( ( pThreadData->sListen = socket( AF_INET, SOCK_STREAM, IP
PROTO_TCP ) ) == INVALID_SOCKET ) {
                                pThreadData->sListen = NULL;
                                LOGNETERR( "socket:" );
                                return 1;
                        }

                        pThreadData->saListen.sin_family = AF_INET;
                        pThreadData->saListen.sin_port = htons( pThreadData->Cfg.uLoc
alPort );
                        if( pThreadData->Cfg.bRejectRemote )
                                pThreadData->saListen.sin_addr.s_addr = htonl( MAKEIP
ADDRESS( 127, 0, 0, 1 ) );
                        else
                                pThreadData->saListen.sin_addr.s_addr = INADDR_ANY;
                        if( bind( pThreadData->sListen, (LPSOCKADDR)&( pThreadData->s
aListen ), sizeof( pThreadData->saListen ) ) == SOCKET_ERROR ) {
                                LOGNETERR( "bind:" );
                                bSocketErr = TRUE;
                        } else if( listen( pThreadData->sListen, 5 ) == SOCKET_ERROR
) {
                                LOGNETERR( "listen:" );
                                bSocketErr = TRUE;
                        } else if( WSAAsyncSelect( pThreadData->sListen, pThreadData-
>hMainWnd, WM_NETEVENT, FD_ACCEPT ) == SOCKET_ERROR ) {
                                LOGNETERR( "WSAAsyncSelect:" );
                                bSocketErr = TRUE;
                        }
                        if( bSocketErr ) {
                                closesocket( pThreadData->sListen );
                                pThreadData->sListen = NULL;
```

```
                                return 1;
                        }

                        LOADSTR2( IDS_INFO_STARTSVC, Str );
                        LOGMSG( LOG_INFO, Str, NULL );
                        // メニューを変更
                        EnableMenuItem( pThreadData->hMainMenu, ID_MENUITEM_STARTSVC,
 MF_BYCOMMAND | MF_GRAYED );
                        EnableMenuItem( pThreadData->hMainMenu, ID_MENUITEM_ENDSVC, M
F_BYCOMMAND | MF_ENABLED );
                        // ステータス行を変更
                        LOADSTR( IDS_STATBAR_IDLE, Str );
                        SendMessage( pThreadData->hStatusWnd, SB_SETTEXT, 0, (LPARAM)
Str );
                        break;

                case ID_MENUITEM_ENDSVC:
                        // 接続待ちのソケットを閉じる
                        closesocket( pThreadData->sListen );
//                      WSACleanup();
                        pThreadData->sListen = NULL;

                        LOADSTR2( IDS_INFO_ENDSVC, Str );
                        LOGMSG( LOG_INFO, Str, NULL );
                        // メニューを変更
                        EnableMenuItem( pThreadData->hMainMenu, ID_MENUITEM_STARTSVC,
 MF_BYCOMMAND | MF_ENABLED );
                        EnableMenuItem( pThreadData->hMainMenu, ID_MENUITEM_ENDSVC, M
F_BYCOMMAND | MF_GRAYED );
                        // ステータス行を変更
                        LOADSTR( IDS_STATBAR_STOP, Str );
                        PostMessage( pThreadData->hStatusWnd, SB_SETTEXT, 0, (LPARAM)
Str );
                        break;

                case IDS_POPMENU_SHOW:
                        ShowMainWindow( pThreadData );
                        break;

                default:
                        PTInstVar pvar;
                        pvar = GetFirstRequest();
                        _RPTF0(_CRT_WARN, "TTXProcessCommand\n");
                        if (pvar && TTXProcessCommand(pvar,hWnd, LOWORD(wParam))) ret
urn TRUE;
                        break;
                }
                break;


        case WM_USER_COMMNOTIFY:
                PTInstVar pvar;
                pvar = GetRequest((SOCKET)wParam);
                switch (WSAGETSELECTEVENT(lParam)) {
                case FD_CONNECT:
                        if(pvar)
                                 CommStart(pvar,lParam);
                        break;

                case FD_WRITE:
                        if(pvar)
                                CommSend(pvar);
                        break;

                case FD_READ:
                        if(pvar)
```



```
                                CommProcRRQ(pvar);
                        break;

                case FD_CLOSE:
//                      Connecting = FALSE;
//                      TCPIPClosed = TRUE;
                        // disable transmition
//                              cv.OutBuffCount = 0;
//                              SetTimer(IdComEndTimer,1,NULL);
                        if(pvar)
                                CommClose(pvar);
                        break;
                }
                break;

        case WM_USER_NOTIFY_CONNRESULT:
                {
                // lParam = pvar wParam = result
                PREQUEST lpReq = FTPGetRequestByPVar((PTInstVar)lParam);
                if( lpReq != NULL )
                        cmd_user2( lpReq, (int)wParam );
                }
                break;

        case WM_USER_NOTIFY_CHOPENRESULT:
                {
                // lParam = pvar wParam = result
                PREQUEST lpReq = FTPGetRequestByPVar((PTInstVar)lParam);
                if( lpReq != NULL )
                        cmd_user3( lpReq, (int)wParam );
                }
                break;

        case WM_NETEVENT:
                {
                int nErrorCode = WSAGETSELECTERROR(lParam);

                switch(WSAGETSELECTEVENT(lParam)) {
                        case FD_ACCEPT:
                                OnAccept(hWnd, (SOCKET)wParam, nErrorCode);
                                break;

                        case FD_READ:
                                OnRead((SOCKET)wParam, nErrorCode);
                                break;

                        case FD_WRITE:
                                OnWrite((SOCKET)wParam, nErrorCode);
                                break;

                        case FD_CLOSE:
                                OnClose((SOCKET)wParam, nErrorCode);
                                break;
                }

                }
                break;

        case WM_STATUSBAR:
                break;

        case WM_SYSTRAYICON:
                switch( wParam ) {
                case ID_SYSTRAYICON1:
                        switch( lParam ) {
                        case WM_LBUTTONDOWN:
```

```
                        case WM_RBUTTONDOWN:
                                buttondown = lParam;
                                break;

                        // ポップアップメニュー
                        case WM_LBUTTONUP:
                                if( buttondown == WM_LBUTTONDOWN ) {
                                        HMENU hMenu;
                                        POINT pt;
                                        GetCursorPos( &pt );
                                        hMenu = CreateContextMenu( pThreadData );
                                        TrackPopupMenu( hMenu, TPM_BOTTOMALIGN, pt.x,
 pt.y, 0, hWnd, NULL );
                                        DestroyMenu( hMenu );
                                }
                                buttondown = 0;
                                break;

                        case WM_RBUTTONUP:
                                if( buttondown == WM_RBUTTONDOWN ) {
                                        HMENU hMenu;
                                        POINT pt;
                                        GetCursorPos( &pt );
                                        hMenu = CreateContextMenu( pThreadData );
                                        TrackPopupMenu( hMenu, TPM_BOTTOMALIGN, pt.x,
 pt.y, 0, hWnd, NULL );
                                        DestroyMenu( hMenu );
                                }
                                buttondown = 0;
                                break;

                        case WM_LBUTTONDBLCLK:
                        case WM_RBUTTONDBLCLK:
                                ShowMainWindow( pThreadData );
                                break;

                        default:
                                break;
                        }
                        break;

                default:
                        break;
                }

                break;

        case WM_SIZE:
                switch( wParam ) {
                case SIZE_MINIMIZED:
                        ShowWindow( hWnd, SW_HIDE );
                        UpdateWindow( hWnd );
                        break;

                case SIZE_MAXIMIZED:
                        GetWindowRect( pThreadData->hStatusWnd, &RectStatus );
                        ShowWindow( hWnd, SW_MAXIMIZE );
                        UpdateWindow( hWnd );
                        MoveWindow( pThreadData->hEditWnd, 0, 0, LOWORD( lParam ),
                                HIWORD( lParam ) - ( RectStatus.bottom - RectStatus.t
op ), TRUE );
                        MoveWindow( pThreadData->hStatusWnd, 0, HIWORD( lParam ) - (
 RectStatus.bottom - RectStatus.top ),
                                LOWORD( lParam ), RectStatus.bottom - RectStatus.top,
 TRUE );
                        break;
```



```
                case SIZE_RESTORED:
                        GetWindowRect( pThreadData->hStatusWnd, &RectStatus );
                        ShowWindow( hWnd, SW_SHOW );
                        UpdateWindow( hWnd );
                        MoveWindow( pThreadData->hEditWnd, 0, 0, LOWORD( lParam ),
                                HIWORD( lParam ) - ( RectStatus.bottom - RectStatus.t
op ), TRUE );
                        MoveWindow( pThreadData->hStatusWnd, 0, HIWORD( lParam ) - (
 RectStatus.bottom - RectStatus.top ),
                                LOWORD( lParam ), RectStatus.bottom - RectStatus.top,
 TRUE );
                        break;

                case SIZE_MAXSHOW:
                case SIZE_MAXHIDE:
                default:
                        ShowWindow( hWnd, SW_SHOW );
                        UpdateWindow( hWnd );
                        break;
                }
                DefWindowProc( hWnd, uMsg, wParam, lParam );
                break;

        case WM_DESTROY:
                // Run エントリ用のバッファを解放
                if( pThreadData->lpCmdLine != NULL )
                        GlobalFree( pThreadData->lpCmdLine );
// TODO: listen していなくても，以前に WSAStartup() されたら WSACleanup() する
                // 接続待ちのソケットを閉じる
                if( pThreadData->sListen != NULL ) {
                        closesocket( pThreadData->sListen );
                        WSACleanup();
                }

                // システムトレイのアイコンを削除
                if( pThreadData->bSysTrayIcon ) {
                        SysTrayData.cbSize = sizeof( SysTrayData );
                        SysTrayData.hWnd = hWnd;
                        SysTrayData.uFlags = 0;
                        SysTrayData.uID = ID_SYSTRAYICON1;
                        Shell_NotifyIcon( NIM_DELETE, &SysTrayData );
                }

                GetWindowRect( hWnd, &( pThreadData->RectMain ) );

                PostQuitMessage( 0 );
                break;

        default:
                return DefWindowProc( hWnd, uMsg, wParam, lParam );
        }
        return 0L;
}



LRESULT CALLBACK SetupDlgProc( HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam ) {
        static PTHREADDATA pThreadData;
        HWND hControlWnd;
//      INT iLength;
        DWORD dwRegDataSz, dwRegData;
        TCHAR Str[256], Str2[64];
        UINT uLocalPortTmp, uRemotePortTmp;
        BOOL bRestart = FALSE;
```

```
        switch ( uMsg ) {
        case WM_INITDIALOG:
                pThreadData = (PTHREADDATA)lParam;
                pThreadData->hDlgWnd = hWnd;
                // プルダウンメニューを拡張 UI に設定
                SendDlgItemMessage( hWnd, IDC_AUTHMETHOD, CB_SETEXTENDEDUI, TRUE, 0 )
;

                // プロトコル
//              SetDlgItemInt( hWnd, IDC_LOCALPORT, (UINT)pThreadData->Cfg.uLocalPort
, FALSE );

                // リモートポート番号
                SetDlgItemInt( hWnd, IDC_REMOTEPORT, (UINT)pThreadData->Cfg.uRemotePo
rt, FALSE );
//              hControlWnd = GetDlgItem( hWnd, IDC_REMOTEPORT );
//              PostMessage( hControlWnd, EM_LIMITTEXT, 5, 0 );
                SendDlgItemMessage( hWnd, IDC_REMOTEPORT, EM_LIMITTEXT, 5, 0 );
// TODO: プルダウンメニューなど

                // ローカルポート番号
                SetDlgItemInt( hWnd, IDC_LOCALPORT, (UINT)pThreadData->Cfg.uLocalPort
, FALSE );
//              hControlWnd = GetDlgItem( hWnd, IDC_LOCALPORT );
//              PostMessage( hControlWnd, EM_LIMITTEXT, 5, 0 );
                SendDlgItemMessage( hWnd, IDC_LOCALPORT, EM_LIMITTEXT, 5, 0 );
// TODO: プルダウンメニューなど

                // リモートからの接続を拒否
                if( pThreadData->Cfg.bRejectRemote )
                        CheckDlgButton( hWnd, IDC_REJECTREMOTE, BST_CHECKED );

                // リモートホスト名
//              hControlWnd = GetDlgItem( hWnd, IDC_REMOTENAME );
//              PostMessage( hControlWnd, EM_LIMITTEXT, sizeof( pThreadData->Cfg.cRem
oteName ) - 1, 0 );
                SendDlgItemMessage( hWnd, IDC_REMOTENAME, EM_LIMITTEXT, sizeof( pThre
adData->Cfg.cRemoteName ) - 1, 0 );
                SetDlgItemText( hWnd, IDC_REMOTENAME, pThreadData->Cfg.cRemoteName );

                // リモートホスト IP アドレス
//              hControlWnd = GetDlgItem( hWnd, IDC_REMOTEIP );
//              SendMessage( hControlWnd, IPM_SETADDRESS, 0, pThreadData->Cfg.uRemote
IP );
                SendDlgItemMessage( hWnd, IDC_REMOTEIP, IPM_SETADDRESS, 0, pThreadDat
a->Cfg.uRemoteIP );

                // リモートホスト指定の方法
                if( pThreadData->Cfg.bRemoteAddrType ) {
                        hControlWnd = GetDlgItem( hWnd, IDC_REMOTEIP );
                        EnableWindow( hControlWnd, FALSE );
                        CheckRadioButton( hWnd, IDC_REMOTEHOST_IP, IDC_REMOTEHOST_NAM
E, IDC_REMOTEHOST_NAME );
                } else {
                        hControlWnd = GetDlgItem( hWnd, IDC_REMOTENAME );
                        EnableWindow( hControlWnd, FALSE );
                        CheckRadioButton( hWnd, IDC_REMOTEHOST_IP, IDC_REMOTEHOST_NAM
E, IDC_REMOTEHOST_IP );
                }

//              // ケロちゃんにおまかせ
//              if( pThreadData->Cfg.bKerberos )
//                      CheckDlgButton( hWnd, IDC_KEROCHAN, BST_CHECKED );

//              hEditBoxWnd = GetDlgItem( hWnd, IDC_TEXTBOX );
//              iLength = GetWindowTextLength( hEditBoxWnd );
```



```
//              SendDlgItemMessage( hWnd, IDC_TEXTBOX, EM_SETSEL, iLength, iLength );
//              SetDlgItemText( hWnd, IDC_TEXTBOX, pBuf );
//              SendDlgItemMessage( hWnd, IDC_TEXTBOX, EM_REPLACESEL, FALSE, (LPARAM)
pBuf );

                // 選択解除
//              hEditBoxWnd = GetDlgItem( hWnd, IDC_TEXTBOX );
//              iLength = GetWindowTextLength( hEditBoxWnd );
//              PostMessage( hEditBoxWnd, EM_SETSEL, 0, 0 );

                break;

        case WM_COMMAND:
// TODO: プルダウンメニュー
                switch( LOWORD( wParam ) ) {
                case IDC_PROTOPOP3:
//                      hControlWnd = GetDlgItem( hWnd, IDC_LOCALPORT_STD );
//                      SetWindowText( hControlWnd, POP3_LOCALLABEL );
                        SetDlgItemText( hWnd, IDC_LOCALPORT_STD, POP3_LOCALLABEL );
//                      hControlWnd = GetDlgItem( hWnd, IDC_REMOTEPORT_STD );
//                      SetWindowText( hControlWnd, POP3_REMOTELABEL );
                        SetDlgItemText( hWnd, IDC_REMOTEPORT_STD, POP3_REMOTELABEL );
                        break;

                case IDC_PROTOIMAP4:
//                      hControlWnd = GetDlgItem( hWnd, IDC_LOCALPORT_STD );
//                      SetWindowText( hControlWnd, IMAP_LOCALLABEL );
                        SetDlgItemText( hWnd, IDC_LOCALPORT_STD, IMAP_LOCALLABEL );
//                      hControlWnd = GetDlgItem( hWnd, IDC_REMOTEPORT_STD );
//                      SetWindowText( hControlWnd, IMAP_REMOTELABEL );
                        SetDlgItemText( hWnd, IDC_REMOTEPORT_STD, IMAP_REMOTELABEL );
                        break;

                case IDC_REMOTEPORT_STD:
                        hControlWnd = GetDlgItem( hWnd, IDC_REMOTEPORT );
                        EnableWindow( hControlWnd, FALSE );
                        break;

                case IDC_REMOTEPORT_OTHER:
                        hControlWnd = GetDlgItem( hWnd, IDC_REMOTEPORT );
                        EnableWindow( hControlWnd, TRUE );
                        break;

                case IDC_LOCALPORT_STD:
                        hControlWnd = GetDlgItem( hWnd, IDC_LOCALPORT );
                        EnableWindow( hControlWnd, FALSE );
                        break;

                case IDC_LOCALPORT_OTHER:
                        hControlWnd = GetDlgItem( hWnd, IDC_LOCALPORT );
                        EnableWindow( hControlWnd, TRUE );
                        break;

                case IDC_REMOTEHOST_NAME:
                        hControlWnd = GetDlgItem( hWnd, IDC_REMOTENAME );
                        EnableWindow( hControlWnd, TRUE );
                        hControlWnd = GetDlgItem( hWnd, IDC_REMOTEIP );
                        EnableWindow( hControlWnd, FALSE );
                        break;

                case IDC_REMOTEHOST_IP:
                        hControlWnd = GetDlgItem( hWnd, IDC_REMOTENAME );
                        EnableWindow( hControlWnd, FALSE );
                        hControlWnd = GetDlgItem( hWnd, IDC_REMOTEIP );
                        EnableWindow( hControlWnd, TRUE );
                        break;
```

```
// TODO: プルダウンメニュー
                case IDC_AUTHMETHOD:
                        break;

                case IDOK:
//
// 設定を適用する
//
                        // ローカルポート番号
                        if( IsDlgButtonChecked( hWnd, IDC_LOCALPORT_STD ) == BST_CHEC
KED ) {
// TODO: プルダウンメニュー
//                              uLocalPortTmp = ( IsDlgButtonChecked( hWnd, IDC_PROTO
POP3 ) == BST_CHECKED ? POP3PORT_POP3 : IMAPPORT_IMAP );
                        } else {
                                uLocalPortTmp = GetDlgItemInt( hWnd, IDC_LOCALPORT, N
ULL, FALSE );
                                if( uLocalPortTmp > 65535 ) {
                                        LOADSTR( IDS_ERR_PORTNUM, Str );
                                        LOADSTR( IDS_CAPTION_ERR, Str2 );
                                        MessageBox( hWnd, Str, Str2, MB_ICONERROR | M
B_OK | MB_SETFOREGROUND );
                                        hControlWnd = GetDlgItem( hWnd, IDC_LOCALPORT
 );
                                        SetFocus( hControlWnd );
                                        break;
                                }
                        }

                        // リモートポート番号
                        if( IsDlgButtonChecked( hWnd, IDC_REMOTEPORT_STD ) == BST_CHE
CKED ) {
// TODO: プルダウンメニューなど
//                              pThreadData->Cfg.uRemotePort = ( IsDlgButtonChecked(
hWnd, IDC_PROTOPOP3 ) == BST_CHECKED ? POP3PORT_POP3 : IMAPPORT_IMAP );
                        } else {
                                uRemotePortTmp = GetDlgItemInt( hWnd, IDC_REMOTEPORT,
 NULL, FALSE );
                                if( uRemotePortTmp > 65535 ) {
                                        LOADSTR( IDS_ERR_PORTNUM, Str );
                                        LOADSTR( IDS_CAPTION_ERR, Str2 );
                                        MessageBox( hWnd, Str, Str2, MB_ICONERROR | M
B_OK | MB_SETFOREGROUND );
                                        hControlWnd = GetDlgItem( hWnd, IDC_REMOTEPOR
T );
                                        SetFocus( hControlWnd );
                                        break;
                                }
                                pThreadData->Cfg.uRemotePort = (USHORT)uRemotePortTmp
;
                        }
                        // ローカルポート番号を設定
                        if( pThreadData->Cfg.uLocalPort != (USHORT)uLocalPortTmp ) {
                                pThreadData->Cfg.uLocalPort = (USHORT)uLocalPortTmp;
                                bRestart = TRUE;
                        }


                        // 認証方式
                        if( ( pThreadData->Cfg.uAuthMethod = SendDlgItemMessage( hWnd
, IDC_AUTHMETHOD, CB_GETCURSEL, 0, 0 ) ) == CB_ERR )
                                pThreadData->Cfg.uAuthMethod = 0;

                        // リモートからの接続を拒否
                        if( pThreadData->Cfg.bRejectRemote != ( IsDlgButtonChecked( h
```



```
Wnd, IDC_REJECTREMOTE ) == BST_CHECKED ) ) {
                                        pThreadData->Cfg.bRejectRemote = IsDlgButtonChecked(
hWnd, IDC_REJECTREMOTE ) == BST_CHECKED;
                                        bRestart = TRUE;
                        }

                        // リモートホスト名
                        GetDlgItemText( hWnd, IDC_REMOTENAME, pThreadData->Cfg.cRemot
eName, sizeof( pThreadData->Cfg.cRemoteName ) );

                        // リモートホスト IP アドレス
//                      hControlWnd = GetDlgItem( hWnd, IDC_REMOTEIP );
//                      SendMessage( hControlWnd, IPM_GETADDRESS, 0, (LPARAM)&( pThre
adData->Cfg.uRemoteIP ) );
                        SendDlgItemMessage( hWnd, IDC_REMOTEIP, IPM_GETADDRESS, 0, (L
PARAM)&( pThreadData->Cfg.uRemoteIP ) );

                        // リモートホスト指定の方法
                        pThreadData->Cfg.bRemoteAddrType = IsDlgButtonChecked( hWnd,
IDC_REMOTEHOST_NAME ) == BST_CHECKED;

//                      // ケロちゃんにおまかせ
//                      pThreadData->Cfg.bKerberos = IsDlgButtonChecked( hWnd, IDC_KE
ROCHAN ) == BST_CHECKED;


//
// レジストリに設定を保存
//
                        dwRegDataSz = sizeof( DWORD );
                        // 認証方式
                        dwRegData = (DWORD)pThreadData->Cfg.uAuthMethod;
                        RegSetValueEx( pThreadData->hRegKey, FTPWRAPAuthMethodREG, 0,
 REG_DWORD, (LPBYTE)&dwRegData, dwRegDataSz );
                        // リモートのポート
                        dwRegData = (DWORD)pThreadData->Cfg.uRemotePort;
                        RegSetValueEx( pThreadData->hRegKey, FTPWRAPRemotePortREG, 0,
 REG_DWORD, (LPBYTE)&dwRegData, dwRegDataSz );
                        // ローカルのポート
                        dwRegData = (DWORD)pThreadData->Cfg.uLocalPort;
                        RegSetValueEx( pThreadData->hRegKey, FTPWRAPLocalPortREG, 0,
REG_DWORD, (LPBYTE)&dwRegData, dwRegDataSz );
                        // リモートからの接続を拒否するかの設定
                        dwRegData = (DWORD)pThreadData->Cfg.bRejectRemote;
                        RegSetValueEx( pThreadData->hRegKey, FTPWRAPRejectRemoteREG,
0, REG_DWORD, (LPBYTE)&dwRegData, dwRegDataSz );
                        // リモートホスト指定の方法
                        dwRegData = (DWORD)pThreadData->Cfg.bRemoteAddrType;
                        RegSetValueEx( pThreadData->hRegKey, FTPWRAPRemoteAddrTypeREG
, 0, REG_DWORD, (LPBYTE)&dwRegData, dwRegDataSz );
                        // リモート IP アドレス
                        dwRegData = (DWORD)pThreadData->Cfg.uRemoteIP;
                        RegSetValueEx( pThreadData->hRegKey, FTPWRAPRemoteAddrREG, 0,
 REG_DWORD, (LPBYTE)&dwRegData, dwRegDataSz );
                        // ケロちゃんにおまかせ
                        dwRegData = (DWORD)pThreadData->Cfg.bKerberos;
                        RegSetValueEx( pThreadData->hRegKey, FTPWRAPKerberosREG, 0, R
EG_DWORD, (LPBYTE)&dwRegData, dwRegDataSz );
                        // リモートホスト名
                        dwRegDataSz = lstrlen( pThreadData->Cfg.cRemoteName ) + 1;
                        RegSetValueEx( pThreadData->hRegKey, FTPWRAPRemoteNameREG, 0,
 REG_SZ, (LPBYTE)pThreadData->Cfg.cRemoteName, dwRegDataSz );

                        // 待機しているソケットを閉じて再待機
                        if( bRestart && pThreadData->sListen != NULL ) {
                                BOOL bRestartOK = TRUE;
```

```
                                closesocket( pThreadData->sListen );
                                Sleep( 3 );
                                if( ( pThreadData->sListen = socket( AF_INET, SOCK_ST
REAM, IPPROTO_TCP ) ) == INVALID_SOCKET ) {
                                        LOGNETERR( "socket:" );
                                        bRestartOK = TRUE;
                                }
                                pThreadData->saListen.sin_family = AF_INET;
                                pThreadData->saListen.sin_port = htons( pThreadData->
Cfg.uLocalPort );
                                if( pThreadData->Cfg.bRejectRemote )
                                        pThreadData->saListen.sin_addr.s_addr = htonl
( MAKEIPADDRESS( 127, 0, 0, 1 ) );
                                else
                                        pThreadData->saListen.sin_addr.s_addr = INADD
R_ANY;
                                if( bind( pThreadData->sListen, (LPSOCKADDR)&( pThrea
dData->saListen ), sizeof( pThreadData->saListen ) ) == SOCKET_ERROR ) {
                                        LOGNETERR( "bind:" );
                                        bRestartOK = FALSE;
                                } else if( listen( pThreadData->sListen, 5 ) == SOCKE
T_ERROR ) {
                                        LOGNETERR( "listen:" );
                                        bRestartOK = FALSE;
                                } else if( WSAAsyncSelect( pThreadData->sListen, pThr
eadData->hMainWnd, WM_NETEVENT, FD_ACCEPT ) == SOCKET_ERROR ) {
                                        LOGNETERR( "WSAAsyncSelect:" );
                                        bRestartOK = FALSE;
                                }
                                if( bRestartOK ) {
                                        LOADSTR2( IDS_EVENT_RESTART, Str );
                                        LOGMSG( LOG_EVENT, Str, NULL );
                                } else {
                                        pThreadData->sListen = NULL;
                                        LOADSTR2( IDS_INFO_ENDSVC, Str );
                                        LOGMSG( LOG_INFO, Str, NULL );
                                        // メニューを変更
                                        EnableMenuItem( pThreadData->hMainMenu, ID_ME
NUITEM_STARTSVC, MF_BYCOMMAND | MF_ENABLED );
                                        EnableMenuItem( pThreadData->hMainMenu, ID_ME
NUITEM_ENDSVC, MF_BYCOMMAND | MF_GRAYED );
                                        // ステータス行を変更
                                        LOADSTR( IDS_STATBAR_STOP, Str );
                                        SendMessage( pThreadData->hStatusWnd, SB_SETT
EXT, 0, (LPARAM)Str );
                                }
                        }

                        EndDialog( hWnd, IDOK );
                        break;

                case IDCANCEL:
                        EndDialog( hWnd, IDNO );
                        break;

                default:
                        break;
                }
                break;

        case WM_CLOSE:
                EndDialog( hWnd, IDNO );
                break;

        case WM_DESTROY:
```



```
                pThreadData->hDlgWnd = NULL;
                break;

        default:
                return FALSE;
        }
        return TRUE;
}


LRESULT CALLBACK VersionDlgProc( HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam )
 {
        static PTHREADDATA pThreadData;
        CHAR Str[256];

        switch ( uMsg ) {
        case WM_INITDIALOG:
                pThreadData = (PTHREADDATA)lParam;
//              pThreadData = &ThreadData;
                LOGFONT Logfont;

                // 文字列をセット
                SetDlgItemText( hWnd, IDC_EMAILADDR, EmailADDR );
                SetDlgItemText( hWnd, IDC_URI, HomePageURI );

                // メールアドレス，ホームページ URI をクリッカブルにする
                pThreadData->hLinkFont = (HFONT)SendMessage( hWnd, WM_GETFONT, 0, 0 )
;
                GetObject( pThreadData->hLinkFont, sizeof( LOGFONT ), &Logfont );
                Logfont.lfUnderline = 1;
                pThreadData->hLinkFont = CreateFontIndirect( &Logfont );
                SendDlgItemMessage( hWnd, IDC_EMAILADDR, WM_SETFONT, (WPARAM)pThreadD
ata->hLinkFont, 0 );
                SendDlgItemMessage( hWnd, IDC_URI, WM_SETFONT, (WPARAM)pThreadData->h
LinkFont, 0 );
                if( pThreadData->hCurHand == NULL )
                        pThreadData->hCurHand = LoadCursor( pThreadData->hRsrcInstanc
e, MAKEINTRESOURCE( IDC_HAND ) );
                pThreadData->AboutDlgEMailAddrLabelOrgProc = (WNDPROC)GetWindowLong(
GetDlgItem( hWnd, IDC_EMAILADDR ), GWL_WNDPROC );
                pThreadData->AboutDlgURILabelOrgProc = (WNDPROC)GetWindowLong( GetDlg
Item( hWnd, IDC_URI ), GWL_WNDPROC );
                LinkProc( hWnd, WM_INITLABEL, 0, (LPARAM)pThreadData );
                // ラベルをサブクラス化
                SetWindowLong( GetDlgItem( hWnd, IDC_EMAILADDR ), GWL_WNDPROC, (LONG)
LinkProc );
                SetWindowLong( GetDlgItem( hWnd, IDC_URI ), GWL_WNDPROC, (LONG)LinkPr
oc );

                break;

        case WM_CTLCOLORSTATIC:
                INT nId;
                HDC hdc;

                hdc = (HDC)wParam;
                nId = GetDlgCtrlID( (HWND)lParam );
                if( nId == IDC_EMAILADDR || nId == IDC_URI ) {
                        SetTextColor( hdc, RGB( 0, 0, 255 ) );
                        SetBkMode( hdc, TRANSPARENT );
                        return (INT)GetSysColorBrush( COLOR_3DFACE );
                }
                break;

        case WM_COMMAND:
                switch( LOWORD( wParam ) ) {
```

```
                case IDOK:
                        EndDialog( hWnd, IDOK );
                        break;

                case IDC_EMAILADDR:
                        if( HIWORD( wParam ) == STN_CLICKED ) {
                                lstrcpy( Str, "mailto:" EmailADDR "?subject=About%20M
ail%20Wrapper" );
                                ShellExecute( hWnd, NULL, Str, NULL, "", SW_SHOW );
                        }
                        break;

                case IDC_URI:
                        if( HIWORD( wParam ) == STN_CLICKED )
                                ShellExecute( hWnd, NULL, HomePageURI, NULL, "", SW_S
HOW );
                        break;

                default:
                        break;
                }
                break;

        case WM_CLOSE:
                EndDialog( hWnd, IDOK );
                break;

        case WM_DESTROY:
                DeleteObject( pThreadData->hLinkFont );
                break;

        default:
                return FALSE;
        }
        return TRUE;
}


LRESULT CALLBACK LinkProc( HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam ) {
        INT nId;
        WNDPROC WndProc;
        static PTHREADDATA pThreadData = NULL;

        switch( uMsg ) {
        case WM_INITLABEL:
                pThreadData = (PTHREADDATA)lParam;
                return 1;
                break;

        case WM_SETCURSOR:
                SetCursor( pThreadData->hCurHand );
                return 1;
                break;

        default:
                break;
        }
        nId = GetDlgCtrlID( hWnd );
        if( nId == IDC_EMAILADDR )
                WndProc = pThreadData->AboutDlgEMailAddrLabelOrgProc;
        else
                WndProc = pThreadData->AboutDlgURILabelOrgProc;

        return CallWindowProc( WndProc, hWnd, uMsg, wParam, lParam );
}
```



```
BOOL LoadRsrcString( HINSTANCE hInst, UINT uRsrcID, LPTSTR lpBuf, INT iBufSize ) {
        TCHAR Buf[512];

        if( ! LoadString( hInst, uRsrcID, lpBuf, iBufSize ) ) {
                if( ! LoadString( hInst, IDS_ERR_DEFAULT, Buf, iBufSize ) ) {
                        lstrcpy( Buf, DefaultErrMessage );
                }
                wsprintf( lpBuf, Buf, uRsrcID );
                return FALSE;
        }
        return TRUE;
}


BOOL LoadRsrcString2( HINSTANCE hInst, UINT uRsrcID, LPTSTR lpBuf, INT iBufSize ) {
        TCHAR Buf[512];

        if( ! LoadString( hInst, uRsrcID, lpBuf, iBufSize ) ) {
                if( ! LoadString( hInst, IDS_ERR_DEFAULT, Buf, iBufSize ) ) {
                        lstrcpy( Buf, DefaultErrMessage );
                }
                wsprintf( lpBuf, Buf, uRsrcID );
                lstrcat( lpBuf, "\n" );
                return FALSE;
        }
        lstrcat( lpBuf, "\n" );
        return TRUE;
}


void LogErr( PTHREADDATA pThreadData, LPCTSTR lpCaption ) {
        CHAR *lpMsgBuf;

        FormatMessage( FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
                NULL, GetLastError(),
            MAKELANGID( LANG_NEUTRAL, SUBLANG_DEFAULT ),
                (LPTSTR) &lpMsgBuf, 0, NULL );
//      MessageBox( hWnd, lpMsgBuf, Caption, MB_OK | MB_ICONERROR | MB_SETFOREGROUND
);
        lstrcat( lpMsgBuf, "\n" );
        LogMsg( LOG_CRITICAL, lpMsgBuf, lpCaption, pThreadData );
        LocalFree( lpMsgBuf );
}


void LogNetErr( PTHREADDATA pThreadData, LPCTSTR lpCaption ) {
//      CHAR *lpMsgBuf;
        TCHAR Str[256];
        INT nErr;

        nErr = WSAGetLastError();
        LOADSTR2( nErr, Str );

//      MessageBox( hWnd, lpMsgBuf, Caption, MB_OK | MB_ICONERROR | MB_SETFOREGROUND
);
        LOGMSG( LOG_CRITICAL, Str, lpCaption );
//      LocalFree( lpMsgBuf );
        return;
}


void LogMsg( LOGLEVEL LogLevel, LPCTSTR lpMsg, LPCTSTR lpCaption, PTHREADDATA pThread
Data ) {
        CHARFORMAT CharFormat;
        CHARRANGE CharRange;
```

```
        TCHAR Str[256];
        SYSTEMTIME SystemTime;

        CharRange.cpMin = -1;
        CharRange.cpMax =-1;
        SendMessage( pThreadData->hEditWnd, EM_EXSETSEL, 0, (LPARAM)&CharRange );
        CharFormat.cbSize = sizeof( CHARFORMAT );
        CharFormat.dwMask = CFM_COLOR | CFM_FACE | CFM_SIZE;
        CharFormat.dwEffects = 0;
        CharFormat.crTextColor = RGB( 0, 0, 0 );
        CharFormat.yHeight = 240;
//      CharFormat.bCharSet = SHIFTJIS_CHARSET;
//      CharFormat.bPitchAndFamily = FIXED_PITCH;
        lstrcpy( CharFormat.szFaceName, DEFAULTFONTNAME );
        SendMessage( pThreadData->hEditWnd, EM_SETCHARFORMAT, SCF_SELECTION, (LPARAM)
&CharFormat );
        GetLocalTime( &SystemTime );
        if( lpCaption != NULL )
                wsprintf( Str, "%02d:%02d:%02d %s ", SystemTime.wHour, SystemTime.wMi
nute, SystemTime.wSecond, lpCaption );
        else
                wsprintf( Str, "%02d:%02d:%02d ", SystemTime.wHour, SystemTime.wMinut
e, SystemTime.wSecond );
        SendMessage( pThreadData->hEditWnd, EM_REPLACESEL, FALSE, (LPARAM)Str );

        CharFormat.dwMask = CFM_COLOR;
        CharFormat.dwEffects = 0;
        CharFormat.crTextColor = pThreadData->crLogColor[LogLevel];
        SendMessage( pThreadData->hEditWnd, EM_SETCHARFORMAT, SCF_SELECTION, (LPARAM)
&CharFormat );
        SendMessage( pThreadData->hEditWnd, EM_REPLACESEL, FALSE, (LPARAM)lpMsg );

//      lstrcpy( Str, "\n" );
//      SendMessage( pThreadData->hEditWnd, EM_REPLACESEL, FALSE, (LPARAM)Str );
        return;
}


void FmtErrMsg( HWND hWnd, CHAR *Caption, LONG ErrNum ) {
        CHAR *lpMsgBuf;

        FormatMessage( FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
                NULL, ErrNum,
            MAKELANGID( LANG_NEUTRAL, SUBLANG_DEFAULT ),
                (LPTSTR) &lpMsgBuf, 0, NULL );
        MessageBox( hWnd, lpMsgBuf, Caption, MB_OK | MB_ICONERROR | MB_SETFOREGROUND
);
        LocalFree( lpMsgBuf );
}


void DispErrMsg( HWND hWnd, CHAR *Caption ) {
        CHAR *lpMsgBuf;

        FormatMessage( FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
                NULL, GetLastError(),
            MAKELANGID( LANG_NEUTRAL, SUBLANG_DEFAULT ),
                (LPTSTR) &lpMsgBuf, 0, NULL );
        MessageBox( hWnd, lpMsgBuf, Caption, MB_OK | MB_ICONERROR | MB_SETFOREGROUND
);
        LocalFree( lpMsgBuf );
}


// similar to PathAppend()
void MakeFullPath( char *fullpath, char *dirname, char *basename ) {
```



```
        lstrcpy( fullpath, dirname );
        if( dirname[lstrlen( dirname ) - 1] != '\\' )
                lstrcat( fullpath, "\\" );
        lstrcat( fullpath, basename );
}


LPTSTR MyStrRChr( LPCTSTR lpStart, LPCTSTR lpEnd, WORD wMatch ) {
        HINSTANCE hInstance;
        LPTSTR (*fpStrRChr)( LPCTSTR, LPCTSTR, WORD ) = NULL;
        LPTSTR lpReturn = NULL;

        if( ( hInstance = LoadLibraryEx( ShlWapiDLL, NULL, 0 ) ) != NULL )
                fpStrRChr = (LPTSTR (*)( LPCTSTR, LPCTSTR, WORD ))GetProcAddress( hIn
stance, STRRCHR );

        if( fpStrRChr == NULL && ( hInstance = LoadLibraryEx( Shell32DLL, NULL, 0 ) )
 != NULL )
                fpStrRChr = (LPTSTR (*)( LPCTSTR, LPCTSTR, WORD ))GetProcAddress( hIn
stance, STRRCHR );

        if( fpStrRChr ) {
                lpReturn = (*fpStrRChr)( lpStart, lpEnd, wMatch );
                FreeLibrary( hInstance );
        }
        return lpReturn;
}


__declspec(dllexport) LONG WINAPI HoehoeGetOSType( HWND hMainWnd, PHOEHOEOS pHoehoeOS
, LPSTR pString ) {
        OSVERSIONINFO ver;

        // Windows 95 か NT かの判定
        ZeroMemory( &ver, sizeof( ver ) );
//      memset( &ver, 0, sizeof( ver ) );
        ver.dwOSVersionInfoSize = sizeof( ver );
        if( ! GetVersionEx( &ver ) ) {
                DispErrMsg( hMainWnd, "GetVersionEx" );
                return -1;
        }
        switch( ver.dwPlatformId ) {
        case VER_PLATFORM_WIN32_WINDOWS:
                pHoehoeOS->OSSeries = Win9x;
                if( ver.dwMajorVersion > 4 || ver.dwMinorVersion > 3 ) {
                        pHoehoeOS->OSType = Win98;
                        if( pString )
                                lstrcpy( pString, (PSZ) "Windows 98" );
                } else {
                        pHoehoeOS->OSType = Win95;
                        if( pString )
                                lstrcpy( pString, (PSZ) "Windows 95" );
                }
                break;
        case VER_PLATFORM_WIN32_NT:
                pHoehoeOS->OSSeries = WinNT;
                if( pString )
                        lstrcpy( pString, (PSZ) "Windows NT" );
                if( ver.dwMajorVersion > 4 ) {
                        pHoehoeOS->OSType = WinNT5;
                        if( pString )
                                lstrcat( pString, (PSZ) " 5.0" );
                } else if( ver.dwMajorVersion == 4 ) {
                        pHoehoeOS->OSType = WinNT4;
                        if( pString )
                                lstrcat( pString, (PSZ) " 4.0" );
```

```
                } else {
                        if( ver.dwMinorVersion > 1) {
                                pHoehoeOS->OSType = WinNT35;
                                if( pString )
                                        lstrcat( pString, (PSZ) " 3.5" );
                        } else {
                                pHoehoeOS->OSType = WinNT31;
                                if( pString )
                                        lstrcat( pString, (PSZ) " 3.1" );
                        }
                }
                break;
        case VER_PLATFORM_WIN32_CE:
                pHoehoeOS->OSSeries = WinCE;
                if( pString )
                        lstrcpy( pString, (PSZ) "Windows CE" );
                if( ver.dwMajorVersion > 1 ) {
                        pHoehoeOS->OSType = WinCE2;
                        if( pString )
                                lstrcat( pString, (PSZ) " 2.0" );
                } else {
                        pHoehoeOS->OSType = WinCE1;
                        if( pString )
                                lstrcat( pString, (PSZ) " 1.0" );
                }
                break;
        case VER_PLATFORM_WIN32s:
                pHoehoeOS->OSSeries = Win32;
                pHoehoeOS->OSType = Win32s;
                if( pString )
                        lstrcpy( pString, (PSZ) "Windows 3.1" );
                break;
        default:
                pHoehoeOS->OSSeries = Win;
                pHoehoeOS->OSType = WinOTHER;
                if( pString )
                        lstrcpy( pString, (PSZ) "unknown OS" );
        }

        return 0;
}
```

```
# Microsoft Developer Studio Project File - Name="ftpwrap" - Package Owner=<4>
# Microsoft Developer Studio Generated Build File, Format Version 6.00
# ** 編集しないでください **

# TARGTYPE "Win32 (x86) Application" 0x0101

CFG=ftpwrap - Win32 Debug
!MESSAGE これは有効なメイクファイルではありません。 このプロジェクトをビルドするため
には NMAKE を使用してください。
!MESSAGE [メイクファイルのエクスポート] コマンドを使用して実行してください
!MESSAGE
!MESSAGE NMAKE /f "ftpwrap.mak".
!MESSAGE
!MESSAGE NMAKE の実行時に構成を指定できます
!MESSAGE コマンド ライン上でマクロの設定を定義します。例:
!MESSAGE
!MESSAGE NMAKE /f "ftpwrap.mak" CFG="ftpwrap - Win32 Debug"
!MESSAGE
!MESSAGE 選択可能なビルド モード:
!MESSAGE
!MESSAGE "ftpwrap - Win32 Release" ("Win32 (x86) Application" 用)
!MESSAGE "ftpwrap - Win32 Debug" ("Win32 (x86) Application" 用)
!MESSAGE

# Begin Project
# PROP AllowPerConfigDependencies 0
# PROP Scc_ProjName ""$/ftpwrap", HBAAAAAA"
# PROP Scc_LocalPath "."
CPP=cl.exe
MTL=midl.exe
RSC=rc.exe

!IF  "$(CFG)" == "ftpwrap - Win32 Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "Release"
# PROP BASE Intermediate_Dir "Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
# PROP Intermediate_Dir "Release"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_WINDOWS" /D "_MBCS" /Y
u"stdafx.h" /FD /c
# ADD CPP /nologo /MT /W3 /GX /O2 /I "..\openssl-0.9.6a\inc32" /I "..\openssl-0.9.6a\
inc32\openssl" /I "..\zlib" /D "NDEBUG" /D "WIN32" /D "_WINDOWS" /D "_MBCS" /D WIN32_
WINNT=0x0400 /FD /c
# ADD BASE MTL /nologo /D "NDEBUG" /mktyplib203 /win32
# ADD MTL /nologo /D "NDEBUG" /mktyplib203 /win32
# ADD BASE RSC /l 0x411 /d "NDEBUG"
# ADD RSC /l 0x411 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi3
2.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /su
bsystem:windows /machine:I386
# ADD LINK32 wsock32.lib kernel32.lib user32.lib gdi32.lib comdlg32.lib advapi32.lib
comctl32.lib shlwapi.lib shell32.lib winmm.lib ..\openssl-0.9.6a\out32dll\libeay32.li
b ..\zlib\Release\zlib.lib /nologo /subsystem:windows /machine:I386
# Begin Special Build Tool
SOURCE="$(InputPath)"
```



```
PostBuild_Cmds=type ftpwraprc.rc | ..\buildcount.exe > ftpwrap.rc
# End Special Build Tool

!ELSEIF  "$(CFG)" == "ftpwrap - Win32 Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /W3 /Gm /GX /ZI /Od /D "WIN32" /D "_DEBUG" /D "_WINDOWS" /D "_
MBCS" /Yu"stdafx.h" /FD /GZ /c
# ADD CPP /nologo /W3 /Gm /GX /ZI /Od /I "..\openssl-0.9.6a\inc32" /I "..\openssl-0.9
.6a\inc32\openssl" /I "..\zlib" /D "_DEBUG" /D WIN32_WINNT=0x0400 /D "WIN32" /D "_WIN
DOWS" /D "_MBCS" /FD /GZ /c
# ADD BASE MTL /nologo /D "_DEBUG" /mktyplib203 /win32
# ADD MTL /nologo /D "_DEBUG" /mktyplib203 /win32
# ADD BASE RSC /l 0x411 /d "_DEBUG"
# ADD RSC /l 0x411 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi3
2.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /su
bsystem:windows /debug /machine:I386 /pdbtype:sept
# ADD LINK32 ws2_32.lib kernel32.lib user32.lib gdi32.lib comdlg32.lib advapi32.lib c
omctl32.lib shlwapi.lib shell32.lib winmm.lib ..\openssl-0.9.6a\out32dll\libeay32.lib
 ..\zlib\Release\zlib.lib /nologo /subsystem:windows /debug /machine:I386 /pdbtype:se
pt
# Begin Special Build Tool
SOURCE="$(InputPath)"
PostBuild_Cmds=type ftpwraprc.rc | ..\buildcount.exe > ftpwrap.rc
# End Special Build Tool

!ENDIF

# Begin Target

# Name "ftpwrap - Win32 Release"
# Name "ftpwrap - Win32 Debug"
# Begin Group "Source Files"

# PROP Default_Filter "cpp;c;cxx;rc;def;r;odl;idl;hpj;bat"
# Begin Source File

SOURCE=.\auth.c
# End Source File
# Begin Source File

SOURCE=.\commlib.cpp
# End Source File
# Begin Source File

SOURCE=.\crypt.c
# End Source File
# Begin Source File

SOURCE=.\ftp_cmds.cpp
# End Source File
```

```
# Begin Source File

SOURCE=.\ftp_get_cmd.cpp
# End Source File
# Begin Source File

SOURCE=.\ftp_msg.cpp
# End Source File
# Begin Source File

SOURCE=.\ftpwrap.cpp
# End Source File
# Begin Source File

SOURCE=.\ftpwrap.rc
# End Source File
# Begin Source File

SOURCE=.\fwd.c
# End Source File
# Begin Source File

SOURCE=.\fwdui.c
# End Source File
# Begin Source File

SOURCE=.\hosts.c
# End Source File
# Begin Source File

SOURCE=.\keyfiles.c
# End Source File
# Begin Source File

SOURCE=.\matcher.c
# End Source File
# Begin Source File

SOURCE=.\netevent.cpp
# End Source File
# Begin Source File

SOURCE=.\pkt.c
# End Source File
# Begin Source File

SOURCE=.\reqlist.cpp
# End Source File
# Begin Source File

SOURCE=.\scm.cpp
# End Source File
# Begin Source File

SOURCE=.\ssh.c
# End Source File
# Begin Source File

SOURCE=.\StdAfx.cpp
# ADD CPP /Yc"stdafx.h"
# End Source File
# Begin Source File

SOURCE=.\ttxssh.c
# End Source File
# Begin Source File
```



```
SOURCE=.\util.c
# End Source File
# End Group
# Begin Group "Header Files"

# PROP Default_Filter "h;hpp;hxx;hm;inl"
# Begin Source File

SOURCE=.\auth.h
# End Source File
# Begin Source File

SOURCE=.\crypt.h
# End Source File
# Begin Source File

SOURCE=.\ftpwrap.h
# End Source File
# Begin Source File

SOURCE=.\fwd.h
# End Source File
# Begin Source File

SOURCE=.\fwdui.h
# End Source File
# Begin Source File

SOURCE=.\hosts.h
# End Source File
# Begin Source File

SOURCE=.\keyfiles.h
# End Source File
# Begin Source File

SOURCE=.\matcher.h
# End Source File
# Begin Source File

SOURCE=.\pkt.h
# End Source File
# Begin Source File

SOURCE=.\resource.h
# End Source File
# Begin Source File

SOURCE=.\ssh.h
# End Source File
# Begin Source File

SOURCE=.\StdAfx.h
# End Source File
# Begin Source File

SOURCE=.\ttxssh.h
# End Source File
# Begin Source File

SOURCE=.\util.h
# End Source File
# End Group
# Begin Group "Resource Files"
```

```
# PROP Default_Filter "ico;cur;bmp;dlg;rc2;rct;bin;rgs;gif;jpg;jpeg;jpe"
# Begin Source File

SOURCE=.\H_POINT.CUR
# End Source File
# Begin Source File

SOURCE=.\icon1.ico
# End Source File
# Begin Source File

SOURCE=.\icon2.ico
# End Source File
# Begin Source File

SOURCE=.\icon3.ico
# End Source File
# Begin Source File

SOURCE=.\ttsecure.ico
# End Source File
# End Group
# End Target
# End Project
```

```
Microsoft Developer Studio Workspace File, Format Version 6.00
# 警告: このワークスペース ファイル を編集または削除しないでください!

###############################################################################

Project: "ftpwrap"=.\ftpwrap.dsp - Package Owner=<4>

Package=<5>
{{{
}}}

Package=<4>
{{{
}}}

###############################################################################

Project: "ttermpro"=..\ttsrcp23\visualc\ttermpro.dsp - Package Owner=<4>

Package=<5>
{{{
}}}

Package=<4>
{{{
}}}

###############################################################################

Global:

Package=<5>
{{{
}}}

Package=<3>
{{{
}}}

###############################################################################
```

```
// ftpwrap.h : ftpwrap アプリケーションのメイン ヘッダー ファイルです。
//

#if !defined(_FTPWRAP_H__5C9B5A85_3F24_11D2_8EA3_0040051B0425__INCLUDED_)
#define _FTPWRAP_H__5C9B5A85_3F24_11D2_8EA3_0040051B0425__INCLUDED_

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

#include "resource.h"           // メイン シンボル
#include <winsock2.h>
#include <windowsx.h>
#include "ttxssh.h"


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

#ifndef VER_PLATFORM_WIN32_CE
#  define VER_PLATFORM_WIN32_CE 0x03                    // WINDOWS CE
#endif


//
// コードの選択
//


//
// 構造体・定数等
//

// OS とそのバージョン
typedef enum {
        Win32s,
        WinNT31,
        WinNT35,
        WinNT4,
        WinNT5,
        Win95,
        Win98,
        WinCE1,
        WinCE2,
        WinOTHER
} HOEHOEOSTYPE;

// OS の系列
typedef enum {
        Win32,
        Win9x,
        WinNT,
        WinCE,
        Win
} HOEHOEOSSERIES;

// HoehoeGetOSType()用の構造体
typedef struct _HOEHOEOS {
        HOEHOEOSTYPE OSType;
        HOEHOEOSSERIES OSSeries;
} HOEHOEOS, *PHOEHOEOS;

typedef enum {
        LOG_INFO,
        LOG_EVENT,
        LOG_NORMAL,
        LOG_CRITICAL
```



```
} LOGLEVEL;

// 設定を保持する構造体
typedef struct _CONFIGDATA {
        USHORT uRemotePort;
        USHORT uLocalPort;
        BOOL bRemoteAddrType;
        UINT uRemoteIP;
        CHAR cRemoteName[256];
        BOOL bRejectRemote;
        BOOL bKerberos;
        BOOL bStartService;
        BOOL bAutoRun;
        UINT uAuthMethod;
} CONFIGDATA, *PCONFIGDATA;


// スレッド間のデータ受け渡し用構造体
typedef struct _THREADDATA {
        HWND hMainWnd;
        HWND hEditWnd;
        HWND hStatusWnd;
        DWORD dwThreadId;
        HANDLE hThread;
        INT iReturnValue;
        HINSTANCE hInstance;
        HMENU hPopupMenu;
        CONFIGDATA Cfg;
        HKEY hRegKey;
        RECT RectMain;
        BOOL bSysTrayIcon;
        HINSTANCE hRsrcInstance;
        HMENU hMainMenu;
        SOCKET sListen;
        SOCKADDR_IN saListen;
        COLORREF crLogColor[4];
        LPTSTR lpCmdLine;
        BOOL bFirstExecute;
        HFONT hLinkFont;
        HCURSOR hCurHand;
        WNDPROC AboutDlgEMailAddrLabelOrgProc;
        WNDPROC AboutDlgURILabelOrgProc;
        HWND hDlgWnd;
} THREADDATA, *PTHREADDATA;

// クライアントと通信を行う子スレッド用構造体
typedef struct _SUBTHREADDATA {
        SOCKET sClient;
        SOCKADDR_IN saClient;
        SOCKET sServer;
        SOCKADDR_IN saServer;
        CONFIGDATA Cfg;
        PTHREADDATA pThreadData;
} SUBTHREADDATA, *PSUBTHREADDATA;


//
// 関数のプロトタイプ宣言
//

// ローカル関数
VOID FmtErrMsg( HWND hWnd, CHAR *Caption, LONG ErrNum );
VOID DispErrMsg( HWND hWnd, CHAR *Caption );
VOID MakeFullPath( PCHAR fullpath, PCHAR dirname, PCHAR basename );
HMENU CreateContextMenu( PTHREADDATA pThreadData );
VOID ShowMainWindow( PTHREADDATA pThreadData );
```

```
LRESULT CALLBACK WndProc( HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam );
LRESULT CALLBACK SetupDlgProc( HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam );
LRESULT CALLBACK VersionDlgProc( HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam )
;
LRESULT CALLBACK LinkProc( HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam );
BOOL LoadRsrcString( HINSTANCE hInst, UINT uRsrcID, LPTSTR lpBuf, INT iBufSize );
BOOL LoadRsrcString2( HINSTANCE hInst, UINT uRsrcID, LPTSTR lpBuf, INT iBufSize );
VOID LogMsg( LOGLEVEL LogColor, LPCTSTR lpMsg, LPCTSTR lpCaption, PTHREADDATA pThread
Data );
VOID LogErr( PTHREADDATA pThreadData, LPCTSTR lpCaption );
VOID LogNetErr( PTHREADDATA pThreadData, LPCTSTR lpCaption );
DWORD WINAPI ServiceThread( PSUBTHREADDATA pSubThreadData );
__declspec(dllexport) LONG WINAPI HoehoeGetOSType( HWND hMainWnd, PHOEHOEOS pHoehoeOS
, LPSTR pString );


//
// マクロ
//

//#define CLOSERETURN(RETVAL)   RegCloseKey( hRegKey ); pThreadData->iReturnValue = R
ETVAL; SendMessage( pThreadData->hwndMain, WM_CLOSEWND, 0, 0 ); _endthreadex( RETVAL
); return RETVAL
//#define RETURN(RETVAL)               pThreadData->iReturnValue = RETVAL; SendMessa
ge( pThreadData->hwndMain, WM_CLOSEWND, 0, 0 ); _endthreadex( RETVAL ); return RETVAL
#define LOADSTR(ID,BUF)              LoadRsrcString( pThreadData->hRsrcInstance, I
D, BUF, sizeof( BUF ) )
#define LOADSTR2(ID,BUF)             LoadRsrcString2( pThreadData->hRsrcInstance,
ID, BUF, sizeof( BUF ) )
#define LOGNETERR(CAPTION)           LogNetErr( pThreadData, CAPTION )
#define LOGERR(CAPTION)              LogErr( pThreadData, CAPTION )
#define LOGMSG(LEV,MSG,CAPTION) LogMsg( LEV, MSG, CAPTION, pThreadData )
#define CLOSERETURN(RETVAL)          RegCloseKey( hRegKey ); pThreadData->iReturnV
alue = RETVAL; SendMessage( pThreadData->hMainWnd, WM_CLOSEWND, 0, 0 ); ReleaseMutex(
 hMutex ); CloseHandle( hMutex ); ExitThread( RETVAL ); return RETVAL
#define SVCRETURN(RETVAL)            GlobalFree( pSubThreadData ); ExitThread( RET
VAL ); return RETVAL



//
// レジストリ
//

// Run/RunOnce
#define RunREGKEY "Software\\Microsoft\\Windows\\CurrentVersion\\Run"
#define RunFTPWRAPREG "Mail wrapper"
// Mail wrapper
#define FTPWRAPREGKEY "Software\\Knights of the Round Table\\Mail wrapper"
#define FTPWRAPRemotePortREG "RemotePort"
#define FTPWRAPLocalPortREG "LocalPort"
#define FTPWRAPRejectRemoteREG "RejectRemote"
#define FTPWRAPRemoteAddrTypeREG "RemoteHostAddrType"
#define FTPWRAPRemoteAddrREG "RemoteAddr"
#define FTPWRAPRemoteNameREG "RemoteName"
#define FTPWRAPKerberosREG "Kerochan"
#define FTPWRAPStartServiceREG "StartService"
#define FTPWRAPMailProtocolREG "MailProtocol"
#define FTPWRAPAuthMethodREG "AuthMethod"
#define FTPWRAPWindowXREG "WindowX"
#define FTPWRAPWindowYREG "WindowY"
#define FTPWRAPWindowWidthREG "WindowWidth"
#define FTPWRAPWindowHeightREG "WindowHeight"


//
```



```
// パス
//

// ローダ・DLL ファイル名
#define RichEditDLL "riched32.dll"
#define ShlWapiDLL "shlwapi.dll"
#define Shell32DLL "shell32.dll"
#define FTPWRAPPATH "ftpwrap.exe"
#define STRRCHR "StrRChr"


//
// その他
//

#define DefaultErrMessage "The messsage for this event doesn't exist. (code: %02x)"

#define POP3_LOCALLABEL "pop3 (&110)"
#define POP3_REMOTELABEL "pop&3 (110)"
#define IMAP_LOCALLABEL "imap (&143)"
#define IMAP_REMOTELABEL "i&map (143)"

#define EmailADDR "lancelot@sakura.sfc.wide.ad.jp"
#define HomePageURI "http://sakura.sfc.wide.ad.jp/ftpwrapper/"
// 複数起動抑制用の ID
#define AppID "lancelot@sakura.sfc.wide.ad.jp/FTP wrapper"
#define SyncID1 "lancelot@sakura.sfc.wide.ad.jp/FTP wrapper/Sync1"
// ポート番号
#define IMAPPORT_IMAP 143
// デフォルトの認証方式
#define IMAPAUTH_DEFAULT 0;

#define DEFAULTFONTNAME "ＭＳ ゴシック"


// dest buf, src buf, dest bufsize, src copy length(0=all)
#define MyStrNCpy(A,B,C,D)                                          \
{                                                                   \
    strncpy( A, B,                                                  \
           ( ( C - 1 ) < D || D == 0 ) ? C - 1 : D );               \
    A[( ( C - 1 ) < D || D == 0 ) ? C - 1 : D] = '\0';              \
}
#define MyStrNCat(A,B,C,D)                                          \
{                                                                   \
  size_t prev_len = strlen( A );                                    \
  strncpy( A + prev_len , B,                                        \
           ( ( (C - prev_len - 1) < D ) || D == 0 ) ?               \
            (C - prev_len - 1) : D );                               \
           A[( (C - 1) < (prev_len + D) || D == 0 ) ?               \
            (C - 1) : (prev_len + D)] = '\0';                       \
}
#define MyStrNCpy2(A,B,C)  MyStrNCpy(A,B,sizeof(A),C)
#define MyStrNCat2(A,B,C)  MyStrNCat(A,B,sizeof(A),C)


extern HWND g_hWnd;

typedef enum {
        ST_GREETING,    // greeting message 送信中
        ST_WAIT_CMD,    // FTP cmd 受信待機
        ST_SEND_RESP,   // FTP cmd に対する応答送信中，送信後受信待機に移行
        ST_CONNECTING,  // USERコマンド実行中
        ST_CONNECTING2,
        ST_CONNECTING3,
        ST_CONNECTING4,
        ST_CONNECT,             // FTP cmd に対する応答送信中またはサーバ転送モード
```

```
        ST_QUIT                 // 接続終了
} STATE_FTP;

#define VERSION "0.01"
#define MAXPARMCOUNT 4  // コマンド自身を含む

#define FTPRDBUFFSIZE 1024 * 16
#define FTPWRBUFFSIZE 128 * 1024
#define SSHRDBUFFSIZE 128 * 1024
#define FTPWRITESIZE 1024
typedef struct _REQUEST {
        SOCKET          socket;
        SOCKADDR_IN             SockAddr;

        unsigned char *WrBuff;
        unsigned long WrDataSize;
        unsigned char *RdBuff;
        unsigned long RdDataSize;

        STATE_FTP               state;
        char     ftp_cmdline_buf[2048];
        char   *ftp_parm[MAXPARMCOUNT];  /*  Parse command parameter list */
        int              parm_count;               /*  Number of parameters in parse
d list */

        int             ftp_resp_code;
        char    ftp_resp_str[2048];

        TCHAR HostName[256];
        TCHAR UserName[256];
        unsigned short TCPPort;

        PTInstVar pvar;
        int localch;
        int lastremoteport;
        unsigned char *SSHRdBuff;
        unsigned long SSHRdDataSize;

        struct          _REQUEST *lpNext;
} REQUEST, *PREQUEST;
#define ftp_command         ftp_parm[0]   /*  FTP command is the first token */

void cmd_user( PREQUEST lpReq );
void cmd_user2( PREQUEST lpReq, int result );
void cmd_user3( PREQUEST lpReq, int result );
void cmd_noop( PREQUEST lpReq );
void cmd_help( PREQUEST lpReq );
void cmd_quit( PREQUEST lpReq );
int cmd_port( PREQUEST lpReq );

void OnAccept(HWND hwnd, SOCKET socket, int nErrorCode);
void OnRead(SOCKET socket, int nErrorCode);
void ParseRecvData(PREQUEST lpReq);
void OnWrite(SOCKET socket, int nErrorCode);
int WriteData(PREQUEST lpReq);
void OnClose(SOCKET socket, int nErrorCode);

VOID ftp_msgf( PREQUEST, int , const char *,... );
int ftp_get_command( PREQUEST lpReq );
int ftp_get_resp( PREQUEST lpReq );


PREQUEST        FTPGetNewRequest(VOID);
PREQUEST        FTPGetRequest(SOCKET socket);
PREQUEST FTPGetRequestByPVar(PTInstVar pvar);
PREQUEST        FTPGetFirstRequest(void);
```



```
PREQUEST        FTPGetNextRequest(PREQUEST lpThis);
void            FTPDelRequest(PREQUEST lpThis);
void            FTPDelAllRequests(void);


PTInstVar       GetNewRequest(VOID);
PTInstVar       GetRequest(SOCKET socket);
PTInstVar       GetFirstRequest(void);
PTInstVar       GetNextRequest(PTInstVar lpThis);
void            DelRequest(PTInstVar lpThis);
void            DelAllRequests(void);

DWORD SCM_connect(HWND hwnd, PREQUEST lpReq);

//
// ユーザ定義メッセージ
//

#define WM_STATUSBAR                    (WM_USER + 1001)
#define WM_SYSTRAYICON                  (WM_USER + 1002)
#define WM_INITLABEL                    (WM_USER + 1003)
#define WM_NETEVENT                             (WM_USER + 1004)
#define ID_SYSTRAYICON1                 0


#endif // !defined(_FTPWRAP_H__5C9B5A85_3F24_11D2_8EA3_0040051B0425__INCLUDED_)
```

```
//Microsoft Developer Studio generated resource script.
//
#include "resource.h"
#include <winsock2.h>

#define APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

////////////////////////////////////////////////////////////////////////////
// 日本語 resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_JPN)
#ifdef _WIN32
LANGUAGE LANG_JAPANESE, SUBLANG_DEFAULT
#pragma code_page(932)
#endif //_WIN32

#ifndef _MAC
////////////////////////////////////////////////////////////////////////////
//
// Version
//

VS_VERSION_INFO VERSIONINFO
 FILEVERSION 0,0,1,364
 PRODUCTVERSION 0,0,1,364
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x40004L
 FILETYPE 0x1L
 FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "041104b0"
        BEGIN
            VALUE "Comments", "\0"
            VALUE "CompanyName", "Knights of the Round Table\0"
            VALUE "FileDescription", "FTP wrapper\0"
            VALUE "FileVersion", "0.01 (build 364)\0"
            VALUE "InternalName", "ftpwrap\0"
            VALUE "LegalCopyright", "Copyright (C) 1998-2001 Lancelot\0"
            VALUE "LegalTrademarks", "\0"
            VALUE "OriginalFilename", "ftpwrap.exe\0"
            VALUE "PrivateBuild", "\0"
            VALUE "ProductName", "FTP wrapper\0"
            VALUE "ProductVersion", "0.01 (build 364)\0"
            VALUE "SpecialBuild", "\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x411, 1200
    END
END
```



```
#endif    // !_MAC


#ifdef APSTUDIO_INVOKED
////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
    "\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED


////////////////////////////////////////////////////////////////////////////
//
// Menu
//

IDR_MENU1 MENU DISCARDABLE
BEGIN
    POPUP "制御(&O)"
    BEGIN
        MENUITEM "サービスの開始(&R)",          ID_MENUITEM_STARTSVC
        MENUITEM "サービスの停止(&S)",          ID_MENUITEM_ENDSVC, GRAYED
        MENUITEM SEPARATOR
        MENUITEM "終了(&X)\tAlt+F4",            ID_MENUITEM_EXIT
    END
    POPUP "設定(&S)"
    BEGIN
        MENUITEM "Windowsの起動と同時に実行(&A)", ID_MENUITEM_AUTORUN
        MENUITEM "実行と同時にサービスを開始(&R)", ID_MENUITEM_AUTOSTARTSVC
        MENUITEM SEPARATOR
        MENUITEM "設定(&S)",                    ID_MENUITEM_SETUP
    END
    POPUP "ヘルプ(&H)"
    BEGIN
        MENUITEM "目次(&C)",                    ID_MENUITEM_HELP, GRAYED
        MENUITEM SEPARATOR
        MENUITEM "バージョン情報(&A)",          ID_MENUITEM_VERSION
    END
END


////////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_ABOUTDIALOG DIALOG DISCARDABLE  0, 0, 263, 249
```

```
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "TTSSHについて"
FONT 10, "ＭＳ Ｐゴシック"
BEGIN
    ICON            IDI_SECURETT,IDC_STATIC,235,7,20,20
    LTEXT           "TTSSH\r\nTeraterm Secure Shell extension, version 1.5.4\r\nJapan
ese version 1.1 beta 6 (build 364)",
                    IDC_STATIC,6,7,219,25
    LTEXT           "Compatible with SSH protocol version 1.5\r\nCopyright Robert O'C
allahan (roc+tt@cs.cmu.edu), 1998-2001\r\nJapanization by Lancelot (lancelot@sakurach
an.org), 1998-2001",
                    IDC_STATIC,6,31,219,24
    LTEXT           "TTSSH home page:  http://www.zip.com.au/~roca/ttssh.html\r\nJapa
nese version:  http://www.sakurachan.org/teraterm-j/ttssh/",
                    IDC_STATIC,6,62,249,17
    LTEXT           "THIS EXTENSION INCORPORATES STRONG CRYPTOGRAPHY. POSESSION, USE,
 AND/OR EXPORT OF THIS EXTENSION MAY BE ILLEGAL IN YOUR COUNTRY.",
                    IDC_STATIC,6,86,249,24
    LTEXT           "This product includes software developed by the OpenSSL Project
for use in the OpenSSL Toolkit (http://www.openssl.org/).",
                    IDC_STATIC,6,116,249,17
    LTEXT           "Thanks also to Ian Goldberg for his Pilot SSH code, which was a
very handy reference. Special thanks to Tanaka Teranishi for his cooperation.",
                    IDC_STATIC,6,132,249,18
    EDITTEXT        IDC_ABOUTTEXT,6,155,249,65,ES_MULTILINE | ES_AUTOVSCROLL |
                    ES_AUTOHSCROLL | ES_READONLY
    DEFPUSHBUTTON   "OK",IDOK,106,228,50,14
END

IDD_SSHAUTH DIALOGEX 0, 0, 309, 214
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "SSH認証"
FONT 10, "ＭＳ Ｐゴシック"
BEGIN
    LTEXT           "pink-floyd.compose.cs.cmu.eduにログインしています.",
                    IDC_SSHAUTHBANNER,6,7,166,10
    LTEXT           "認証が必要です.",IDC_SSHAUTHBANNER2,6,16,296,10
    RTEXT           "ユーザ名(&N):",IDC_SSHUSERNAMELABEL,14,31,52,8,0,
                    WS_EX_RIGHT
    EDITTEXT        IDC_SSHUSERNAME,71,29,145,12,ES_AUTOHSCROLL | WS_GROUP
    RTEXT           "パスフレーズ(&P):",IDC_SSHPASSWORDCAPTION,14,48,52,8,0,
                    WS_EX_RIGHT
    EDITTEXT        IDC_SSHPASSWORD,71,46,145,12,ES_PASSWORD |
                    ES_AUTOHSCROLL | WS_GROUP
    CONTROL         "プレインテキストを使う(&L)",IDC_SSHUSEPASSWORD,"Button",
                    BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP,13,71,149,10
    CONTROL         "&RSA鍵を使う",IDC_SSHUSERSA,"Button",BS_AUTORADIOBUTTON,
                    13,99,84,10
    CONTROL         "r&hostsを使う",IDC_SSHUSERHOSTS,"Button",
                    BS_AUTORADIOBUTTON,13,128,59,10
    CONTROL         "&TISを使う",IDC_SSHUSETIS,"Button",BS_AUTORADIOBUTTON,
                    14,172,191,10
    PUSHBUTTON      "個人鍵(&K):",IDC_CHOOSERSAFILE,107,97,55,14,WS_GROUP
    EDITTEXT        IDC_RSAFILENAME,167,98,127,13,ES_AUTOHSCROLL | WS_GROUP
    LTEXT           "ローカルのユーザ名(&U):",IDC_LOCALUSERNAMELABEL,84,128,
                    76,8,0,WS_EX_RIGHT
    EDITTEXT        IDC_LOCALUSERNAME,167,126,127,13,ES_AUTOHSCROLL |
                    WS_GROUP
    PUSHBUTTON      "ホスト鍵(&F):",IDC_CHOOSEHOSTRSAFILE,90,141,70,14,
                    WS_GROUP
    EDITTEXT        IDC_HOSTRSAFILENAME,167,142,127,13,ES_AUTOHSCROLL |
                    WS_GROUP
    DEFPUSHBUTTON   "OK",IDOK,96,193,50,14,WS_GROUP
    PUSHBUTTON      "切断(&D)",IDCANCEL,162,193,50,14
    GROUPBOX        "",IDC_STATIC,6,62,296,25
    GROUPBOX        "",IDC_STATIC,6,90,296,25
```



```
    GROUPBOX        "",IDC_STATIC,6,119,296,42
    GROUPBOX        "",IDC_STATIC,6,163,296,26
END

IDD_HOSTDLG DIALOG MOVEABLE IMPURE  20, 20, 243, 115
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Tera Term: 新規接続"
FONT 11, "ＭＳ Ｐゴシック"
BEGIN
    CONTROL         "ホスト(&T):",IDC_HOSTNAMELABEL,"Static",
                    SS_LEFTNOWORDWRAP,49,10,31,8
    COMBOBOX        IDC_HOSTNAME,82,8,149,54,CBS_DROPDOWN | CBS_AUTOHSCROLL |
                    WS_VSCROLL | WS_GROUP | WS_TABSTOP
    CONTROL         "サービス:",IDC_SERVICELABEL,"Static",SS_LEFTNOWORDWRAP,
                    47,25,30,8
    CONTROL         "Te&lnet",IDC_HOSTTELNET,"Button",BS_AUTORADIOBUTTON |
                    WS_GROUP | WS_TABSTOP,80,24,44,12
    CONTROL         "&SSH",IDC_HOSTSSH,"Button",BS_AUTORADIOBUTTON,80,35,44,
                    12
    CONTROL         "その他(&O)",IDC_HOSTOTHER,"Button",BS_AUTORADIOBUTTON,
                    80,48,47,10
    CONTROL         "TCPポート#(&P):",IDC_HOSTTCPPORTLABEL,"Static",
                    SS_LEFTNOWORDWRAP,144,26,48,8
    EDITTEXT        IDC_HOSTTCPPORT,195,24,36,12,WS_GROUP
    CONTROL         "ポート(&R):",IDC_HOSTCOMLABEL,"Static",
                    SS_LEFTNOWORDWRAP,66,74,30,8
    COMBOBOX        IDC_HOSTCOM,99,72,42,46,CBS_DROPDOWNLIST |
                    CBS_AUTOHSCROLL | WS_VSCROLL | WS_GROUP | WS_TABSTOP
    DEFPUSHBUTTON   "OK",IDOK,47,95,41,14,WS_GROUP
    PUSHBUTTON      "キャンセル",IDCANCEL,101,95,41,14
    PUSHBUTTON      "ヘルプ(&H)",IDC_HOSTHELP,155,95,41,14
    CONTROL         "TCP/&IP",IDC_HOSTTCPIP,"Button",BS_AUTORADIOBUTTON |
                    WS_GROUP | WS_TABSTOP,8,9,41,10
    CONTROL         "シリアル(&E)",IDC_HOSTSERIAL,"Button",
                    BS_AUTORADIOBUTTON,8,73,51,10
    GROUPBOX        "",-1,4,0,234,60
    GROUPBOX        "",-1,4,64,234,26
END

IDD_SSHSETUP DIALOG DISCARDABLE  0, 0, 223, 233
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "TTSSH: 設定"
FONT 10, "ＭＳ Ｐゴシック"
BEGIN
    RTEXT           "圧縮レベル(&C):",IDC_STATIC,11,18,65,10
    LTEXT           "なし",IDC_STATIC,85,4,18,8
    LTEXT           "最高",IDC_STATIC,175,4,25,8
    CONTROL         "Slider1",IDC_SSHCOMPRESSIONLEVEL,"msctls_trackbar32",
                    TBS_AUTOTICKS | WS_GROUP | WS_TABSTOP,85,12,107,20
    GROUPBOX        "優先する暗号アルゴリズムの順(&P)",IDC_STATIC,17,33,187,
                    98
    LISTBOX         IDC_SSHCIPHERPREFS,31,45,159,60,LBS_NOINTEGRALHEIGHT |
                    WS_VSCROLL | WS_GROUP | WS_TABSTOP
    PUSHBUTTON      "上へ(&U)",IDC_SSHMOVECIPHERUP,43,110,58,13,WS_GROUP
    PUSHBUTTON      "下へ(&D)",IDC_SSHMOVECIPHERDOWN,117,110,58,13
    PUSHBUTTON      "読み書き(&W):",IDC_CHOOSEREADWRITEFILE,11,149,60,14,
                    WS_GROUP
    EDITTEXT        IDC_READWRITEFILENAME,77,150,129,13,ES_AUTOHSCROLL |
                    WS_GROUP
    PUSHBUTTON      "読み取り専用(&R):",IDC_CHOOSEREADONLYFILE,11,167,60,14,
                    WS_GROUP
    EDITTEXT        IDC_READONLYFILENAME,77,168,129,13,ES_AUTOHSCROLL |
                    WS_GROUP
    LTEXT           "いずれの変更も次回のセッション以降有効になります.",
                    IDC_STATIC,19,197,186,16
    GROUPBOX        "SSH Known Hosts",IDC_STATIC,7,138,209,52
```

```
    DEFPUSHBUTTON   "OK",IDOK,49,213,50,14,WS_GROUP
    PUSHBUTTON      "キャンセル",IDCANCEL,119,213,50,14
END

IDD_SSHUNKNOWNHOST DIALOG DISCARDABLE  0, 0, 215, 116
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "セキュリティ警告"
FONT 10, "ＭＳ Ｐゴシック"
BEGIN
    CONTROL         "このホストをknown hostsリストに追加する(&A)",
                    IDC_ADDTOKNOWNHOSTS,"Button",BS_AUTOCHECKBOX | WS_GROUP |
                    WS_TABSTOP,18,74,181,13
    LTEXT           "known hostsリストにサーバ ""###################################
#"" のエントリはありません．\r\n悪意を持ったホストが、接続しようとしているサーバのふ
りをしている可能性もありますので、充分注意して下さい！",
                    IDC_HOSTWARNING,15,7,184,41
    LTEXT           "known hostsリストにこのホストを追加して続行すると、次回からこの
警告は出なくなります．",
                    IDC_STATIC,15,52,184,20
    DEFPUSHBUTTON   "続行(&C)",IDC_CONTINUE,49,95,50,14,WS_GROUP
    PUSHBUTTON      "切断(&D)",IDCANCEL,115,95,50,14
END

IDD_SSHDIFFERENTHOST DIALOG DISCARDABLE  0, 0, 215, 114
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "セキュリティ警告"
FONT 10, "ＭＳ Ｐゴシック"
BEGIN
    CONTROL         "このホストをknown hostsリストに追加する(&A)",
                    IDC_ADDTOKNOWNHOSTS,"Button",BS_AUTOCHECKBOX | WS_GROUP |
                    WS_TABSTOP,34,74,154,10
    LTEXT           "known hostsリスト中のサーバ ""###################################
###"" のエントリと、接続先のホストの鍵が *一致していません*\r\n悪意を持ったホストが、
接続しようとしているサーバのふりをしている可能性があります！",
                    IDC_HOSTWARNING,15,7,184,43
    LTEXT           "known hostsリストにこのホストの新たな鍵を追加して続行すると、次
回からこの警告は出なくなります．",
                    -1,15,52,184,20
    DEFPUSHBUTTON   "切断(&D)",IDCANCEL,115,93,50,14,WS_GROUP
    PUSHBUTTON      "続行(&C)",IDC_CONTINUE,49,93,50,14
END

IDD_SSHAUTHSETUP DIALOGEX 0, 0, 309, 193
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "TTSSH: 認証設定"
FONT 10, "ＭＳ Ｐゴシック", 0, 0, 0x1
BEGIN
    LTEXT           "通常使用する認証方式:",IDC_SSHAUTHBANNER,6,7,166,10
    LTEXT           "ユーザ名(&N):",IDC_SSHUSERNAMELABEL,14,24,40,8,0,
                    WS_EX_RIGHT
    EDITTEXT        IDC_SSHUSERNAME,58,22,145,12,ES_AUTOHSCROLL | WS_GROUP
    CONTROL         "ブレインテキストを使う(&L)",IDC_SSHUSEPASSWORD,"Button",
                    BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP,13,46,149,10
    CONTROL         "&RSA鍵を使う",IDC_SSHUSERSA,"Button",BS_AUTORADIOBUTTON,
                    13,76,84,10
    CONTROL         "r&hostsを使う",IDC_SSHUSERHOSTS,"Button",
                    BS_AUTORADIOBUTTON,13,105,56,10
    CONTROL         "&TISを使う",IDC_SSHUSETIS,"Button",BS_AUTORADIOBUTTON,
                    14,150,191,10
    PUSHBUTTON      "個人鍵(&K):",IDC_CHOOSERSAFILE,107,73,55,14,WS_GROUP
    EDITTEXT        IDC_RSAFILENAME,167,73,127,13,ES_AUTOHSCROLL | WS_GROUP
    LTEXT           "ローカルのユーザ名(&U):",IDC_LOCALUSERNAMELABEL,80,105,
                    83,8,0,WS_EX_RIGHT
    EDITTEXT        IDC_LOCALUSERNAME,167,103,127,13,ES_AUTOHSCROLL |
                    WS_GROUP
    PUSHBUTTON      "ホスト鍵(&F):",IDC_CHOOSEHOSTRSAFILE,90,116,70,14,
```



```
                    WS_GROUP
    EDITTEXT        IDC_HOSTRSAFILENAME,167,119,127,13,ES_AUTOHSCROLL |
                    WS_GROUP
    DEFPUSHBUTTON   "OK",IDOK,96,172,50,14,WS_GROUP
    PUSHBUTTON      "キャンセル",IDCANCEL,162,172,50,14
    GROUPBOX        "",IDC_STATIC,6,36,296,25
    GROUPBOX        "",IDC_STATIC,6,65,296,25
    GROUPBOX        "",IDC_STATIC,6,95,296,42
    GROUPBOX        "",IDC_STATIC,6,141,296,26
END

IDD_SSHFWDSETUP DIALOG DISCARDABLE  0, 0, 309, 189
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "TTSSH: ポート転送設定"
FONT 10, "ＭＳ Ｐゴシック"
BEGIN
    GROUPBOX        "ポート転送(&P)",IDC_STATIC,6,7,296,120
    LISTBOX         IDC_SSHFWDLIST,17,20,275,76,LBS_SORT |
                    LBS_NOINTEGRALHEIGHT | WS_VSCROLL | WS_GROUP |
                    WS_TABSTOP
    PUSHBUTTON      "追加(&A)...",IDC_ADD,63,105,50,14,WS_GROUP
    PUSHBUTTON      "編集(&E)...",IDC_EDIT,129,105,50,14
    PUSHBUTTON      "削除(&R)",IDC_REMOVE,195,105,50,14
    GROUPBOX        "Xクライアントアプリケーションの転送",IDC_STATIC,6,132,
                    296,27
    CONTROL         "リモートの&Xアプリケーションをローカルのxサーバに表示する",
                    IDC_SSHFWDX11,"Button",BS_AUTOCHECKBOX | WS_GROUP |
                    WS_TABSTOP,18,142,248,14
    DEFPUSHBUTTON   "OK",IDOK,86,168,50,14,WS_GROUP
    PUSHBUTTON      "キャンセル",IDCANCEL,173,168,50,14
END

IDD_SSHFWDEDIT DIALOG DISCARDABLE  0, 0, 309, 146
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "SSHポート転送"
FONT 10, "ＭＳ Ｐゴシック"
BEGIN
    LTEXT           "ポート転送を行う向きの選択:",IDC_STATIC,6,7,176,11
    CONTROL         "ローカルのポート(&L)",IDC_SSHFWDLOCALTOREMOTE,"Button",
                    BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP,13,32,77,10
    CONTROL         "リモートサーバのポート(&R)",IDC_SSHFWDREMOTETOLOCAL,
                    "Button",BS_AUTORADIOBUTTON,13,81,97,10
    GROUPBOX        "",IDC_STATIC,6,21,296,46
    COMBOBOX        IDC_SSHLTRFROMPORT,93,31,56,126,CBS_DROPDOWN |
                    WS_VSCROLL | WS_GROUP | WS_TABSTOP
    LTEXT           "から",IDC_STATIC,154,33,21,8
    LTEXT           "リモート側ホスト(&M)",IDC_STATIC,23,50,61,8
    EDITTEXT        IDC_SSHLTRTOHOST,88,48,80,13,ES_AUTOHSCROLL | WS_GROUP
    LTEXT           "のポート(&P)",IDC_STATIC,172,50,36,8
    COMBOBOX        IDC_SSHLTRTOPORT,213,48,56,126,CBS_DROPDOWN | WS_VSCROLL |
                    WS_GROUP | WS_TABSTOP
    GROUPBOX        "",IDC_STATIC,6,71,296,46
    COMBOBOX        IDC_SSHRTLFROMPORT,114,80,56,126,CBS_DROPDOWN |
                    WS_VSCROLL | WS_GROUP | WS_TABSTOP
    LTEXT           "から",IDC_STATIC,175,83,21,8
    LTEXT           "ローカル側ホスト(&A)",IDC_STATIC,23,100,62,8
    EDITTEXT        IDC_SSHRTLTOHOST,88,98,80,13,ES_AUTOHSCROLL | WS_GROUP
    LTEXT           "のポート(&O)",IDC_STATIC,172,100,36,8
    COMBOBOX        IDC_SSHRTLTOPORT,213,98,56,126,CBS_DROPDOWN | WS_VSCROLL |
                    WS_GROUP | WS_TABSTOP
    DEFPUSHBUTTON   "OK",IDOK,96,125,50,14,WS_GROUP
    PUSHBUTTON      "キャンセル",IDCANCEL,162,125,50,14
    LTEXT           "へ転送",IDC_STATIC,273,50,25,8
    LTEXT           "へ転送",IDC_STATIC,273,101,25,8
END
```

```
IDD_SSHTISAUTH DIALOG DISCARDABLE  0, 0, 216, 109
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "SSH認証チャレンジ"
FONT 10, "ＭＳ Ｐゴシック"
BEGIN
    EDITTEXT        IDC_SSHPASSWORD,13,69,188,12,ES_PASSWORD |
                    ES_AUTOHSCROLL | WS_GROUP
    DEFPUSHBUTTON   "OK",IDOK,50,88,50,14,WS_GROUP
    PUSHBUTTON      "切断(&D)",IDCANCEL,116,88,50,14
    LTEXT           "pink-floyd.compose.cs.cmu.eduにログインします",
                    IDC_SSHAUTHBANNER,6,7,203,8
    LTEXT           "",IDC_SSHAUTHBANNER2,6,16,203,48
END

IDD_VERSIONDLG DIALOG DISCARDABLE  0, 0, 223, 123
STYLE DS_MODALFRAME | DS_3DLOOK | WS_POPUP | WS_VISIBLE | WS_CAPTION |
    WS_SYSMENU
CAPTION "FTP wrapperのバージョン情報"
FONT 9, "ＭＳ Ｐゴシック"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,79,102,50,14
    ICON            IDI_ICON2,IDC_STATIC,7,18,20,20,SS_REALSIZEIMAGE
    LTEXT           "FTP wrapper  バージョン 0.01 (build 364)\n(c) Copyright 1998-200
1 Lancelot, All Rights Reserved.",
                    IDC_STATIC,30,13,186,51
    LTEXT           "ホームページ:",IDC_STATIC,30,83,45,8
    LTEXT           "ほげ",IDC_URI,82,83,134,12,SS_NOTIFY
    LTEXT           "メールアドレス:",IDC_STATIC,30,71,48,8
    LTEXT           "ふが",IDC_EMAILADDR,82,71,134,12,SS_NOTIFY
END

IDD_SETUPDLG DIALOG DISCARDABLE  0, 0, 255, 212
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "基本設定"
FONT 9, "ＭＳ Ｐゴシック"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,69,191,50,14,WS_GROUP
    PUSHBUTTON      "キャンセル",IDCANCEL,135,191,50,14
    GROUPBOX        "プロトコル",IDC_STATIC,7,7,241,45
    CONTROL         "&POP3",IDC_PROTOPOP3,"Button",BS_AUTORADIOBUTTON |
                    WS_GROUP,13,20,35,10
    CONTROL         "&IMAP4",IDC_PROTOIMAP4,"Button",BS_AUTORADIOBUTTON |
                    WS_DISABLED,13,34,37,10
    LTEXT           "認証方式(&A):",IDC_STATIC,120,26,44,8
    COMBOBOX        IDC_AUTHMETHOD,169,25,72,72,CBS_DROPDOWNLIST | CBS_SORT |
                    CBS_NOINTEGRALHEIGHT | WS_VSCROLL | WS_GROUP |
                    WS_TABSTOP
    GROUPBOX        "ローカルのポート番号",IDC_STATIC,7,56,133,64
    CONTROL         "pop3 (&110)",IDC_LOCALPORT_STD,"Button",
                    BS_AUTORADIOBUTTON | WS_GROUP,13,71,51,10
    CONTROL         "その他(&O):",IDC_LOCALPORT_OTHER,"Button",
                    BS_AUTORADIOBUTTON,13,86,50,10
    EDITTEXT        IDC_LOCALPORT,63,84,40,14,ES_AUTOHSCROLL | ES_NOHIDESEL |
                    ES_NUMBER | WS_GROUP
    CONTROL         "リモートホストからの接続を拒否(&D)",IDC_REJECTREMOTE,
                    "Button",BS_AUTOCHECKBOX | WS_GROUP | WS_TABSTOP,14,103,
                    124,10
    GROUPBOX        "接続先ポート番号",IDC_STATIC,145,56,103,64
    CONTROL         "pop&3 (110)",IDC_REMOTEPORT_STD,"Button",
                    BS_AUTORADIOBUTTON | WS_GROUP,152,71,51,10
    CONTROL         "その他(&T):",IDC_REMOTEPORT_OTHER,"Button",
                    BS_AUTORADIOBUTTON,152,86,49,10
    EDITTEXT        IDC_REMOTEPORT,201,84,40,14,ES_AUTOHSCROLL |
                    ES_NOHIDESEL | ES_NUMBER | WS_GROUP
    GROUPBOX        "接続先ホスト",IDC_STATIC,7,124,241,58
    CONTROL         "IPアドレス(&N):",IDC_REMOTEHOST_IP,"Button",
```



```
                    BS_AUTORADIOBUTTON | WS_GROUP,13,141,60,10
    CONTROL         "ホスト名(&H):",IDC_REMOTEHOST_NAME,"Button",
                    BS_AUTORADIOBUTTON,13,161,54,10
    CONTROL         "IPAddress2",IDC_REMOTEIP,"SysIPAddress32",WS_GROUP |
                    WS_TABSTOP,73,139,100,15
    EDITTEXT        IDC_REMOTENAME,73,160,100,13,ES_AUTOHSCROLL | WS_GROUP
END

IDD_LOGSETUPDLG DIALOG DISCARDABLE  0, 0, 246, 190
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "ログ設定"
FONT 9, "ＭＳ Ｐゴシック"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,65,169,50,14,WS_GROUP
    PUSHBUTTON      "キャンセル",IDCANCEL,131,169,50,14
    CONTROL         "Slider1",IDC_SLIDER1,"msctls_trackbar32",TBS_TOP |
                    WS_TABSTOP,76,22,130,15
    CONTROL         "ファイルへ出力(&F):",IDC_CHECK1,"Button",
                    BS_AUTOCHECKBOX | WS_GROUP | WS_TABSTOP,13,106,74,10
    CONTROL         "ケロちゃんにおまかせ(&K)",IDC_CHECK2,"Button",
                    BS_AUTOCHECKBOX | WS_DISABLED | WS_GROUP | WS_TABSTOP,15,
                    141,95,10
    GROUPBOX        "おまけ",IDC_STATIC,7,128,232,31
    PUSHBUTTON      "参照(&B)",IDC_BUTTON1,192,104,40,14,WS_GROUP
    EDITTEXT        IDC_EDIT1,89,104,97,14,ES_AUTOHSCROLL | WS_GROUP
    LTEXT           "ログレベル(&L):",IDC_STATIC,15,25,46,8
    CTEXT           "レベル1: イベント・エラー",IDC_STATIC,96,13,89,8
    LTEXT           "ＭＳゴシック 10pt",IDC_STATIC,56,54,88,8
    LTEXT           "時刻",IDC_STATIC,17,78,20,8
    PUSHBUTTON      "Button2",IDC_BUTTON3,41,75,22,14,BS_BITMAP
    PUSHBUTTON      "Button2",IDC_BUTTON4,98,75,22,14,BS_BITMAP
    LTEXT           "情報",IDC_STATIC,73,78,24,8
    PUSHBUTTON      "Button2",IDC_BUTTON5,155,75,22,14,BS_BITMAP
    LTEXT           "エラー",IDC_STATIC,129,78,22,8
    PUSHBUTTON      "Button2",IDC_BUTTON6,208,75,22,14,BS_BITMAP
    LTEXT           "背景",IDC_STATIC,184,78,22,8
    PUSHBUTTON      "変更(&C)",IDC_BUTTON7,163,51,39,14
    LTEXT           "フォント:",IDC_STATIC,17,55,26,8
END


/////////////////////////////////////////////////////////////////////////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_ABOUTDIALOG, DIALOG
    BEGIN
        LEFTMARGIN, 6
        RIGHTMARGIN, 255
        TOPMARGIN, 7
        BOTTOMMARGIN, 242
    END

    IDD_SSHAUTH, DIALOG
    BEGIN
        LEFTMARGIN, 6
        RIGHTMARGIN, 302
        TOPMARGIN, 7
        BOTTOMMARGIN, 207
    END

    IDD_HOSTDLG, DIALOG
```

```
    BEGIN
        LEFTMARGIN, 4
        RIGHTMARGIN, 238
        BOTTOMMARGIN, 109
    END

    IDD_SSHSETUP, DIALOG
    BEGIN
        BOTTOMMARGIN, 191
    END

    IDD_SSHUNKNOWNHOST, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 208
        TOPMARGIN, 7
        BOTTOMMARGIN, 109
    END

    IDD_SSHDIFFERENTHOST, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 208
        TOPMARGIN, 7
        BOTTOMMARGIN, 107
    END

    IDD_SSHAUTHSETUP, DIALOG
    BEGIN
        LEFTMARGIN, 6
        RIGHTMARGIN, 302
        TOPMARGIN, 7
        BOTTOMMARGIN, 186
    END

    IDD_SSHFWDSETUP, DIALOG
    BEGIN
        LEFTMARGIN, 6
        RIGHTMARGIN, 302
        TOPMARGIN, 7
        BOTTOMMARGIN, 182
    END

    IDD_SSHFWDEDIT, DIALOG
    BEGIN
        LEFTMARGIN, 6
        RIGHTMARGIN, 302
        TOPMARGIN, 7
        BOTTOMMARGIN, 139
    END

    IDD_SSHTISAUTH, DIALOG
    BEGIN
        LEFTMARGIN, 6
        RIGHTMARGIN, 209
        TOPMARGIN, 7
        BOTTOMMARGIN, 102
    END
    IDD_VERSIONDLG, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 216
        TOPMARGIN, 7
        BOTTOMMARGIN, 116
    END
```



```
    IDD_SETUPDLG, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 248
        TOPMARGIN, 7
        BOTTOMMARGIN, 205
    END

    IDD_LOGSETUPDLG, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 239
        VERTGUIDE, 7
        TOPMARGIN, 7
        BOTTOMMARGIN, 183
    END
END
#endif    // APSTUDIO_INVOKED


////////////////////////////////////////////////////////////////////////////
//
// Icon
//

// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDI_ICON1               ICON    DISCARDABLE     "icon1.ico"
IDI_ICON2               ICON    DISCARDABLE     "icon2.ico"
IDI_ICON3               ICON    DISCARDABLE     "icon3.ico"
IDI_SECURETT            ICON    DISCARDABLE     "ttsecure.ico"

////////////////////////////////////////////////////////////////////////////
//
// Accelerator
//

IDR_ACCELERATOR1 ACCELERATORS DISCARDABLE
BEGIN
    "S",            ID_ACCEL40011,          VIRTKEY, CONTROL, NOINVERT
    VK_CANCEL,      ID_ACCEL40012,          VIRTKEY, CONTROL, NOINVERT
END


////////////////////////////////////////////////////////////////////////////
//
// Dialog Info
//

IDD_SETUPDLG DLGINIT
BEGIN
    IDC_AUTHMETHOD, 0x403, 5, 0
0x5041, 0x504f, "\000"
    0
END


////////////////////////////////////////////////////////////////////////////
//
// Cursor
//

IDC_HAND                CURSOR  DISCARDABLE     "H_POINT.CUR"

////////////////////////////////////////////////////////////////////////////
//
```

```
// String Table
//

STRINGTABLE DISCARDABLE
BEGIN
    IDS_ERR_PORTNUM         "ポート番号には0～65535の値を指定してください．"
    IDS_ERR_EXECMULTIPLE    "同時に複数のFTP wrapperを実行することはできません．"
    IDS_STATBAR_IDLE        "待機中"
    IDS_APPNAME             "FTP wrapper"
    IDS_ERR_DEFAULT         "このイベントに対するメッセージは用意されていません．(エ
ラーコード: %d)"
    IDS_CAPTION_ERR         "FTP wrapper: エラー"
    IDS_ERR_WINSOCKVER      "Winsockの初期化に失敗しました．"
    IDS_STATBAR_STOP        "停止中"
    IDS_INFO_STARTSVC       "サービスを開始しました．"
    IDS_INFO_ENDSVC         "サービスを停止しました．"
    IDS_EVENT_ACCEPT        "クライアントから接続されました．"
    IDS_EVENT_CONNECT       "サーバに接続しました．"
    IDS_EVENT_DISCONNECT    "通信のセッションを終了しました．"
    IDS_ERR_TOOSHORTRES     "サーバのレスポンスコードが短すぎます．"
    IDS_ERR_NOTRES          "サーバのレスポンスコードが不正です．"
END

STRINGTABLE DISCARDABLE
BEGIN
    IDS_ERR_NOTAUTHSTATE    "トランザクション段階で認証コマンドが発行されました．"
    IDS_ERR_NOTTRANSSTATE   "認証段階でトランザクションコマンドが発行されました．"
    IDS_ERR_NOTIMESTAMP     "POPサーバのグリーティングメッセージ中にタイムスタンプが
見つかりませんでした．APOP認証は利用できません．"
    IDS_ERR_NOUSER          "USERコマンドが正しく発行されずにPASSコマンドが発行されま
した．"
    IDS_ERR_NOPASSWORD      "PASSコマンドにパスワードが含まれていません．"
    IDS_ERR_INVALIDCMD      "無効なコマンドが発行されました．"
    IDS_EVENT_RESTART       "サービスを再起動しました．"
    IDS_POPMENU_RUN         "サービスの開始(&R)"
    IDS_POPMENU_STOP        "サービスの停止(&S)"
    IDS_POPMENU_CFG         "設定(&C)"
    IDS_POPMENU_SHOW        "ウィンドウを表示(&H)"
    IDS_POPMENU_EXIT        "プログラムの終了(&X)"
    IDS_ERR_NOOPIEDLL       "opie.dllがロードできませんでした．"
    IDS_ERR_NOOPIEGEN       "opiegeneratorのプロシージャエントリポイントを取得できま
せんでした．"
    IDS_ERR_OTPGEN          "OTP文字列の生成に失敗しました．"
    IDS_ERR_NOOTPCHALLENGE  "POPサーバのレスポンス中にOTPチャレンジ文字列が見つかりま
せんでした．OTP認証は利用できません．"
END

STRINGTABLE DISCARDABLE
BEGIN
    WSAEINTR                "関数呼び出しが中断されました．"
    WSAEBADF                "不良なファイルディスクリプタです．"
    WSAEACCES               "アクセスが拒否されました．"
    WSAEFAULT               "不良なアドレスです．"
END

STRINGTABLE DISCARDABLE
BEGIN
    WSAEINVAL               "無効な引数です．"
    WSAEMFILE               "開いているソケットの数が多すぎます．"
END

STRINGTABLE DISCARDABLE
BEGIN
    WSAEWOULDBLOCK          "操作はブロッキングされます．"
    WSAEINPROGRESS          "ブロッキング操作は既に実行中です．"
    WSAEALREADY             "操作は既に実行中です．"
```



```
    WSAENOTSOCK             "指定されたソケットは無効です．"
    WSAEDESTADDRREQ         "受け側アドレスが必要です．"
    WSAEMSGSIZE             "メッセージが長すぎます．"
    WSAEPROTOTYPE           "ソケットに対するプロトコルタイプが間違っています．"
    WSAENOPROTOOPT          "不良なプロトコルオプションです．"
    WSAEPROTONOSUPPORT      "そのプロトコルはサポートされていません．"
    WSAESOCKTNOSUPPORT      "そのソケットタイプはサポートされていません．"
    WSAEOPNOTSUPP           "その操作はサポートされていません．"
    WSAEPFNOSUPPORT         "そのプロトコルファミリはサポートされていません．"
    WSAEAFNOSUPPORT         "そのアドレスファミリはプロトコルファミリでサポートされま
せん．"
END

STRINGTABLE DISCARDABLE
BEGIN
    WSAEADDRINUSE           "アドレスは既に使用されています．"
    WSAEADDRNOTAVAIL        "無効なアドレスです．"
    WSAENETDOWN             "ネットワークがダウンしています．"
    WSAENETUNREACH          "ネットワークに到達できません．"
    WSAENETRESET            "ネットワーク接続が破棄されました．"
    WSAECONNABORTED         "ソフトウェアによって接続が中断されました．"
    WSAECONNRESET           "接続は相手によってリセットされました．"
    WSAENOBUFS              "バッファに空き領域がありません．"
    WSAEISCONN              "ソケットは既に接続されています．"
    WSAENOTCONN             "ソケットは接続されていません．"
    WSAESHUTDOWN            "ソケットのシャットダウンの後には通信できません．"
    WSAETOOMANYREFS         "参照の数が多すぎます．"
    WSAETIMEDOUT            "接続がタイムアウトしました．"
    WSAECONNREFUSED         "接続が拒否されました．"
    WSAELOOP                "ループしています．"
    WSAENAMETOOLONG         "名前が長すぎます．"
END

STRINGTABLE DISCARDABLE
BEGIN
    WSAEHOSTDOWN            "ホストがダウンしています．"
    WSAEHOSTUNREACH         "ホストへのルートがありません．"
    WSAENOTEMPTY            "ディレクトリは空ではありません．"
    WSAEPROCLIM             "プロセスの数が多すぎます．"
    WSAEUSERS               "ユーザの数が多すぎます．"
    WSAEDQUOT               "ディスククォータ．"
    WSAESTALE               "ストールしました．"
    WSAEREMOTE              "リモートのエラーです．"
END

STRINGTABLE DISCARDABLE
BEGIN
    WSASYSNOTREADY          "ネットワークサブシステムが利用できません．"
    WSAVERNOTSUPPORTED      "サポートされていないWinsockバージョンです．"
    WSANOTINITIALISED       "Winsockが初期化されていません．"
END

STRINGTABLE DISCARDABLE
BEGIN
    WSAEDISCON              "シャットダウン処理中です．"
    WSAENOMORE              "データはこれ以上ありません．"
    WSAECANCELLED           "検索が取り消されました．"
    WSAEINVALIDPROCTABLE    "サービスプロバイダからのプロシージャテーブルが無効です．
"
    WSAEINVALIDPROVIDER     "無効なサービスプロバイダバージョン番号です．"
    WSAEPROVIDERFAILEDINIT  "サービスプロバイダを初期化できません．"
    WSASYSCALLFAILURE       "システムコールが失敗しました．"
    WSASERVICE_NOT_FOUND    "サービスが見つかりません．"
    WSATYPE_NOT_FOUND       "タイプが見つかりません．"
    WSA_E_NO_MORE           "データはこれ以上ありません．"
    WSA_E_CANCELLED         "検索が取り消されました．"
```

```
END

STRINGTABLE DISCARDABLE
BEGIN
    WSAEREFUSED             "接続が拒否されました．"
END

STRINGTABLE DISCARDABLE
BEGIN
    WSAHOST_NOT_FOUND       "ホストが見つかりません．"
    WSATRY_AGAIN            "ホストが見つからないかサーバの異常です．"
    WSANO_RECOVERY          "回復不能なエラーです．"
    WSANO_DATA              "名前は有効ですが，要求したタイプのデータレコードはありま
せん．"
END

STRINGTABLE DISCARDABLE
BEGIN
    IDS_ERR_CANTGETPRIVILEDGEDPORT
                            "予約ポートを取得できませんでした．RPOP認証は利用できませ
ん．"
    IDS_ERR_NOSSLEAYDLL     "ssleay32.dllがロードできませんでした．"
    IDS_ERR_NOLIBEAYDLL     "libeay32.dllがロードできませんでした．"
    IDS_ERR_NOSSLPROC       "SSLeayのプロシージャエントリポイントを取得できませんでし
た．"
    IDS_ERR_SSLERROR        "SSLエラー:"
    IDS_ERR_SSLINIT         "SSL初期化エラーです．"
    IDS_MSG_SSLCIPHER       "接続に使用している暗号方式"
END

#endif    // 日本語 resources
////////////////////////////////////////////////////////////////////////////



#ifndef APSTUDIO_INVOKED
////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//


////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
This code is copyright (C) 1998-1999 Robert O'Callahan.
See LICENSE.TXT for the license.
*/

#include "ttxssh.h"

//#include "x11util.h"

#include <assert.h>

#define WM_SOCK_ACCEPT (WM_APP+9999)
#define WM_SOCK_IO     (WM_APP+9998)
#define WM_SOCK_GOTNAME (WM_APP+9997)

#define CHANNEL_READ_BUF_SIZE 30000

static LRESULT CALLBACK accept_wnd_proc(HWND wnd, UINT msg, WPARAM wParam, LPARAM lPa
ram);

static int find_request_num(PTInstVar pvar, SOCKET s) {
  int i;

  if (s == INVALID_SOCKET) return -1;

  for (i = 0; i < pvar->fwd_state.num_requests; i++) {
    if ((pvar->fwd_state.requests[i].status & FWD_DELETED) == 0
      && pvar->fwd_state.requests[i].listening_socket == s) {
      return i;
    }
  }

  return -1;
```



```
}

static int find_channel_num(PTInstVar pvar, SOCKET s) {
  int i;

  if (s == INVALID_SOCKET) return -1;

  for (i = 0; i < pvar->fwd_state.num_channels; i++) {
    if (pvar->fwd_state.channels[i].local_socket == s) {
      return i;
    }
  }

  return -1;
}

static int find_request_num_from_async_request(PTInstVar pvar, HANDLE request) {
  int i;

  if (request == 0) return -1;

  for (i = 0; i < pvar->fwd_state.num_requests; i++) {
    if ((pvar->fwd_state.requests[i].status & FWD_DELETED) == 0
      && pvar->fwd_state.requests[i].to_host_lookup_handle == request) {
      return i;
    }
  }

  return -1;
}

static void drain_matching_messages(HWND wnd, UINT msg, WPARAM wParam) {
  MSG m;
  MSG FAR * buf;
  int buf_len;
  int buf_size;
  int i;

  /* fast path for case where there are no suitable messages */
  if (!PeekMessage(&m, wnd, msg, msg, PM_NOREMOVE)) {
    return;
  }

  /* suck out all the messages */
  buf_size = 1;
  buf_len = 0;
  buf = (MSG FAR *)malloc(sizeof(MSG)*buf_size);

  while (PeekMessage(&m, wnd, msg, msg, PM_REMOVE)) {
    if (buf_len == buf_size) {
      buf_size *= 2;
      buf = (MSG FAR *)realloc(buf, sizeof(MSG)*buf_size);
    }

    buf[buf_len] = m;
    buf_len++;
  }

  for (i = 0; i < buf_len; i++) {
    if (buf[i].wParam != wParam) {
      PostMessage(wnd, msg, buf[i].wParam, buf[i].lParam);
    }
  }

  free(buf);
}
```

```
/* hideous hack to fix broken Winsock API.
   After we close a socket further WM_ messages may arrive for it, according to
   the docs for WSAAsyncSelect. This sucks because we may allocate a new socket
   before these messages are processed, which may get the same handle value as
   the old socket, which may mean that we get confused and try to process the old
   message according to the new request or channel, which could cause chaos and
   possibly even security problems.
   "Solution": scan the accept_wnd message queue whenever we close a socket and
   remove all pending messages for that socket. Possibly this might not even be
   enough if there is a system thread that is in the process of posting
   a new message while during execution of the closesocket. I'm hoping
   that those actions are serialized...
*/
static void safe_closesocket(PTInstVar pvar, SOCKET s) {
  HWND wnd = pvar->fwd_state.accept_wnd;

  closesocket(s);
  if (wnd != NULL) {
    drain_matching_messages(wnd, WM_SOCK_ACCEPT, (WPARAM)s);
    drain_matching_messages(wnd, WM_SOCK_IO, (WPARAM)s);
  }
}

static void safe_WSACancelAsyncRequest(PTInstVar pvar, HANDLE request) {
  HWND wnd = pvar->fwd_state.accept_wnd;

  WSACancelAsyncRequest(request);
  if (wnd != NULL) {
    drain_matching_messages(wnd, WM_SOCK_GOTNAME, (WPARAM)request);
  }
}

static int check_local_channel_num(PTInstVar pvar, int local_num) {
  if (local_num < 0 || local_num >= pvar->fwd_state.num_channels
    || pvar->fwd_state.channels[local_num].status == 0) {
    notify_nonfatal_error(pvar, "サーバが存在しない転送チャネルを操作しようとしました
．\n"
      "サーバソフトウェアにバグがあるか、悪意ある攻撃の可能性があります．この接続を閉
じて下さい．");
    return 0;
  } else {
    return 1;
  }
}

static void request_error(PTInstVar pvar, int request_num, int err) {
  SOCKET s = pvar->fwd_state.requests[request_num].listening_socket;
  char str[512], errstr[256];

  if (s != INVALID_SOCKET) {
    safe_closesocket(pvar, s);
    pvar->fwd_state.requests[request_num].listening_socket = INVALID_SOCKET;
  }

  LoadString(hInst, WSAGetLastError(), errstr, sizeof(errstr));
  wsprintf(str, "転送ポートの接続待ちの際に通信エラーが発生しました．(%s)\n"
      "接続待ちポートは終了します．", errstr);
  notify_nonfatal_error(pvar, str);
}

static void send_local_connection_closure(PTInstVar pvar, int channel_num) {
  FWDChannel FAR * channel = pvar->fwd_state.channels + channel_num;

  if ((channel->status & (FWD_REMOTE_CONNECTED | FWD_LOCAL_CONNECTED))
    == (FWD_REMOTE_CONNECTED | FWD_LOCAL_CONNECTED))  {
```



```
    SSH_channel_input_eof(pvar, channel->remote_num);
    SSH_channel_output_eof(pvar, channel->remote_num);
    channel->status |= FWD_CLOSED_LOCAL_IN | FWD_CLOSED_LOCAL_OUT;
  }
}

static void closed_local_connection(PTInstVar pvar, int channel_num) {
  FWDChannel FAR * channel = pvar->fwd_state.channels + channel_num;

  if (channel->local_socket != INVALID_SOCKET) {
    safe_closesocket(pvar, channel->local_socket);
    channel->local_socket = INVALID_SOCKET;

    send_local_connection_closure(pvar, channel_num);
  }
}

/* This gets called when a request becomes both deleted and has no
   active channels. */
static void really_delete_request(PTInstVar pvar, int request_num) {
  FWDRequest FAR * request = pvar->fwd_state.requests + request_num;

  if (request->to_host_lookup_handle != 0) {
    safe_WSACancelAsyncRequest(pvar, request->to_host_lookup_handle);
    request->to_host_lookup_handle = 0;
  }
  free(request->to_host_hostent_buf);
  request->to_host_hostent_buf = NULL;
}

static void free_channel(PTInstVar pvar, int i) {
  FWDChannel FAR * channel = &pvar->fwd_state.channels[i];

  UTIL_destroy_sock_write_buf(&channel->writebuf);
  if (channel->filter != NULL) {
    channel->filter(channel->filter_closure, 0, NULL, NULL);
    channel->filter = NULL;
    channel->filter_closure = NULL;
  }
  channel->status = 0;
  if (channel->local_socket != INVALID_SOCKET) {
    safe_closesocket(pvar, channel->local_socket);
    channel->local_socket = INVALID_SOCKET;
  }

  if (channel->request_num >= 0) {
    FWDRequest FAR * request = &pvar->fwd_state.requests[channel->request_num];

    request->num_channels--;
    if (request->num_channels == 0 && (request->status & FWD_DELETED) != 0) {
      really_delete_request(pvar, channel->request_num);
    }
    channel->request_num = -1;
  }
}

void FWD_channel_input_eof(PTInstVar pvar, uint32 local_channel_num) {
  FWDChannel FAR * channel;

  if (!check_local_channel_num(pvar, local_channel_num)) return;

  channel = pvar->fwd_state.channels + local_channel_num;

  if (channel->local_socket != INVALID_SOCKET) {
    shutdown(channel->local_socket, 1);
  }
```

```
  channel->status |= FWD_CLOSED_REMOTE_IN;
  if ((channel->status & FWD_CLOSED_REMOTE_OUT) == FWD_CLOSED_REMOTE_OUT) {
    closed_local_connection(pvar, local_channel_num);
    free_channel(pvar, local_channel_num);
  }
}

void FWD_channel_output_eof(PTInstVar pvar, uint32 local_channel_num) {
  FWDChannel FAR * channel;

  if (!check_local_channel_num(pvar, local_channel_num)) return;

  channel = pvar->fwd_state.channels + local_channel_num;

  if (channel->local_socket != INVALID_SOCKET) {
    shutdown(channel->local_socket, 0);
  }
  channel->status |= FWD_CLOSED_REMOTE_OUT;
  if ((channel->status & FWD_CLOSED_REMOTE_IN) == FWD_CLOSED_REMOTE_IN) {
    closed_local_connection(pvar, local_channel_num);
    free_channel(pvar, local_channel_num);
  }
}

static char FAR * describe_socket_error(int code) {
  switch (code) {
  case WSAECONNREFUSED:
    return "接続が拒否されました(サービスが現在行われていない可能性があります)";
  case WSAENETDOWN:
  case WSAENETUNREACH:
  case WSAEHOSTUNREACH:
    return "接続先ホストに到達できませんでした(ネットワークに問題がある可能性があります)";
  case WSAETIMEDOUT:
  case WSAEHOSTDOWN:
    return "接続先ホストに到達できませんでした(ホストが動作していない可能性があります)";
  case WSATRY_AGAIN:
  case WSANO_RECOVERY:
  case WSANO_ADDRESS:
  case WSAHOST_NOT_FOUND:
    return "ホストのアドレスが見つかりませんでした";
  default:
    return "ポート転送のための接続を確立できませんでした";
  }
}

static void channel_error(PTInstVar pvar, char FAR * action, int channel_num, int err) {
  char FAR * err_msg;

  closed_local_connection(pvar, channel_num);

  switch (err) {
  case WSAECONNRESET:
  case WSAECONNABORTED:
    err_msg = NULL;
    break;
  default:
    err_msg = describe_socket_error(err);
  }

  if (err_msg != NULL) {
    char buf[1024];

    _snprintf(buf, sizeof(buf), "ローカルポート%sの%s時の通信エラーです．\n"
```



```
      "%s (エラーコード %d)\n"
      "ポート転送の接続を閉じます．",
      pvar->fwd_state.requests[pvar->fwd_state.channels[channel_num].request_num].spec.from_port_name,
      action,
      err_msg, err);
    buf[sizeof(buf) - 1] = 0;
    notify_nonfatal_error(pvar, buf);
  }
}

static void channel_opening_error(PTInstVar pvar, int channel_num, int err) {
  char buf[1024];
  FWDChannel FAR * channel = &pvar->fwd_state.channels[channel_num];
  FWDRequest FAR * request = &pvar->fwd_state.requests[channel->request_num];

  SSH_fail_channel_open(pvar, channel->remote_num);
  if (request->spec.type == FWD_REMOTE_X11_TO_LOCAL) {
    _snprintf(buf, sizeof(buf), "サーバはホスト %s 上のXサーバ(スクリーン%d)への接続の転送を試みました．\n"
      "%s．\n"
      "ポート転送の接続を閉じます．",
      request->spec.to_host, request->spec.to_port - 6000,
      describe_socket_error(err));
  } else {
    _snprintf(buf, sizeof(buf), "サーバはこのホストを通じてホスト %s、ポート%dへの接続の転送を試みました．\n"
      "%s．\n"
      "ポート転送の接続を閉じます．",
      request->spec.to_host, request->spec.to_port,
      describe_socket_error(err));
  }
  buf[sizeof(buf) - 1] = 0;
  notify_nonfatal_error(pvar, buf);
  free_channel(pvar, channel_num);
}

static void init_local_IP_numbers(PTInstVar pvar) {
  HOSTENT FAR * hostent;
  char buf[1024];
  int num_addrs = 0;
  uint32 FAR * addrs;

  if (!gethostname(buf, sizeof(buf))) {
    hostent = gethostbyname(buf);

    if (hostent != NULL) {
      for (; hostent->h_addr_list[num_addrs] != NULL; num_addrs++) {
      }
    }
  } else {
    hostent = NULL;
  }

  addrs = (uint32 FAR *)malloc(sizeof(uint32)*(num_addrs + 1));
  pvar->fwd_state.local_host_IP_numbers = addrs;
  if (hostent != NULL) {
    int i;

    for (i = 0; i < num_addrs; i++) {
      addrs[i] = ntohl(*(uint32 FAR *)hostent->h_addr_list[i]);
    }
  }
  addrs[num_addrs] = 0;
}
```

```
static BOOL validate_IP_number(PTInstVar pvar, uint32 addr) {
  int i;

  if (pvar->settings.LocalForwardingIdentityCheck) {
    /* Should we allow a wider range of loopback addresses here?
       i.e. 127.xx.xx.xx
       Wouldn't want to introduce a security hole if there's
       some OS bug that lets an intruder get packets from us
       to such an address */
    if (addr == INADDR_LOOPBACK) {
      return TRUE;
    }

    if (pvar->fwd_state.local_host_IP_numbers == NULL) {
      init_local_IP_numbers(pvar);
    }

    for (i = 0; pvar->fwd_state.local_host_IP_numbers[i] != 0; i++) {
      if (pvar->fwd_state.local_host_IP_numbers[i] == addr) {
        return TRUE;
      }
    }

    return FALSE;
  } else {
    return TRUE;
  }
}

int alloc_channel(PTInstVar pvar, int new_status, int new_request_num) {
  int i;
  int new_num_channels;
  int new_channel = -1;
  FWDChannel FAR * channel;

  for (i = 0; i < pvar->fwd_state.num_channels && new_channel < 0; i++) {
    if (pvar->fwd_state.channels[i].status == 0) {
      new_channel = i;
    }
  }

  if (new_channel < 0) {
    new_num_channels = pvar->fwd_state.num_channels*2 + 1;
    pvar->fwd_state.channels = (FWDChannel FAR *)realloc(pvar->fwd_state.channels,
      sizeof(FWDChannel)*new_num_channels);

    for (; i < new_num_channels; i++) {
      channel = pvar->fwd_state.channels + i;

      channel->status = 0;
      channel->local_socket = INVALID_SOCKET;
      channel->request_num = -1;
      channel->filter = NULL;
      channel->filter_closure = NULL;
      UTIL_init_sock_write_buf(&channel->writebuf);
    }

    new_channel = pvar->fwd_state.num_channels;
    pvar->fwd_state.num_channels = new_num_channels;
  }

  channel = pvar->fwd_state.channels + new_channel;

  channel->status = new_status;
  channel->request_num = new_request_num;
  pvar->fwd_state.requests[new_request_num].num_channels++;
```



```
  UTIL_init_sock_write_buf(&channel->writebuf);

  return new_channel;
}

static HWND make_accept_wnd(PTInstVar pvar) {
  if (pvar->fwd_state.accept_wnd == NULL) {
    pvar->fwd_state.accept_wnd = CreateWindow("STATIC", "TTSSHポート転送モニタ",
      WS_DISABLED | WS_POPUP, 0, 0, 1, 1, NULL, NULL, hInst, NULL);
    if (pvar->fwd_state.accept_wnd != NULL) {
      pvar->fwd_state.old_accept_wnd_proc = (WNDPROC)SetWindowLong(pvar->fwd_state.ac
cept_wnd, GWL_WNDPROC, (LONG)accept_wnd_proc);
      SetWindowLong(pvar->fwd_state.accept_wnd, GWL_USERDATA, (LONG)pvar);
    }
  }

  return pvar->fwd_state.accept_wnd;
}

static void connected_local_connection(PTInstVar pvar, int channel_num) {
  SSH_confirm_channel_open(pvar, pvar->fwd_state.channels[channel_num].remote_num,
    channel_num);
  pvar->fwd_state.channels[channel_num].status |= FWD_LOCAL_CONNECTED;
}

static void make_local_connection(PTInstVar pvar, int channel_num) {
  FWDChannel FAR * channel = pvar->fwd_state.channels + channel_num;
  FWDRequest FAR * request = pvar->fwd_state.requests + channel->request_num;
  struct sockaddr_in addr;

  addr.sin_family = AF_INET;
  addr.sin_port = htons((unsigned short)request->spec.to_port);
  addr.sin_addr.s_addr = htonl(request->to_host_addr);

  if ((channel->local_socket = socket(AF_INET, SOCK_STREAM, 0)) != INVALID_SOCKET
    && WSAAsyncSelect(channel->local_socket, make_accept_wnd(pvar), WM_SOCK_IO,
      FD_CONNECT | FD_READ | FD_CLOSE | FD_WRITE) != SOCKET_ERROR) {
    if (connect(channel->local_socket, (struct sockaddr FAR *)&addr, sizeof(addr)) !=
 SOCKET_ERROR) {
      connected_local_connection(pvar, channel_num);
      return;
    } else if (WSAGetLastError() == WSAEWOULDBLOCK) {
      /* do nothing, we'll just wait */
      return;
    }
  }

  channel_opening_error(pvar, channel_num, WSAGetLastError());
}

static void accept_local_connection(PTInstVar pvar, int request_num) {
  int channel_num;
  SOCKET s;
  struct sockaddr addr;
  int addrlen = sizeof(addr);
  char buf[1024];
  BYTE FAR * IP;
  FWDChannel FAR * channel;
  FWDRequest FAR * request = &pvar->fwd_state.requests[request_num];

  s = accept(request->listening_socket, &addr, &addrlen);
  if (s == INVALID_SOCKET) return;

  IP = (BYTE FAR *)&((struct sockaddr_in *)(&addr))->sin_addr.s_addr;

  if (!validate_IP_number(pvar, ntohl(((struct sockaddr_in *)(&addr))->sin_addr.S_un.
```

```
S_addr))) {
    _snprintf(buf, sizeof(buf), "IPアドレス %d.%d.%d.%d のホストがローカルの転送ポートへの接続を試みようとしました．\n"
      "ある種の悪意を持った攻撃である可能性があります！",
      IP[0], IP[1], IP[2], IP[3], request->spec.from_port);
    buf[NUM_ELEM(buf) - 1] = 0;
    notify_nonfatal_error(pvar, buf);
    safe_closesocket(pvar, s);
    return;
  }

  _snprintf(buf, sizeof(buf),
    "Host %d.%d.%d.%d connecting to port %d; forwarding to %s:%d",
    IP[0], IP[1], IP[2], IP[3], request->spec.from_port, request->spec.to_host,
    request->spec.to_port);
  buf[NUM_ELEM(buf) - 1] = 0;
  notify_verbose_message(pvar, buf, LOG_LEVEL_VERBOSE);

  _snprintf(buf, sizeof(buf), "%d.%d.%d.%d", IP[0], IP[1], IP[2], IP[3]);
  buf[NUM_ELEM(buf) - 1] = 0;

  channel_num = alloc_channel(pvar, FWD_LOCAL_CONNECTED, request_num);
  channel = pvar->fwd_state.channels + channel_num;

  channel->local_socket = s;
  channel->filter_closure = NULL;
  channel->filter = NULL;

  SSH_open_channel(pvar, channel_num, request->spec.to_host,
    request->spec.to_port, buf);
}

static void write_local_connection_buffer(PTInstVar pvar, int channel_num) {
  FWDChannel FAR * channel = pvar->fwd_state.channels + channel_num;

  if ((channel->status & (FWD_REMOTE_CONNECTED | FWD_LOCAL_CONNECTED))
    == (FWD_REMOTE_CONNECTED | FWD_LOCAL_CONNECTED))  {
    if (!UTIL_sock_write_more(pvar, &channel->writebuf, channel->local_socket)) {
      channel_error(pvar, "書き込み", channel_num, WSAGetLastError());
    }
  }
}

static void read_local_connection(PTInstVar pvar, int channel_num) {
  FWDChannel FAR * channel = pvar->fwd_state.channels + channel_num;

  if ((channel->status & (FWD_REMOTE_CONNECTED | FWD_LOCAL_CONNECTED))
    != (FWD_REMOTE_CONNECTED | FWD_LOCAL_CONNECTED))  {
    return;
  }

  while (channel->local_socket != INVALID_SOCKET) {
    char buf[CHANNEL_READ_BUF_SIZE];
    int amount = recv(channel->local_socket, buf, sizeof(buf), 0);
    int err;

    if (amount > 0) {
      char FAR * new_buf = buf;
      int action = FWD_FILTER_RETAIN;

      if (channel->filter != NULL) {
        action = channel->filter(channel->filter_closure, FWD_FILTER_FROM_CLIENT,
            &amount, &new_buf);
      }

      if (amount > 0 && (channel->status & FWD_CLOSED_REMOTE_OUT) == 0) {
```



```
        SSH_channel_send(pvar, channel->remote_num, new_buf, amount);
      }

      switch (action) {
      case FWD_FILTER_REMOVE:
        channel->filter(channel->filter_closure, 0, NULL, NULL);
        channel->filter = NULL;
        channel->filter_closure = NULL;
        break;
      case FWD_FILTER_CLOSECHANNEL:
        closed_local_connection(pvar, channel_num);
        break;
      }
    } else if (amount == 0 || (err = WSAGetLastError()) == WSAEWOULDBLOCK) {
      return;
    } else {
      channel_error(pvar, "読み取り", channel_num, err);
      return;
    }
  }
}

static void failed_to_host_addr(PTInstVar pvar, int request_num, int err) {
  int i;

  for (i = 0; i < pvar->fwd_state.num_channels; i++) {
    if (pvar->fwd_state.channels[i].request_num == request_num) {
      channel_opening_error(pvar, i, err);
    }
  }
}

static void found_to_host_addr(PTInstVar pvar, int request_num) {
  int i;

  pvar->fwd_state.requests[request_num].to_host_addr =
    ntohl(*(uint32 FAR *)((HOSTENT FAR *)pvar->fwd_state.requests[request_num].to_hos
t_hostent_buf)->h_addr_list[0]);

  for (i = 0; i < pvar->fwd_state.num_channels; i++) {
    if (pvar->fwd_state.channels[i].request_num == request_num) {
      make_local_connection(pvar, i);
    }
  }
}

static LRESULT CALLBACK accept_wnd_proc(HWND wnd, UINT msg, WPARAM wParam, LPARAM lPa
ram) {
  PTInstVar pvar = (PTInstVar)GetWindowLong(wnd, GWL_USERDATA);

  if (msg == WM_SOCK_ACCEPT
    && (LOWORD(lParam) == FD_READ || LOWORD(lParam) == FD_CLOSE
      || LOWORD(lParam) == FD_WRITE)) {
    msg = WM_SOCK_IO;
  }

  switch (msg) {
  case WM_SOCK_ACCEPT: {
    int request_num = find_request_num(pvar, (SOCKET)wParam);

    if (request_num < 0) return TRUE;

    if (HIWORD(lParam) != 0) {
      request_error(pvar, request_num, HIWORD(lParam));
    } else {
      switch (LOWORD(lParam)) {
```

```
      case FD_ACCEPT:
        accept_local_connection(pvar, request_num);
        break;
      }
    }
    return TRUE;
  }

  case WM_SOCK_GOTNAME: {
    int request_num = find_request_num_from_async_request(pvar, (HANDLE)wParam);

    if (request_num < 0) return TRUE;

    if (HIWORD(lParam) != 0) {
      failed_to_host_addr(pvar, request_num, HIWORD(lParam));
    } else {
      found_to_host_addr(pvar, request_num);
    }
    pvar->fwd_state.requests[request_num].to_host_lookup_handle = 0;
    free(pvar->fwd_state.requests[request_num].to_host_hostent_buf);
    pvar->fwd_state.requests[request_num].to_host_hostent_buf = NULL;
    return TRUE;
  }

  case WM_SOCK_IO: {
    int channel_num = find_channel_num(pvar, (SOCKET)wParam);

    if (channel_num < 0) return TRUE;

    if (HIWORD(lParam) != 0) {
      if (LOWORD(lParam) == FD_CONNECT) {
        channel_opening_error(pvar, channel_num, HIWORD(lParam));
      } else {
        channel_error(pvar, "アクセス", channel_num, HIWORD(lParam));
      }
    } else {
      switch (LOWORD(lParam)) {
      case FD_CONNECT:
        connected_local_connection(pvar, channel_num);
        break;
      case FD_READ:
        read_local_connection(pvar, channel_num);
        break;
      case FD_CLOSE:
        read_local_connection(pvar, channel_num);
        closed_local_connection(pvar, channel_num);
        break;
      case FD_WRITE:
        write_local_connection_buffer(pvar, channel_num);
        break;
      }
    }
    return TRUE;
  }
  }

  return CallWindowProc(pvar->fwd_state.old_accept_wnd_proc, wnd, msg, wParam, lParam
);
}

int FWD_compare_specs(void const FAR * void_spec1, void const FAR * void_spec2) {
  FWDRequestSpec FAR * spec1 = (FWDRequestSpec FAR *)void_spec1;
  FWDRequestSpec FAR * spec2 = (FWDRequestSpec FAR *)void_spec2;
  int delta = spec1->from_port - spec2->from_port;

  if (delta == 0) {
```



```
    delta = spec1->type - spec2->type;
  }

  return delta;
}

/* Check if the server can listen for 'spec' using the old listening spec 'listener'.
   We check that the to_port and (for non-X connections) the to_host are equal
   so that we never lie to the server about where its forwarded connection is
   ending up. Maybe some SSH implementation depends on this information being
   reliable, for security? */
static BOOL can_server_listen_using(FWDRequestSpec FAR * listener, FWDRequestSpec FAR
 * spec) {
  return listener->type == spec->type
    && listener->from_port == spec->from_port
    && listener->to_port == spec->to_port
    && (spec->type == FWD_REMOTE_X11_TO_LOCAL
      || strcmp(listener->to_host, spec->to_host) == 0);
}

BOOL FWD_can_server_listen_for(PTInstVar pvar, FWDRequestSpec FAR * spec) {
  int num_server_listening_requests = pvar->fwd_state.num_server_listening_specs;

  if (num_server_listening_requests < 0) {
    return TRUE;
  } else {
    FWDRequestSpec FAR * listener =
      bsearch(spec, pvar->fwd_state.server_listening_specs, num_server_listening_requ
ests,
        sizeof(FWDRequestSpec), FWD_compare_specs);

    if (listener == NULL) {
      return FALSE;
    } else {
      return can_server_listen_using(listener, spec);
    }
  }
}

int FWD_get_num_request_specs(PTInstVar pvar) {
  int num_request_specs = 0;
  int i;

  for (i = 0; i < pvar->fwd_state.num_requests; i++) {
    if ((pvar->fwd_state.requests[i].status & FWD_DELETED) == 0) {
      num_request_specs++;
    }
  }

  return num_request_specs;
}

void FWD_get_request_specs(PTInstVar pvar, FWDRequestSpec FAR * specs, int num_specs)
 {
  int i;

  for (i = 0; i < pvar->fwd_state.num_requests && num_specs > 0; i++) {
    if ((pvar->fwd_state.requests[i].status & FWD_DELETED) == 0) {
      *specs = pvar->fwd_state.requests[i].spec;
      num_specs--;
      specs++;
    }
  }
}

/* This function can be called while channels remain open on the request.
```

```
   Take care.
   It returns the listening socket for the request, if there is one.
   The caller must close this socket if it is not INVALID_SOCKET.
*/
static SOCKET delete_request(PTInstVar pvar, int request_num) {
  FWDRequest FAR * request = pvar->fwd_state.requests + request_num;
  SOCKET result = INVALID_SOCKET;

/* safe to shut down the listening socket here. Any pending connections
   that haven't yet been turned into channels will be broken, but that's
   just tough luck. */
  if (request->listening_socket != INVALID_SOCKET) {
    result = request->listening_socket;
    request->listening_socket = INVALID_SOCKET;
  }

  request->status |= FWD_DELETED;

  if (request->num_channels == 0) {
    really_delete_request(pvar, request_num);
  }

  return result;
}

static BOOL are_specs_identical(FWDRequestSpec FAR * spec1, FWDRequestSpec FAR * spec
2) {
  return spec1->type == spec2->type
    && spec1->from_port == spec2->from_port
    && spec1->to_port == spec2->to_port
    && strcmp(spec1->to_host, spec2->to_host) == 0;
}

static BOOL interactive_init_request(PTInstVar pvar, int request_num, BOOL report_err
or) {
  FWDRequest FAR * request = pvar->fwd_state.requests + request_num;

//
// by Lancelot: if type2 is set, local socket for listen is not prepared
//
  if (request->spec.type == FWD_LOCAL_TO_REMOTE && request->spec.type2 == 0) {
    SOCKET s = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in addr;

    addr.sin_family = AF_INET;
    addr.sin_port = htons((unsigned short)request->spec.from_port);
    addr.sin_addr.s_addr = htonl(INADDR_ANY);

    request->listening_socket = s;
    if (s == INVALID_SOCKET
      || WSAAsyncSelect(s, make_accept_wnd(pvar), WM_SOCK_ACCEPT, FD_ACCEPT | FD_READ
 | FD_CLOSE | FD_WRITE) == SOCKET_ERROR
      || bind(s, (struct sockaddr FAR *)&addr, sizeof(addr)) == SOCKET_ERROR
      || listen(s, SOMAXCONN) == SOCKET_ERROR) {
      if (report_error) {
                char str[512], errstr[256];
                LoadString(hInst, WSAGetLastError(), errstr, sizeof(errstr));
                wsprintf(str, "ポート転送に必要ないくつかのソケットを初期化できません
でした．(%s)\n"
                        "ポート転送サービスが利用できない可能性があります．", errstr)
;
                notify_nonfatal_error(pvar, str);
      }
      return FALSE;
    }
  }
```



```
  return TRUE;
}

/* This function will only be called on a request when all its channels are
   closed. */
static BOOL init_request(PTInstVar pvar, int request_num,
  BOOL report_error, SOCKET listening_socket) {
  FWDRequest FAR * request = pvar->fwd_state.requests + request_num;

  request->listening_socket = INVALID_SOCKET;
  request->to_host_addr = 0;
  request->to_host_hostent_buf = NULL;
  request->to_host_lookup_handle = 0;
  request->status = 0;

  if (pvar->fwd_state.in_interactive_mode) {
    if (listening_socket != INVALID_SOCKET) {
      request->listening_socket = listening_socket;
      return TRUE;
    } else {
      return interactive_init_request(pvar, request_num, report_error);
    }
  } else {
    assert(listening_socket == INVALID_SOCKET);
    return TRUE;
  }
}

void FWD_set_request_specs(PTInstVar pvar, FWDRequestSpec FAR * specs, int num_specs)
 {
  FWDRequestSpec FAR * new_specs = (FWDRequestSpec FAR *)malloc(sizeof(FWDRequestSpec
)*num_specs);
  char FAR * specs_accounted_for;
  SOCKET FAR * saved_sockets;
  int i;
  int num_new_requests = num_specs;
  int num_free_requests = 0;
  int free_request = 0;
  BOOL report_err = TRUE;

  memcpy(new_specs, specs, sizeof(FWDRequestSpec)*num_specs);
  qsort(new_specs, num_specs, sizeof(FWDRequestSpec), FWD_compare_specs);

  for (i = 0; i < num_specs - 1; i++) {
    if (FWD_compare_specs(new_specs + i, new_specs + i + 1) == 0) {
      notify_nonfatal_error(pvar, "TTSSH内部エラー: タイプ・ポートの重複したポート転
送が検出されたため、ポート転送を設定できませんでした．");
      free(new_specs);
      return;
    }
  }

  specs_accounted_for = (char FAR *)malloc(sizeof(char)*num_specs);
  saved_sockets = (SOCKET FAR *)malloc(sizeof(SOCKET)*num_specs);

  memset(specs_accounted_for, 0, num_specs);
  for (i = 0; i < num_specs; i++) {
    saved_sockets[i] = INVALID_SOCKET;
  }

  for (i = pvar->fwd_state.num_requests - 1; i >= 0; i--) {
    if ((pvar->fwd_state.requests[i].status & FWD_DELETED) == 0) {
      FWDRequestSpec FAR * cur_spec = &pvar->fwd_state.requests[i].spec;
      FWDRequestSpec FAR * new_spec = bsearch(cur_spec, new_specs, num_specs, sizeof(
FWDRequestSpec), FWD_compare_specs);
```

```
      if (new_spec != NULL && are_specs_identical(cur_spec, new_spec)) {
        specs_accounted_for[new_spec - new_specs] = 1;
        num_new_requests--;
      } else {
        SOCKET listening_socket = delete_request(pvar, i);

        if (new_spec != NULL) {
          saved_sockets[new_spec - new_specs] = listening_socket;
        } else if (listening_socket != INVALID_SOCKET) {
          safe_closesocket(pvar, listening_socket);
        }

        if (pvar->fwd_state.requests[i].num_channels == 0) {
          num_free_requests++;
        }
      }
    } else {
      if (pvar->fwd_state.requests[i].num_channels == 0) {
        num_free_requests++;
      }
    }
  }

  if (num_new_requests > num_free_requests) {
    int total_requests =
      pvar->fwd_state.num_requests + num_new_requests - num_free_requests;

    pvar->fwd_state.requests =
      (FWDRequest FAR *)realloc(pvar->fwd_state.requests, sizeof(FWDRequest)*total_re
quests);
    for (i = pvar->fwd_state.num_requests; i < total_requests; i++) {
      pvar->fwd_state.requests[i].status = FWD_DELETED;
      pvar->fwd_state.requests[i].num_channels = 0;
    }
    pvar->fwd_state.num_requests = total_requests;
  }

  for (i = 0; i < num_specs; i++) {
    if (!specs_accounted_for[i]) {
      while ((pvar->fwd_state.requests[free_request].status & FWD_DELETED) == 0
        || pvar->fwd_state.requests[free_request].num_channels != 0) {
        free_request++;
      }

      assert(free_request < pvar->fwd_state.num_requests);

      pvar->fwd_state.requests[free_request].spec = new_specs[i];
      if (!init_request(pvar, free_request, report_err, saved_sockets[i])) {
        report_err = FALSE;
      }

      free_request++;
    }
  }

  free(saved_sockets);
  free(specs_accounted_for);
  free(new_specs);
}

void FWD_prep_forwarding(PTInstVar pvar) {
  int i;
  int num_server_listening_requests = 0;

  for (i = 0; i < pvar->fwd_state.num_requests; i++) {
```



```
    FWDRequest FAR * request = pvar->fwd_state.requests + i;

    if ((request->status & FWD_DELETED) == 0) {
      switch (request->spec.type) {
      case FWD_REMOTE_TO_LOCAL:
        SSH_request_forwarding(pvar, request->spec.from_port,
          request->spec.to_host, request->spec.to_port);
        num_server_listening_requests++;
        break;
      }
    }
  }

  pvar->fwd_state.num_server_listening_specs = num_server_listening_requests;

  if (num_server_listening_requests > 0) {
    FWDRequestSpec FAR * server_listening_requests =
      (FWDRequestSpec FAR *)malloc(sizeof(FWDRequestSpec)*num_server_listening_reques
ts);

    pvar->fwd_state.server_listening_specs = server_listening_requests;

    for (i = 0; i < pvar->fwd_state.num_requests; i++) {
      if ((pvar->fwd_state.requests[i].status & FWD_DELETED) == 0) {
        switch (pvar->fwd_state.requests[i].spec.type) {
        case FWD_REMOTE_TO_LOCAL:
          *server_listening_requests = pvar->fwd_state.requests[i].spec;
          server_listening_requests++;
          break;
        }
      }
    }

    /* No two server-side request specs can have the same port. */
    qsort(pvar->fwd_state.server_listening_specs, num_server_listening_requests,
      sizeof(FWDRequestSpec), FWD_compare_specs);
  }
}

void FWD_enter_interactive_mode(PTInstVar pvar) {
  BOOL report_error = TRUE;
  int i;

  pvar->fwd_state.in_interactive_mode = TRUE;

  for (i = 0; i < pvar->fwd_state.num_requests; i++) {
        /* modified 2001/06/02 Lancelot */
        if( pvar->fwd_state.requests[i].status & FWD_DELETED )
                continue;
    if (!interactive_init_request(pvar, i, report_error)) {
      report_error = FALSE;
    }
  }
}

static void create_local_channel(PTInstVar pvar, uint32 remote_channel_num,
                                 int request_num, void * filter_closure, FWDFilter filter)
 {
  char buf[1024];
  int channel_num;
  FWDChannel FAR * channel;
  FWDRequest FAR * request = pvar->fwd_state.requests + request_num;

  if (request->to_host_addr == 0
    && request->to_host_lookup_handle == 0) {
    HANDLE task_handle;
```

```
    if (request->to_host_hostent_buf == NULL) {
      request->to_host_hostent_buf = (char FAR *)malloc(MAXGETHOSTSTRUCT);
    }

    task_handle =
      WSAAsyncGetHostByName(make_accept_wnd(pvar), WM_SOCK_GOTNAME,
      request->spec.to_host, request->to_host_hostent_buf, MAXGETHOSTSTRUCT);

    if (task_handle == 0) {
      SSH_fail_channel_open(pvar, remote_channel_num);
      _snprintf(buf, sizeof(buf), "サーバはこのホストを通じてホスト %s、ポート%dへの
接続の転送を試みましたが、"
        "処理中にエラーが発生したため、転送は拒否されました．",
        request->spec.to_host, request->spec.to_port_name);
      buf[sizeof(buf) - 1] = 0;
      notify_nonfatal_error(pvar, buf);
      free(request->to_host_hostent_buf);
      request->to_host_hostent_buf = NULL;
      return;
    } else {
      request->to_host_lookup_handle = task_handle;
    }
  }

  channel_num = alloc_channel(pvar, FWD_REMOTE_CONNECTED, request_num);
  channel = pvar->fwd_state.channels + channel_num;

  channel->remote_num = remote_channel_num;
  channel->filter_closure = filter_closure;
  channel->filter = filter;

  if (request->to_host_addr != 0) {
    make_local_connection(pvar, channel_num);
  }
}

void FWD_open(PTInstVar pvar, uint32 remote_channel_num,
              char FAR * local_hostname, int local_port, char FAR * originator, int o
riginator_len) {
  int i;
  char buf[1024];

  for (i = 0; i < pvar->fwd_state.num_requests; i++) {
    FWDRequest FAR * request = pvar->fwd_state.requests + i;

    if ((request->status & FWD_DELETED) == 0
      && request->spec.type == FWD_REMOTE_TO_LOCAL
      && request->spec.to_port2 == local_port
//      && strcmp(request->spec.to_host, local_hostname) == 0
        ) {
      create_local_channel(pvar, remote_channel_num, i, NULL, NULL);
      return;
    }
  }

  SSH_fail_channel_open(pvar, remote_channel_num);

  /* now, before we panic, maybe we TOLD the server we could forward this port
     and then the user changed the settings. */
  for (i = 0; i < pvar->fwd_state.num_server_listening_specs; i++) {
    FWDRequestSpec FAR * spec = pvar->fwd_state.server_listening_specs + i;

    if (spec->type == FWD_REMOTE_TO_LOCAL
      && spec->to_port == local_port
      && strcmp(spec->to_host, local_hostname) == 0) {
```



```
      return; /* no error message needed. The user just turned this off, that's all *
/
    }
  }

  /* this forwarding was not prespecified */
  _snprintf(buf, sizeof(buf), "サーバはこのホストを通じてホスト %s、ポート%dへの接続
の転送を試みましたが、"
    "あらかじめTTSSHにこの転送設定がされていないため、転送要求を拒否しました．",
    local_hostname, local_port);
  buf[sizeof(buf) - 1] = 0;
  notify_nonfatal_error(pvar, buf);
}

void FWD_X11_open(PTInstVar pvar, uint32 remote_channel_num,
  char FAR * originator, int originator_len) {
  int i;

  for (i = 0; i < pvar->fwd_state.num_requests; i++) {
    FWDRequest FAR * request = pvar->fwd_state.requests + i;

  }

  SSH_fail_channel_open(pvar, remote_channel_num);

  /* now, before we panic, maybe we TOLD the server we could forward this port
     and then the user changed the settings. */
  for (i = 0; i < pvar->fwd_state.num_server_listening_specs; i++) {
    FWDRequestSpec FAR * spec = pvar->fwd_state.server_listening_specs + i;

    if (spec->type == FWD_REMOTE_X11_TO_LOCAL) {
      return; /* no error message needed. The user just turned this off, that's all *
/
    }
  }

  /* this forwarding was not prespecified */
  notify_nonfatal_error(pvar, "サーバはローカルホストのXサーバへの接続の転送を試みま
したが、"
    "あらかじめTTSSHに転送設定がされていないため、転送要求を拒否しました．");
}

void FWD_confirmed_open(PTInstVar pvar, uint32 local_channel_num,
                        uint32 remote_channel_num) {
  SOCKET s;
  FWDChannel FAR * channel;

  if (!check_local_channel_num(pvar, local_channel_num)) return;

  channel = pvar->fwd_state.channels + local_channel_num;
  s = channel->local_socket;

  if( pvar->fwd_state.requests[channel->request_num].spec.type2 == 1 ) {
              channel->remote_num = remote_channel_num;
              channel->status |= FWD_REMOTE_CONNECTED;
              PostMessage(pvar->cv.HWin, WM_USER_NOTIFY_CHOPENRESULT, (WPARAM)1, (L
PARAM)pvar);
  } else {
          if (s != INVALID_SOCKET) {
              channel->remote_num = remote_channel_num;
              channel->status |= FWD_REMOTE_CONNECTED;

              read_local_connection(pvar, local_channel_num);
          } else {
              SSH_channel_input_eof(pvar, remote_channel_num);
              SSH_channel_output_eof(pvar, remote_channel_num);
```

```
                channel->status |= FWD_CLOSED_LOCAL_IN | FWD_CLOSED_LOCAL_OUT;
        }
  }
}

void FWD_failed_open(PTInstVar pvar, uint32 local_channel_num) {
  if (!check_local_channel_num(pvar, local_channel_num)) return;

  if( pvar->fwd_state.requests[pvar->fwd_state.channels[local_channel_num].request_nu
m].spec.type2 == 1 )
        PostMessage(pvar->cv.HWin, WM_USER_NOTIFY_CHOPENRESULT, (WPARAM)2, (LPARAM)pv
ar);

  notify_nonfatal_error(pvar, "ローカルホスト上のプログラムが転送ポートへの接続を試み
ましたが、"
    "T転送された要求はサーバに拒否されました．接続を閉じます．");
  free_channel(pvar, local_channel_num);
}

static BOOL blocking_write(PTInstVar pvar, SOCKET s, const char FAR * data, int lengt
h) {
  u_long do_block = 0;

  return WSAAsyncSelect(s, make_accept_wnd(pvar), 0, 0) == SOCKET_ERROR
    || ioctlsocket(s, FIONBIO, &do_block) == SOCKET_ERROR
    || send(s, data, length, 0) != length
    || WSAAsyncSelect(s, pvar->fwd_state.accept_wnd,
         WM_SOCK_ACCEPT, FD_READ | FD_CLOSE | FD_WRITE) == SOCKET_ERROR;
}

void FWD_received_data(PTInstVar pvar, uint32 local_channel_num,
                       unsigned char FAR * data, int length) {
  SOCKET s;
  FWDChannel FAR * channel;
  int action = FWD_FILTER_RETAIN;
  char str[512], errstr[256];
  unsigned int uSockErr;

  if (!check_local_channel_num(pvar, local_channel_num)) return;

  channel = pvar->fwd_state.channels + local_channel_num;

  if (channel->filter != NULL) {
    action = channel->filter(channel->filter_closure, FWD_FILTER_FROM_SERVER,
      &length, &data);
  }

  if( pvar->fwd_state.requests[channel->request_num].spec.type2 == 1 ) {
               _RPTF0(_CRT_WARN, "FWD_received_data() called.\n");
               OnSSHRead(pvar, local_channel_num, data, length);
  } else {
        s = channel->local_socket;
        if (length > 0 && s != INVALID_SOCKET) {
              if (!UTIL_sock_buffered_write(pvar, &channel->writebuf, blocking_writ
e,
                        s, data, length)) {
                uSockErr = WSAGetLastError();
                closed_local_connection(pvar, local_channel_num);
                LoadString(hInst, uSockErr, errstr, sizeof(errstr));
                wsprintf(str, "ローカルポートへデータを転送する際に通信エラーが発生
しました．(%s)\n"
                        "転送されていた接続は閉じられます．", errstr);
                notify_nonfatal_error(pvar, str);
              }
        }
  }
```



```
  switch (action) {
  case FWD_FILTER_REMOVE:
    channel->filter(channel->filter_closure, 0, NULL, NULL);
    channel->filter = NULL;
    channel->filter_closure = NULL;
    break;
  case FWD_FILTER_CLOSECHANNEL:
    closed_local_connection(pvar, local_channel_num);
    break;
  }
}

void FWD_init(PTInstVar pvar) {
  pvar->fwd_state.requests = NULL;
  pvar->fwd_state.num_requests = 0;
  pvar->fwd_state.num_server_listening_specs = -1; /* forwarding prep not yet done */
  pvar->fwd_state.server_listening_specs = NULL;
  pvar->fwd_state.num_channels = 0;
  pvar->fwd_state.channels = NULL;
  pvar->fwd_state.local_host_IP_numbers = NULL;
  pvar->fwd_state.X11_auth_data = NULL;
  pvar->fwd_state.accept_wnd = NULL;
  pvar->fwd_state.in_interactive_mode = FALSE;
}

void FWD_end(PTInstVar pvar) {
  int i;

  if (pvar->fwd_state.channels != NULL) {
    for (i = 0; i < pvar->fwd_state.num_channels; i++) {
      free_channel(pvar, i);
    }
    free(pvar->fwd_state.channels);
  }

  if (pvar->fwd_state.requests != NULL) {
    for (i = 0; i < pvar->fwd_state.num_requests; i++) {
      SOCKET s = delete_request(pvar, i);

      if (s != INVALID_SOCKET) {
        closesocket(s);
      }
    }
    free(pvar->fwd_state.requests);
  }

  free(pvar->fwd_state.local_host_IP_numbers);
  free(pvar->fwd_state.server_listening_specs);

  if (pvar->fwd_state.accept_wnd != NULL) {
    DestroyWindow(pvar->fwd_state.accept_wnd);
  }
}
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
This code is copyright (C) 1998-1999 Robert O'Callahan.
See LICENSE.TXT for the license.
*/

#ifndef __FWD_H
#define __FWD_H

#define FWD_REMOTE_CONNECTED 0x01
#define FWD_LOCAL_CONNECTED  0x02
#define FWD_CLOSED_REMOTE_IN 0x04
#define FWD_CLOSED_REMOTE_OUT 0x08
#define FWD_CLOSED_LOCAL_IN  0x10
#define FWD_CLOSED_LOCAL_OUT 0x20

#define FWD_FILTER_REMOVE       0
#define FWD_FILTER_RETAIN       1
#define FWD_FILTER_CLOSECHANNEL 2

#define FWD_FILTER_FROM_CLIENT 0
#define FWD_FILTER_FROM_SERVER 1

/* a length == 0 means that we're killing the channel and the filter
   should deallocate */
typedef int (* FWDFilter)(void FAR * closure, int direction,
  int FAR * length, unsigned char FAR * FAR * buf);

typedef struct {
  int status;
  int remote_num;
  SOCKET local_socket;
  int request_num;
  UTILSockWriteBuf writebuf;
```



```
  void FAR * filter_closure;
  FWDFilter filter;
} FWDChannel;

/* Request types */
#define FWD_LOCAL_TO_REMOTE             1
#define FWD_REMOTE_TO_LOCAL             2
#define FWD_REMOTE_X11_TO_LOCAL         3

/* If 'type' is FWD_REMOTE_X11_TO_LOCAL, then from_port must be
   -1, to_port must be 6000 + screen number, and to_host must
   be the desired X server host.

   There can be no more than one spec of a given type and from_port
   at one time.
*/
typedef struct {
  int type;
  int type2;
  int from_port;
  char from_port_name[32];
  int to_port;
  int to_port2;
  char to_port_name[32];
  char to_host[256];
} FWDRequestSpec;

#define FWD_DELETED                     0x01

typedef struct {
  SOCKET listening_socket;

  uint32 to_host_addr;
  char FAR * to_host_hostent_buf;
  HANDLE to_host_lookup_handle;

  int num_channels;

  int status;
  FWDRequestSpec spec;
} FWDRequest;

typedef struct {
  HWND accept_wnd;
  WNDPROC old_accept_wnd_proc;
  int num_server_listening_specs;
  /* stored in sorted order */
  FWDRequestSpec FAR * server_listening_specs;
  int num_requests;
  FWDRequest FAR * requests;
  int num_channels;
  FWDChannel FAR * channels;
  uint32 * local_host_IP_numbers;
  struct _X11AuthData FAR * X11_auth_data;
  BOOL in_interactive_mode;
} FWDState;

void FWD_init(PTInstVar pvar);
/* This function checks to see whether the server is (or can) listen for a
   given request. Before the SSH session's prep phase, this returns true for
   all requests. After the SSH session's prep phase, this returns true only
   for requests that the server actually was told about during the prep phase. */
BOOL FWD_can_server_listen_for(PTInstVar pvar, FWDRequestSpec FAR * spec);
int FWD_get_num_request_specs(PTInstVar pvar);
void FWD_get_request_specs(PTInstVar pvar, FWDRequestSpec FAR * specs, int num_specs)
```

```
;
void FWD_prep_forwarding(PTInstVar pvar);
void FWD_enter_interactive_mode(PTInstVar pvar);
void FWD_open(PTInstVar pvar, uint32 remote_channel_num,
  char FAR * local_hostname, int local_port, char FAR * originator, int originator_le
n);
void FWD_X11_open(PTInstVar pvar, uint32 remote_channel_num,
  char FAR * originator, int originator_len);
void FWD_confirmed_open(PTInstVar pvar, uint32 local_channel_num,
  uint32 remote_channel_num);
void FWD_failed_open(PTInstVar pvar, uint32 local_channel_num);
void FWD_received_data(PTInstVar pvar, uint32 local_channel_num,
  unsigned char FAR * data, int length);
void FWD_channel_input_eof(PTInstVar pvar, uint32 local_channel_num);
void FWD_channel_output_eof(PTInstVar pvar, uint32 local_channel_num);
void FWD_end(PTInstVar pvar);


#ifdef __cplusplus
extern "C" {
#endif

void FWD_set_request_specs(PTInstVar pvar, FWDRequestSpec FAR * specs, int num_specs)
;
int FWD_compare_specs(void const FAR * void_spec1, void const FAR * void_spec2);
int alloc_channel(PTInstVar pvar, int new_status, int new_request_num);

#ifdef __cplusplus
}
#endif

#endif
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
This code is copyright (C) 1998-1999 Robert O'Callahan.
See LICENSE.TXT for the license.
*/

#include "fwdui.h"

#include "resource.h"
//#include "x11util.h"

typedef struct {
  int port;
  char FAR * name;
} TCP_service_name;

typedef struct {
  FWDRequestSpec FAR * spec;
  PTInstVar pvar;
} FWDEditClosure;

static TCP_service_name service_DB[] = {
  { 384, "arns" },
  { 204, "at-echo" },
  { 202, "at-nbp" },
  { 201, "at-rtmp" },
  { 206, "at-zis" },
  { 705, "auditd" },
  { 113, "auth" },
  { 113, "authentication" },
  { 152, "bftp" },
  { 179, "bgp" },
  { 67, "bootp" },
  { 68, "bootpc" },
```



```
  { 67, "bootps" },
  { 881, "bshell" },
  { 1133, "bshelldbg" },
  { 877, "buildd" },
  { 2003, "cfinger" },
  { 19, "chargen" },
  { 531, "chat" },
  { 2030, "client" },
  { 514, "cmd" },
  { 164, "cmip-agent" },
  { 163, "cmip-man" },
  { 1431, "coda_aux1" },
  { 1433, "coda_aux2" },
  { 1435, "coda_aux3" },
  { 1407, "coda_backup" },
  { 1423, "codacon" },
  { 1285, "codger" },
  { 1541, "codgerdbg" },
  { 531, "conference" },
  { 1389, "controller" },
  { 530, "courier" },
  { 1100, "cpsys" },
  { 1346, "csd" },
  { 1602, "csddbg" },
  { 105, "csnet-ns" },
  { 105, "cso-ns" },
  { 1999, "cvskserver" },
  { 987, "daserver" },
  { 13, "daytime" },
  { 1429, "dbp" },
  { 10300, "dictionary" },
  { 9, "discard" },
  { 2100, "discuss" },
  { 1399, "disptool0" },
  { 1401, "disptool1" },
  { 1403, "disptool2" },
  { 1405, "disptool3" },
  { 53, "domain" },
  { 7000, "dos" },
  { 7, "echo" },
  { 8100, "eda1_mbx" },
  { 8101, "eda2_mbx" },
  { 8000, "eda_mbx" },
  { 520, "efs" },
  { 2105, "eklogin" },
  { 545, "ekshell" },
  { 1377, "erim" },
  { 1617, "erimdbg" },
  { 888, "erlogin" },
  { 512, "exec" },
  { 2001, "filesrv" },
  { 79, "finger" },
  { 21, "ftp" },
  { 20, "ftp-data" },
  { 1397, "gds_db" },
  { 70, "gopher" },
  { 1025, "greendbg" },
  { 5999, "grmd" },
  { 5710, "hcserver" },
  { 751, "hesupd" },
  { 101, "hostname" },
  { 101, "hostnames" },
  { 80, "http" },
  { 7489, "iasqlsvr" },
  { 113, "ident" },
  { 143, "imap" },
```

```
{ 143, "imap2" },
{ 220, "imap3" },
{ 1524, "ingreslock" },
{ 1234, "instsrv" },
{ 1387, "ipt" },
{ 213, "ipx" },
{ 194, "irc" },
{ 6667, "irc-alt" },
{ 883, "ishell" },
{ 102, "iso-tsap" },
{ 885, "itkt" },
{ 1439, "jeeves" },
{ 1389, "joysticknav" },
{ 1629, "joysticknavdbg" },
{ 544, "kcmd" },
{ 750, "kdc" },
{ 10401, "kdebug" },
{ 750, "kerberos" },
{ 749, "kerberos-adm" },
{ 88, "kerberos-sec" },
{ 751, "kerberos_master" },
{ 1445, "kjdbc" },
{ 543, "klogin" },
{ 561, "kopexec" },
{ 562, "kopshell" },
{ 761, "kpasswd" },
{ 1109, "kpop" },
{ 1110, "kpopr" },
{ 761, "kpwd" },
{ 88, "krb5" },
{ 4444, "krb524" },
{ 754, "krb5_prop" },
{ 754, "krb_prop" },
{ 760, "krbupdate" },
{ 545, "krcmd" },
{ 1443, "krcp" },
{ 760, "kreg" },
{ 544, "kshell" },
{ 750, "ktc" },
{ 549, "kxct" },
{ 5696, "lanmgrx.osb" },
{ 1441, "lcladm" },
{ 87, "link" },
{ 98, "linuxconf" },
{ 2766, "listen" },
{ 1025, "listener" },
{ 135, "loc-srv" },
{ 4045, "lockd" },
{ 513, "login" },
{ 25, "mail" },
{ 9535, "man" },
{ 3881, "mbatchd" },
{ 1393, "motionnav" },
{ 1633, "motionnavdbg" },
{ 7000, "msdos" },
{ 18, "msp" },
{ 57, "mtp" },
{ 42, "name" },
{ 42, "nameserver" },
{ 773, "nanny" },
{ 138, "nbdgm" },
{ 137, "nbns" },
{ 139, "nbssn" },
{ 1419, "ndim" },
{ 138, "netbios-dgm" },
{ 137, "netbios-ns" },
```



```
{ 139, "netbios-ssn" },
{ 138, "netbios_dgm" },
{ 137, "netbios_ns" },
{ 139, "netbios_ssn" },
{ 2106, "netdist" },
{ 1287, "netimage" },
{ 1543, "netimagedbg" },
{ 532, "netnews" },
{ 1353, "netreg" },
{ 1609, "netregdbg" },
{ 77, "netrjs" },
{ 15, "netstat" },
{ 1026, "network_terminal" },
{ 526, "newdate" },
{ 144, "news" },
{ 178, "nextstep" },
{ 1536, "nft" },
{ 43, "nicname" },
{ 119, "nntp" },
{ 1026, "nterm" },
{ 123, "ntp" },
{ 123, "ntpd" },
{ 9, "null" },
{ 891, "odexm" },
{ 560, "opcmd" },
{ 560, "opshell" },
{ 879, "pag" },
{ 893, "papyrus" },
{ 1379, "parvis" },
{ 1619, "parvisdbg" },
{ 600, "pcserver" },
{ 2026, "pcserverbulk" },
{ 2025, "pcserverrpc" },
{ 1385, "pharos" },
{ 1373, "pierunt" },
{ 1613, "pieruntdbg" },
{ 1351, "piesrv" },
{ 1607, "piesrvdbg" },
{ 1889, "pmlockd" },
{ 109, "pop" },
{ 109, "pop-2" },
{ 110, "pop-3" },
{ 109, "pop2" },
{ 110, "pop3" },
{ 111, "portmap" },
{ 111, "portmapper" },
{ 109, "postoffice" },
{ 170, "print-srv" },
{ 515, "printer" },
{ 191, "prospero" },
{ 1525, "prospero-np" },
{ 17, "qotd" },
{ 17, "quote" },
{ 601, "rauth" },
{ 5347, "rcisimmux" },
{ 1530, "rdeliver" },
{ 532, "readnews" },
{ 7815, "recserv" },
{ 64, "rem" },
{ 1025, "remote_file_sharing" },
{ 1026, "remote_login" },
{ 556, "remotefs" },
{ 3878, "res" },
{ 875, "resolve" },
{ 1131, "resolvedbg" },
{ 4672, "rfa" },
```

```
  { 5002, "rfe" },
  { 1025, "rfs" },
  { 556, "rfs_server" },
  { 1027, "rfsdbg" },
  { 77, "rje" },
  { 1260, "rlb" },
  { 1425, "rndb2" },
  { 530, "rpc" },
  { 111, "rpcbind" },
  { 1347, "rpl" },
  { 1603, "rpldbg" },
  { 107, "rtelnet" },
  { 3882, "sbatchd" },
  { 778, "serv" },
  { 1375, "service_warp" },
  { 1615, "service_warpdbg" },
  { 115, "sftp" },
  { 5232, "sgi-dgl" },
  { 514, "shell" },
  { 1135, "shelob" },
  { 1137, "shelobsrv" },
  { 9, "sink" },
  { 775, "sms_db" },
  { 777, "sms_update" },
  { 25, "smtp" },
  { 199, "smux" },
  { 108, "snagas" },
  { 19, "source" },
  { 6111, "spc" },
  { 515, "spooler" },
  { 23523, "sshprop" },
  { 1391, "statnav" },
  { 1621, "statnavdbg" },
  { 1395, "stm_switch" },
  { 1283, "sunmatrox" },
  { 1539, "sunmatroxdbg" },
  { 111, "sunrpc" },
  { 95, "supdup" },
  { 1127, "supfiledbg" },
  { 871, "supfilesrv" },
  { 1125, "supnamedbg" },
  { 869, "supnamesrv" },
  { 1529, "support" },
  { 11, "systat" },
  { 601, "ta-rauth" },
  { 113, "tap" },
  { 1381, "task_control" },
  { 1, "tcpmux" },
  { 23, "telnet" },
  { 24, "telnet2" },
  { 526, "tempo" },
  { 17, "text" },
  { 37, "time" },
  { 37, "timserver" },
  { 1600, "tnet" },
  { 102, "tsap" },
  { 87, "ttylink" },
  { 19, "ttytst" },
  { 372, "ulistserv" },
  { 119, "untp" },
  { 119, "usenet" },
  { 11, "users" },
  { 1400, "usim" },
  { 540, "uucp" },
  { 117, "uucp-path" },
  { 540, "uucpd" },
```



```
  { 1387, "vapor" },
  { 712, "vexec" },
  { 712, "vice-exec" },
  { 713, "vice-login" },
  { 714, "vice-shell" },
  { 1371, "visim" },
  { 1611, "visimdbg" },
  { 713, "vlogin" },
  { 714, "vshell" },
  { 210, "wais" },
  { 1427, "warplite" },
  { 765, "webster" },
  { 43, "whois" },
  { 1421, "wiztemp" },
  { 2000, "wm" },
  { 1383, "wnn" },
  { 22273, "wnn4" },
  { 22289, "wnn4_cn" },
  { 22273, "wnn4_jp" },
  { 1348, "worldc" },
  { 1604, "worldcdbg" },
  { 2401, "writesrv" },
  { 80, "www" },
  { 6000, "x-server" },
  { 103, "x400" },
  { 104, "x400-snd" },
  { 177, "xdmcp" },
  { 873, "xtermd" },
  { 210, "z3950" }
};

static int compare_services(void const FAR * elem1, void const FAR * elem2) {
  TCP_service_name FAR * s1 = (TCP_service_name FAR *)elem1;
  TCP_service_name FAR * s2 = (TCP_service_name FAR *)elem2;

  return strcmp(s1->name, s2->name);
}

static void make_X_forwarding_spec(FWDRequestSpec FAR * spec) {
  spec->type = FWD_REMOTE_X11_TO_LOCAL;
  spec->from_port = -1;
//  X11_get_DISPLAY_info(spec->to_host, sizeof(spec->to_host), &spec->to_port);
  strncpy(spec->from_port_name, "remote X server", sizeof(spec->from_port_name));
  spec->from_port_name[sizeof(spec->from_port_name) - 1] = 0;
  _snprintf(spec->to_port_name, sizeof(spec->to_port_name), "X server (screen %d)",
    spec->to_port - 6000);
  spec->to_port_name[sizeof(spec->to_port_name) - 1] = 0;
}

static BOOL is_service_name_char(char ch) {
  return (ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z')
    || ch == '_' || ch == '-' || (ch >= '0' && ch <= '9');
}

static int parse_port_from_buf(char FAR * buf) {
  int i;

  if (buf[0] >= '0' && buf[0] <= '9') {
    int result = atoi(buf);

    if (result < 1 || result > 65535) {
      return -1;
    } else {
      return result;
    }
  } else {
```

```
    char lower_buf[32];
    TCP_service_name FAR * result;
    TCP_service_name key;

    for (i = 0; buf[i] != 0 && i < sizeof(lower_buf) - 1; i++) {
      lower_buf[i] = tolower(buf[i]);
    }
    lower_buf[i] = 0;

    key.name = lower_buf;
    result = (TCP_service_name FAR *)
      bsearch(&key, service_DB, NUM_ELEM(service_DB), sizeof(service_DB[0]), compare_
services);

    if (result == NULL) {
      return -1;
    } else {
      return result->port;
    }
  }
}

static int parse_port(char FAR * FAR * str, char FAR * buf, int bufsize) {
  int i = 0;

  while (is_service_name_char(**str) && i < bufsize - 1) {
    buf[i] = **str;
    (*str)++;
    i++;
  }
  buf[i] = 0;

  return parse_port_from_buf(buf);
}

static BOOL parse_request(FWDRequestSpec FAR * request, char FAR * str) {
  char FAR * host_start;

  if (str[0] == 'L' || str[0] == 'l') {
    request->type = FWD_LOCAL_TO_REMOTE;
  } else if (str[0] == 'R' || str[0] == 'r') {
    request->type = FWD_REMOTE_TO_LOCAL;
  } else if (str[0] == 'X' || str[0] == 'x') {
    make_X_forwarding_spec(request);
    return TRUE;
  } else {
    return FALSE;
  }
  str++;

  request->from_port = parse_port(&str, request->from_port_name, sizeof(request->from
_port_name));
  if (request->from_port < 0) {
    return FALSE;
  }

  if (*str != ':') {
    return FALSE;
  }
  str++;

  host_start = str;
  while (*str != ':' && *str != 0 && *str != ';') {
    str++;
  }
  if (*str != ':') {
```



```
    return FALSE;
  }
  *str = 0;
  strncpy(request->to_host, host_start, sizeof(request->to_host));
  request->to_host[sizeof(request->to_host) - 1] = 0;
  *str = ':';
  str++;

  request->to_port = parse_port(&str, request->to_port_name, sizeof(request->to_port_
name));
  if (request->to_port < 0) {
    return FALSE;
  }

  if (*str != ';' && *str != 0) {
    return FALSE;
  }

  return TRUE;
}

void FWDUI_save_settings(PTInstVar pvar) {
  int num_specs = FWD_get_num_request_specs(pvar);
  FWDRequestSpec FAR * requests =
    (FWDRequestSpec FAR *)malloc(sizeof(FWDRequestSpec)*num_specs);
  int i;
  char FAR * str = pvar->settings.DefaultForwarding;
  int str_remaining = sizeof(pvar->settings.DefaultForwarding) - 1;

  FWD_get_request_specs(pvar, requests, num_specs);

  *str = 0;
  str[str_remaining] = 0;

  for (i = 0; i < num_specs; i++) {
    if (str_remaining > 0 && i > 0) {
      str[0] = ';';
      str[1] = 0;
      str++;
      str_remaining--;
    }

    if (str_remaining > 0) {
      FWDRequestSpec FAR * spec = requests + i;
      int chars;

      switch (spec->type) {
      case FWD_LOCAL_TO_REMOTE:
        _snprintf(str, str_remaining, "L%s:%s:%s",
          spec->from_port_name, spec->to_host, spec->to_port_name);
        break;
      case FWD_REMOTE_TO_LOCAL:
        _snprintf(str, str_remaining, "R%s:%s:%s",
          spec->from_port_name, spec->to_host, spec->to_port_name);
        break;
      case FWD_REMOTE_X11_TO_LOCAL:
        _snprintf(str, str_remaining, "X");
        break;
      }

      chars = strlen(str);
      str += chars;
      str_remaining -= chars;
    }
  }
  free(requests);
```

```
}

void FWDUI_load_settings(PTInstVar pvar) {
  char FAR * str = pvar->settings.DefaultForwarding;
//  if (str[0] != 0) {
    int i, ch, j = 0;
    FWDRequestSpec FAR * requests;

    if (str[0] != 0)
          j = 1;
    for (i = 0; (ch = str[i]) != 0; i++) {
      if (ch == ';') {
        j++;
      }
    }
    requests = (FWDRequestSpec FAR *)malloc(sizeof(FWDRequestSpec)*j);

    j = 0;
    if (parse_request(requests, str)) {
      j++;
    }
    for (i = 0; (ch = str[i]) != 0; i++) {
      if (ch == ';') {
        if (parse_request(requests + j, str + i + 1)) {
          j++;
        }
      }
    }

    qsort(requests, j, sizeof(FWDRequestSpec), FWD_compare_specs);

    for (i = 0; i < j - 1; i++) {
      if (FWD_compare_specs(requests + i, requests + i + 1) == 0) {
        memmove(requests + i, requests + i + 1, sizeof(FWDRequestSpec)*(j - 1 - i));
        i--;
        j--;
      }
    }

    FWD_set_request_specs(pvar, requests, j);
    FWDUI_save_settings(pvar);

    free(requests);
//  }
}

void FWDUI_init(PTInstVar pvar) {
}

void FWDUI_end(PTInstVar pvar) {
}

void FWDUI_open(PTInstVar pvar) {
}

static void set_verbose_port(char FAR * buf, int bufsize, int port, char FAR * name)
{
  if (*name >= '0' && *name <= '9') {
    strncpy(buf, name, bufsize);
  } else {
    _snprintf(buf, bufsize, "%d (%s)", port, name);
  }

  buf[bufsize - 1] = 0;
}
```



```
static void get_spec_string(FWDRequestSpec FAR * spec, char FAR * buf, int bufsize) {
  char verbose_from_port[64];
  char verbose_to_port[64];

  set_verbose_port(verbose_from_port, sizeof(verbose_from_port), spec->from_port, spe
c->from_port_name);
  set_verbose_port(verbose_to_port, sizeof(verbose_to_port), spec->to_port, spec->to_
port_name);

  switch (spec->type) {
  case FWD_REMOTE_TO_LOCAL:
    _snprintf(buf, bufsize, "リモートのポート %s からローカル側ホスト \"%s\" ポート %
s へ",
      verbose_from_port, spec->to_host, verbose_to_port);
    buf[bufsize - 1] = 0;
    break;
  case FWD_LOCAL_TO_REMOTE:
    _snprintf(buf, bufsize, "ローカルのポート %s からリモート側ホスト \"%s\" ポート %
s へ",
      verbose_from_port, spec->to_host, verbose_to_port);
    buf[bufsize - 1] = 0;
    break;
  case FWD_REMOTE_X11_TO_LOCAL:
    _snprintf(buf, bufsize, "リモートのXアプリケーションからローカルのXサーバへ");
    buf[bufsize - 1] = 0;
    return;
  }
}

static void update_listbox_selection(HWND dlg) {
  HWND listbox = GetDlgItem(dlg, IDC_SSHFWDLIST);
  int cursel = SendMessage(listbox, LB_GETCURSEL, 0, 0);

  EnableWindow(GetDlgItem(dlg, IDC_EDIT), cursel >= 0);
  EnableWindow(GetDlgItem(dlg, IDC_REMOVE), cursel >= 0);
}

static void init_listbox_selection(HWND dlg) {
  SendMessage(GetDlgItem(dlg, IDC_SSHFWDLIST), LB_SETCURSEL, 0, 0);
  update_listbox_selection(dlg);
}

static int add_spec_to_listbox(HWND dlg, FWDRequestSpec FAR * spec) {
   char buf[1024];
   HWND listbox = GetDlgItem(dlg, IDC_SSHFWDLIST);
   int index;

   get_spec_string(spec, buf, sizeof(buf));

   index = SendMessage(listbox, LB_ADDSTRING, 0, (LPARAM)buf);

   if (index >= 0) {
     FWDRequestSpec FAR * listbox_spec = malloc(sizeof(FWDRequestSpec));

     *listbox_spec = *spec;
     if (SendMessage(listbox, LB_SETITEMDATA, index, (LPARAM)listbox_spec) == LB_ERR)
 {
       free(listbox_spec);
     }
   }

   return index;
}

static void init_fwd_dlg(PTInstVar pvar, HWND dlg) {
  int num_specs = FWD_get_num_request_specs(pvar);
```

```
  FWDRequestSpec FAR * requests =
    (FWDRequestSpec FAR *)malloc(sizeof(FWDRequestSpec)*num_specs);
  int i;

  FWD_get_request_specs(pvar, requests, num_specs);

  for (i = 0; i < num_specs; i++) {
    if (requests[i].type == FWD_REMOTE_X11_TO_LOCAL) {
      CheckDlgButton(dlg, IDC_SSHFWDX11, TRUE);
    } else {
      add_spec_to_listbox(dlg, requests + i);
    }
  }

  free(requests);

  init_listbox_selection(dlg);
}

static void free_listbox_spec(HWND listbox, int selection) {
  FWDRequestSpec FAR * spec = (FWDRequestSpec FAR *)
    SendMessage(listbox, LB_GETITEMDATA, selection, 0);

  if (spec != NULL) {
    free(spec);
  }
}

static void free_all_listbox_specs(HWND dlg) {
  HWND listbox = GetDlgItem(dlg, IDC_SSHFWDLIST);
  int i;

  for (i = SendMessage(listbox, LB_GETCOUNT, 0, 0) - 1; i >= 0; i--) {
    free_listbox_spec(listbox, i);
  }
}

static BOOL end_fwd_dlg(PTInstVar pvar, HWND dlg) {
  char buf[1024];
  HWND listbox = GetDlgItem(dlg, IDC_SSHFWDLIST);
  int num_items = SendMessage(listbox, LB_GETCOUNT, 0, 0);
  BOOL X_enabled = IsDlgButtonChecked(dlg, IDC_SSHFWDX11);
  int num_specs = X_enabled ? 1 : 0;
  FWDRequestSpec FAR * specs = (FWDRequestSpec FAR *)malloc(sizeof(FWDRequestSpec)*(n
um_specs + num_items));
  int i;
  int num_unspecified_forwardings = 0;

  for (i = 0; i < num_items; i++) {
    FWDRequestSpec FAR * spec = (FWDRequestSpec FAR *)
      SendMessage(listbox, LB_GETITEMDATA, i, 0);

    if (spec != NULL) {
      specs[num_specs] = *spec;
      num_specs++;
    }
  }

  if (X_enabled) {
    make_X_forwarding_spec(specs);
  }

  qsort(specs, num_specs, sizeof(FWDRequestSpec), FWD_compare_specs);

  for (i = 0; i < num_specs; i++) {
    if (i < num_specs - 1 && FWD_compare_specs(specs + i, specs + i + 1) == 0) {
```



```
      switch (specs[i].type) {
      case FWD_REMOTE_TO_LOCAL:
        _snprintf(buf, sizeof(buf),
          "リモートサーバの同じポートを利用する転送設定を複数指定することはできません
.(ポート %d)",
          specs[i].from_port);
        break;
      case FWD_LOCAL_TO_REMOTE:
        _snprintf(buf, sizeof(buf),
          "同じローカルポートを利用する転送設定を複数指定することはできません.(ポー
ト %d)",
          specs[i].from_port);
        break;
      }
      buf[sizeof(buf) - 1] = 0;
      notify_nonfatal_error(pvar, buf);

      free(specs);
      return FALSE;
    }

    if (specs[i].type != FWD_LOCAL_TO_REMOTE
      && !FWD_can_server_listen_for(pvar, specs + i)) {
      num_unspecified_forwardings++;
    }
  }

  if (num_unspecified_forwardings > 0) {
    int buf_count;

//    if (num_unspecified_forwardings > 1) {
      strncpy(buf, "次のポート転送設定はSSHセッション開始前に指定されたものではありま
せん:\n\n", sizeof(buf));
//    } else {
//      strncpy(buf, "The following forwarding was not specified when this SSH sessio
n began:\n\n", sizeof(buf));
//    }
    buf_count = strlen(buf);

    for (i = 0; i < num_specs; i++) {
      if (specs[i].type != FWD_LOCAL_TO_REMOTE
        && !FWD_can_server_listen_for(pvar, specs + i)) {
        char buf2[1024];

        get_spec_string(specs + i, buf2, sizeof(buf2));

        if (buf_count < sizeof(buf)) {
          strncpy(buf + buf_count, buf2, sizeof(buf) - buf_count);
          buf_count += strlen(buf + buf_count);
        }
        if (buf_count < sizeof(buf)) {
          buf[buf_count] = '\n';
          buf_count++;
        }
      }
    }

    if (buf_count < sizeof(buf)) {
//      if (num_unspecified_forwardings > 1) {
        strncpy(buf + buf_count,
                "\nSSHプロトコル上の制限のため，以上の転送設定は開始されている現在
のSSHセッションでは有効になりません.\n"
                "設定を保存後、次回のSSHセッション以降で利用できます.",
          sizeof(buf) - buf_count);
//      } else {
//        strncpy(buf + buf_count,
```

```
//          "\nDue to a limitation of the SSH protocol, this forwarding will not work
 in the current SSH session.\n"
//          "If you save these settings and start a new SSH session, the forwarding s
hould work.",
//          sizeof(buf) - buf_count);
//      }
    }
    buf[sizeof(buf) - 1] = 0;

    notify_nonfatal_error(pvar, buf);
  }

  FWD_set_request_specs(pvar, specs, num_specs);
  FWDUI_save_settings(pvar);
  free_all_listbox_specs(dlg);
  EndDialog(dlg, 1);

  free(specs);

  return TRUE;
}

static void fill_service_names(HWND dlg, WORD item) {
  HWND cbox = GetDlgItem(dlg, item);
  int i;

  for (i = 0; i < NUM_ELEM(service_DB); i++) {
    SendMessage(cbox, CB_ADDSTRING, 0, (LPARAM)service_DB[i].name);
  }
}

static void shift_over_input(HWND dlg, int type, WORD rtl_item, WORD ltr_item) {
  HWND shift_from;
  HWND shift_to;

  if (type == FWD_REMOTE_TO_LOCAL) {
    shift_from = GetDlgItem(dlg, ltr_item);
    shift_to = GetDlgItem(dlg, rtl_item);
  } else {
    shift_from = GetDlgItem(dlg, rtl_item);
    shift_to = GetDlgItem(dlg, ltr_item);
  }

  EnableWindow(shift_from, FALSE);
  EnableWindow(shift_to, TRUE);

  if (GetWindowTextLength(shift_to) == 0) {
    char buf[128];

    GetWindowText(shift_from, buf, sizeof(buf));
    buf[sizeof(buf) - 1] = 0;
    SetWindowText(shift_to, buf);
    SetWindowText(shift_from, "");
  }
}

static void set_dir_options_status(HWND dlg) {
  int type = IsDlgButtonChecked(dlg, IDC_SSHFWDREMOTETOLOCAL)
    ? FWD_REMOTE_TO_LOCAL : FWD_LOCAL_TO_REMOTE;

  shift_over_input(dlg, type, IDC_SSHRTLFROMPORT, IDC_SSHLTRFROMPORT);
  shift_over_input(dlg, type, IDC_SSHRTLTOHOST, IDC_SSHLTRTOHOST);
  shift_over_input(dlg, type, IDC_SSHRTLTOPORT, IDC_SSHLTRTOPORT);
}

static void setup_edit_controls(HWND dlg, FWDRequestSpec FAR * spec,
```



```
  WORD radio_item, WORD from_port_item, WORD to_host_item, WORD to_port_item) {
  CheckDlgButton(dlg, radio_item, TRUE);
  SetFocus(GetDlgItem(dlg, radio_item));
  SetDlgItemText(dlg, from_port_item, spec->from_port_name);
  SetDlgItemText(dlg, to_port_item, spec->to_port_name);
  if (strcmp(spec->to_host, "localhost") != 0) {
    SetDlgItemText(dlg, to_host_item, spec->to_host);
  }

  set_dir_options_status(dlg);
}

static void init_fwd_edit_dlg(FWDRequestSpec FAR * spec, HWND dlg) {
  switch (spec->type) {
  case FWD_REMOTE_TO_LOCAL:
    setup_edit_controls(dlg, spec, IDC_SSHFWDREMOTETOLOCAL,
      IDC_SSHRTLFROMPORT, IDC_SSHRTLTOHOST, IDC_SSHRTLTOPORT);
    break;
  case FWD_LOCAL_TO_REMOTE:
    setup_edit_controls(dlg, spec, IDC_SSHFWDLOCALTOREMOTE,
      IDC_SSHLTRFROMPORT, IDC_SSHLTRTOHOST, IDC_SSHLTRTOPORT);
    break;
  }

  fill_service_names(dlg, IDC_SSHRTLFROMPORT);
  fill_service_names(dlg, IDC_SSHLTRFROMPORT);
  fill_service_names(dlg, IDC_SSHRTLTOPORT);
  fill_service_names(dlg, IDC_SSHLTRTOPORT);
}

static void grab_control_text(HWND dlg, int type, WORD rtl_item, WORD ltr_item,
                              char FAR * buf, int bufsize) {
  GetDlgItemText(dlg, type == FWD_REMOTE_TO_LOCAL ? rtl_item : ltr_item,
    buf, bufsize);
  buf[bufsize - 1] = 0;
}

static BOOL end_fwd_edit_dlg(PTInstVar pvar, FWDRequestSpec FAR * spec, HWND dlg) {
  FWDRequestSpec new_spec;
  int type = IsDlgButtonChecked(dlg, IDC_SSHFWDREMOTETOLOCAL)
    ? FWD_REMOTE_TO_LOCAL : FWD_LOCAL_TO_REMOTE;
  char buf[1024];

  new_spec.type = type;
  grab_control_text(dlg, type, IDC_SSHRTLFROMPORT, IDC_SSHLTRFROMPORT,
    new_spec.from_port_name, sizeof(new_spec.from_port_name));
  grab_control_text(dlg, type, IDC_SSHRTLTOHOST, IDC_SSHLTRTOHOST,
    new_spec.to_host, sizeof(new_spec.to_host));
  if (new_spec.to_host[0] == 0) {
    strcpy(new_spec.to_host, "localhost");
  }
  grab_control_text(dlg, type, IDC_SSHRTLTOPORT, IDC_SSHLTRTOPORT,
    new_spec.to_port_name, sizeof(new_spec.to_port_name));

  new_spec.from_port = parse_port_from_buf(new_spec.from_port_name);
  if (new_spec.from_port < 0) {
    _snprintf(buf, sizeof(buf), "ポート \"%s\" は有効なポート番号ではありません．\n"
      "一覧からポート名を選択するか，1～65535の数値を入力してください．",
      new_spec.from_port_name);
    buf[sizeof(buf) - 1] = 0;
    notify_nonfatal_error(pvar, buf);
    return FALSE;
  }

  new_spec.to_port = parse_port_from_buf(new_spec.to_port_name);
  if (new_spec.to_port < 0) {
```

```
    _snprintf(buf, sizeof(buf), "ポート \"%s\" は有効なポート番号ではありません．\n"
      "一覧からポート名を選択するか，1～65535の数値を入力してください．",
      new_spec.to_port_name);
    buf[sizeof(buf) - 1] = 0;
    notify_nonfatal_error(pvar, buf);
    return FALSE;
  }

  *spec = new_spec;

  EndDialog(dlg, 1);
  return TRUE;
}

static BOOL CALLBACK fwd_edit_dlg_proc(HWND dlg, UINT msg, WPARAM wParam, LPARAM lPar
am) {
  FWDEditClosure FAR * closure;

  switch (msg) {
  case WM_INITDIALOG:
    closure = (FWDEditClosure FAR *)lParam;
    SetWindowLong(dlg, DWL_USER, lParam);

    init_fwd_edit_dlg(closure->spec, dlg);
    return FALSE; /* because we set the focus */

  case WM_COMMAND:
    closure = (FWDEditClosure FAR *)GetWindowLong(dlg, DWL_USER);

    switch (LOWORD(wParam)) {
    case IDOK:
      return end_fwd_edit_dlg(closure->pvar, closure->spec, dlg);

    case IDCANCEL:
      EndDialog(dlg, 0);
      return TRUE;

    case IDC_SSHFWDLOCALTOREMOTE:
    case IDC_SSHFWDREMOTETOLOCAL:
      set_dir_options_status(dlg);
      return TRUE;

    default:
      return FALSE;
    }

  default:
    return FALSE;
  }
}

static void add_forwarding_entry(PTInstVar pvar, HWND dlg) {
  FWDRequestSpec new_spec;
  int result;
  FWDEditClosure closure = { &new_spec, pvar };

  new_spec.type = FWD_LOCAL_TO_REMOTE;
  new_spec.from_port_name[0] = 0;
  new_spec.to_host[0] = 0;
  new_spec.to_port_name[0] = 0;

  result = DialogBoxParam(hInst, MAKEINTRESOURCE(IDD_SSHFWDEDIT),
    dlg, fwd_edit_dlg_proc, (LPARAM)&closure);

  if (result == -1) {
    notify_nonfatal_error(pvar, "ポート転送編集ダイアログボックスを表示できませんでし
```



```
た．");
  } else if (result) {
    int index = add_spec_to_listbox(dlg, &new_spec);

    if (index >= 0) {
      SendMessage(GetDlgItem(dlg, IDC_SSHFWDLIST), LB_SETCURSEL, index, 0);
    }
    update_listbox_selection(dlg);
  }
}

static void edit_forwarding_entry(PTInstVar pvar, HWND dlg) {
  HWND listbox = GetDlgItem(dlg, IDC_SSHFWDLIST);
  int cursel = SendMessage(listbox, LB_GETCURSEL, 0, 0);

  if (cursel >= 0) {
    FWDRequestSpec FAR * spec = (FWDRequestSpec FAR *)
      SendMessage(listbox, LB_GETITEMDATA, cursel, 0);

    if (spec != NULL) {
      FWDEditClosure closure = { spec, pvar };

      int result = DialogBoxParam(hInst, MAKEINTRESOURCE(IDD_SSHFWDEDIT),
        dlg, fwd_edit_dlg_proc, (LPARAM)&closure);

      if (result == -1) {
        notify_nonfatal_error(pvar, "ポート転送編集ダイアログボックスを表示できません
でした．");
      } else if (result) {
        SendMessage(listbox, LB_DELETESTRING, cursel, 0);

        cursel = add_spec_to_listbox(dlg, spec);
        free(spec);
        if (cursel >= 0) {
          SendMessage(GetDlgItem(dlg, IDC_SSHFWDLIST), LB_SETCURSEL, cursel, 0);
        }
        update_listbox_selection(dlg);
      }
    }
  }
}

static void remove_forwarding_entry(HWND dlg) {
  HWND listbox = GetDlgItem(dlg, IDC_SSHFWDLIST);
  int cursel = SendMessage(listbox, LB_GETCURSEL, 0, 0);

  if (cursel >= 0) {
    free_listbox_spec(listbox, cursel);
    SendMessage(listbox, LB_DELETESTRING, cursel, 0);
    init_listbox_selection(dlg);
  }
}

static BOOL CALLBACK fwd_dlg_proc(HWND dlg, UINT msg, WPARAM wParam, LPARAM lParam) {
  PTInstVar pvar;

  switch (msg) {
  case WM_INITDIALOG:
    pvar = (PTInstVar)lParam;
    SetWindowLong(dlg, DWL_USER, lParam);

    init_fwd_dlg(pvar, dlg);
    return TRUE; /* because we do not set the focus */

  case WM_COMMAND:
    pvar = (PTInstVar)GetWindowLong(dlg, DWL_USER);
```

```
    switch (LOWORD(wParam)) {
    case IDOK:
      return end_fwd_dlg(pvar, dlg);

    case IDCANCEL:
      free_all_listbox_specs(dlg);
      EndDialog(dlg, 0);
      return TRUE;

    case IDC_ADD:
      add_forwarding_entry(pvar, dlg);
      return TRUE;

    case IDC_EDIT:
      edit_forwarding_entry(pvar, dlg);
      return TRUE;

    case IDC_REMOVE:
      remove_forwarding_entry(dlg);
      return TRUE;

    case IDC_SSHFWDLIST:
      update_listbox_selection(dlg);
      return TRUE;

    default:
      return FALSE;
    }

  default:
    return FALSE;
  }
}

void FWDUI_do_forwarding_dialog(PTInstVar pvar) {
  HWND cur_active = GetActiveWindow();

  if (DialogBoxParam(hInst, MAKEINTRESOURCE(IDD_SSHFWDSETUP),
    cur_active != NULL ? cur_active : pvar->NotificationWindow, fwd_dlg_proc, (LPARAM
)pvar) == -1) {
    notify_nonfatal_error(pvar, "ポート転送設定ダイアログボックスを表示できませんでし
た．");
  }
}
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
This code is copyright (C) 1998-1999 Robert O'Callahan.
See LICENSE.TXT for the license.
*/

#ifndef __FWDUI_H
#define __FWDUI_H

#include "ttxssh.h"

void FWDUI_init(PTInstVar pvar);
void FWDUI_load_settings(PTInstVar pvar);
void FWDUI_open(PTInstVar pvar);
void FWDUI_do_forwarding_dialog(PTInstVar pvar);
void FWDUI_end(PTInstVar pvar);

#ifdef __cplusplus
extern "C" {
#endif
void FWDUI_save_settings(PTInstVar pvar);
#ifdef __cplusplus
}
#endif


#endif
```

```
#include <winsock2.h>
#include "ftpwrap.h"


void OnAccept(HWND hwnd, SOCKET socket, int nErrorCode)
{
        SOCKADDR_IN     SockAddr;
        PREQUEST        lpReq;
        SOCKET          peerSocket;
        int                     nRet;
        int                     nLen;

        //
        // accept the new socket descriptor
        //
        nLen = sizeof(SOCKADDR_IN);
        peerSocket = accept(socket,
                                        (LPSOCKADDR)&SockAddr,
                                            &nLen);
        if (peerSocket == SOCKET_ERROR)
        {
                nRet = WSAGetLastError();
                if (nRet != WSAEWOULDBLOCK)
                {
                        //
                        // Just log the error and return
                        //
                        _RPTF0(_CRT_WARN, "accept() error\n");
                        return;
                }
        }

        //
        // Make sure we get async notices for this socket
        //
    nRet = WSAAsyncSelect(peerSocket,
                                                hwnd,
                                                WM_NETEVENT,
                                                FD_READ | FD_WRITE | FD_CLOSE);
        if (peerSocket == SOCKET_ERROR)
        {
                nRet = WSAGetLastError();
                //
                // Just log the error and return
                //
                _RPTF0(_CRT_WARN, "accept/WSAAsyncSelect error\n");
                return;
        }

        //
        // Add the connection to the linked list
        //
        lpReq = FTPGetNewRequest();
        if (lpReq == NULL)
        {
                // We're probably out of memory
                shutdown( peerSocket, SD_BOTH );
                closesocket(peerSocket);
                return;
        }

        lpReq->socket = peerSocket;
        lpReq->SockAddr = SockAddr;
        lpReq->state = ST_GREETING;

        _RPTF2(_CRT_WARN, "Connection accepted on socket %d from: %s\n",
```



```
                        peerSocket,
                        inet_ntoa(SockAddr.sin_addr));

        //
        // greeting msg
        //
        ftp_msgf(lpReq, 220, "FTP wrapper version " VERSION " ready.\n"
                "You can connect to a SERVER by 'user' command:\n"
                "    USER username@SERVER");
        return;
}


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

void OnRead(SOCKET socket, int nErrorCode) {
        PREQUEST                lpReq;
        int                             nRet;

        //
        // Find this socket in the linked list
        //
        lpReq = FTPGetRequest(socket);
        if (lpReq == NULL)
        {
                //
                // Not in the list. Log the error,
                // read the data to clear the buffers
                // and close the connection
                //
                nRet = 0;
                while(nRet != SOCKET_ERROR) {
                        char buf[1024];
                        nRet = recv(socket, buf, sizeof(buf)-1, 0);
                }
                shutdown( socket, SD_BOTH );
                closesocket(socket);
                _RPTF1(_CRT_WARN, "internal error. invalid read event occurred.\n"
                        "Connection closed on socket %d\n", socket );
                return;
        }

        //
        // Recv the data
        //
        nRet = recv(lpReq->socket, (PCHAR)(lpReq->RdBuff + lpReq->RdDataSize), FTPRDB
UFFSIZE - lpReq->RdDataSize, 0);
        if (nRet == SOCKET_ERROR)
        {
                if (WSAGetLastError() == WSAEWOULDBLOCK)
                        return;

                _RPTF1(_CRT_WARN, "recv() error.\nConnection closed on socket %d\n",
lpReq->socket);
                shutdown( lpReq->socket, SD_BOTH );
                closesocket(lpReq->socket);

                FTPDelRequest(lpReq);
                return;
        }

        //
        // Keep track of what has actually been sent
        //
        lpReq->RdDataSize += nRet;
```

```
        //
        // Parse the request
        //

        if( lpReq->state == ST_WAIT_CMD || lpReq->state == ST_CONNECT ) {
                ParseRecvData(lpReq);
        }

        return;
}

void ParseRecvData(PREQUEST lpReq) {
        unsigned int line_offset;
        int FoundNL = 0;

        //
        // check newline in the read buffer and if it finds, the command will be proc
essed
        //
        for( line_offset = 0; line_offset < lpReq->RdDataSize; line_offset++ ) {
                if( *(lpReq->RdBuff + line_offset) == '\n' ) {
                        FoundNL++;
                        break;
                }
        }

        if( ! FoundNL )
                return;

        if( line_offset > 0 &&  *(lpReq->RdBuff + line_offset - 1) == '\r' )
                line_offset--;

        if( line_offset > 0 ) {
                MyStrNCpy2( lpReq->ftp_cmdline_buf, (char *)lpReq->RdBuff, line_offse
t );
        } else {
                lpReq->ftp_cmdline_buf[0] = '\0';
        }

        if( lpReq->state == ST_WAIT_CMD )
                lpReq->state = ST_SEND_RESP;

        int retcode;
        retcode = ftp_get_command( lpReq );

        if( retcode ) { // 0 = QUIT cmd issued
                if(*(lpReq->RdBuff + line_offset) == '\r')
                        line_offset++;
                line_offset++;

                if( lpReq->state == ST_CONNECT && retcode == 1 ) { // retcode 2 = POR
T cmd was processed, don't need to forward data
                        SSH_channel_send(lpReq->pvar, lpReq->pvar->fwd_state.channels
[lpReq->localch].remote_num,
                                         lpReq->RdBuff, line_offset);
                }

                lpReq->RdDataSize -= line_offset;
                memmove( lpReq->RdBuff, lpReq->RdBuff + line_offset, lpReq->RdDataSiz
e );
        }
        return;
}
```



```
////////////////////////////////////////////////////////

void OnWrite(SOCKET socket, int nErrorCode) {
        PREQUEST lpReq;
        int nRet;

        lpReq = FTPGetRequest(socket);
        if (lpReq == NULL) {
                //
                // Not in our list!?
                //
                nRet = 0;
                while(nRet != SOCKET_ERROR) {
                        char buf[1024];
                        nRet = recv(socket, buf, sizeof(buf)-1, 0);
                }
                _RPTF1(_CRT_WARN, "internal error. invalid write event occurred.\n"
                        "Connection closed on socket %d\n", socket);
                shutdown( socket, SD_BOTH );
                closesocket(socket);

                return;
        }

        //
        // if write buffer is empty, do nothing
        //
        if (lpReq->WrDataSize == 0)
                return;

        //
        // Continue sending data
        //
        WriteData(lpReq);
        return;
}


int WriteData(PREQUEST lpReq) {
        UINT uBytes;
        long nBytesSent;
        unsigned char *WrBufPtr = lpReq->WrBuff;

        //
        // Send as much of the buffer as we can
        //
        while(1) {

                if( lpReq->WrDataSize > FTPWRITESIZE )
                        uBytes = FTPWRITESIZE;
                else
                        uBytes = lpReq->WrDataSize;

                //
                // Send this buffer to the client
                //
                nBytesSent = send(lpReq->socket,
                                                  (const char *)WrBufPtr,
                                                  uBytes,
                                                  0);
                if (nBytesSent == SOCKET_ERROR)
                {
                        if (WSAGetLastError() != WSAEWOULDBLOCK)
                        {
                                char szBuf[256];
```

```
                                LoadString(GetWindowInstance(g_hWnd),
                                                WSAGetLastError(),
                                                szBuf,
                                                sizeof(szBuf));
                                _RPTF2(_CRT_WARN, "send() error. (%s)\n"
                                        "Connection closed on socket %d\n", szBuf, lp
Req->socket);
                                shutdown( lpReq->socket, SD_BOTH );
                                closesocket(lpReq->socket);

                                FTPDelRequest(lpReq);
                                return -1;
                        }
                        nBytesSent = 0;
                }

                //
                // Keep track of what has actually been sent
                //
                lpReq->WrDataSize -= nBytesSent;
                WrBufPtr += nBytesSent;


                //
                // Is sent buffer empty ?
                //
                if (lpReq->WrDataSize == 0)
                        break;

                //
                // Are the protocol stack buffers full?
                //
                if (nBytesSent < (int)uBytes)
                {
                        memmove( lpReq->WrBuff, WrBufPtr, lpReq->WrDataSize );
                        // We'll have to finish later
                        return 1;
                }
        }

        //
        // Are we finished?
        //
        if( lpReq->state == ST_GREETING ) {
                lpReq->state = ST_WAIT_CMD;
        } else if( lpReq->state == ST_SEND_RESP ) {
                lpReq->state = ST_WAIT_CMD;
        } else if (lpReq->state == ST_QUIT) {
                _RPTF1(_CRT_WARN, "Connection closed on socket %d\n", lpReq->socket);
                shutdown( lpReq->socket, SD_BOTH );
                closesocket(lpReq->socket);

                FTPDelRequest(lpReq);
                return 2;
        }
        return 0;
}

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

void OnClose(SOCKET socket, int nErrorCode) {
        PREQUEST lpReq;

        //
        // Have we already deleted this entry?
        //
```



```
        lpReq = FTPGetRequest(socket);
        if (lpReq == NULL)
                return;

        //
        // It's still in our list
        // The client must have reset the connection.
        // Clean up.
        //
        _RPTF1(_CRT_WARN, "Connection closed on socket %d\n", lpReq->socket);
        shutdown( lpReq->socket, SD_BOTH );
        closesocket(lpReq->socket);

        if( lpReq->state == ST_CONNECTING2 || lpReq->state == ST_CONNECTING3 || lpReq
->state == ST_CONNECT ) {
                SSH_channel_input_eof(lpReq->pvar, lpReq->pvar->fwd_state.channels[lp
Req->localch].remote_num);
                SSH_channel_output_eof(lpReq->pvar, lpReq->pvar->fwd_state.channels[l
pReq->localch].remote_num);
                lpReq->pvar->fwd_state.channels[lpReq->localch].status |= FWD_CLOSED_
LOCAL_IN | FWD_CLOSED_LOCAL_OUT;

                // TODO: SSH close
        }
        FTPDelRequest(lpReq);
}

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

void OnSSHRead(PTInstVar pvar, int localch, unsigned char *data, unsigned int length)
 {
        PREQUEST lpReq;
        int datasize;

        lpReq = FTPGetRequestByPVar(pvar);
        if( lpReq == NULL )
                return;

        if( lpReq->state == ST_CONNECT ) {
//              _RPTF0(_CRT_WARN, "OnSSHRead: ST_CONNECT\n");

                datasize = min( FTPWRBUFFSIZE - lpReq->WrDataSize, length );
                memcpy(lpReq->WrBuff + lpReq->WrDataSize, data, datasize );
                length -= datasize;
                lpReq->WrDataSize += datasize;

                // normal write
                if( WriteData(lpReq) == 1 && length > 0 ) {
                        // Winsock buffer full & SSH buffer still remains

                        // switch to blocking I/O
                        unsigned long do_block = 0;
                        WSAAsyncSelect(lpReq->socket, pvar->cv.HWin, 0, 0);
                        ioctlsocket(lpReq->socket, FIONBIO, &do_block);
                        // send remaining write buffer
                        send(lpReq->socket, (char *)lpReq->WrBuff, lpReq->WrDataSize,
 0);
                        lpReq->WrDataSize = 0;

                        if( length > FTPWRBUFFSIZE )
                                send(lpReq->socket, (char *)data, length - FTPWRBUFFS
IZE, 0);

                        // turn on normal non blocking I/O
                        WSAAsyncSelect(lpReq->socket, pvar->cv.HWin,
                                WM_NETEVENT, FD_READ | FD_CLOSE | FD_WRITE);
```

```
                        // and buffers
                        memcpy(lpReq->WrBuff + lpReq->WrDataSize, data, length );
                        lpReq->WrDataSize = length;
                        WriteData(lpReq);
                }

        } else {
                datasize = min( SSHRDBUFFSIZE - lpReq->SSHRdDataSize, length );
                memcpy(lpReq->SSHRdBuff + lpReq->SSHRdDataSize, data, datasize );
                length -= datasize;
                lpReq->SSHRdDataSize += datasize;

                *(lpReq->SSHRdBuff + lpReq->SSHRdDataSize) = '\0';
//              _RPTF1(_CRT_WARN, "from sv: %s", lpReq->SSHRdBuff);

                // manipulate server response strings
                int retcode;
                retcode = ftp_get_resp( lpReq );
//              _RPTF1(_CRT_WARN, "resp: %s", lpReq->ftp_resp_str);
                if( retcode == 1 ) {
                        if( lpReq->state == ST_CONNECTING2 ) {
                                if( lpReq->UserName[0] == NULL ) {
                                        // USER cmd without username
                                        _RPTF0(_CRT_WARN, "OnSSHRead: sending server
response to the client. (next cmd: USER)\n");
                                        ftp_msgf( lpReq, lpReq->ftp_resp_code, "%s",
lpReq->ftp_resp_str );
                                        lpReq->state = ST_CONNECT;
                                } else {
                                        unsigned char new_buf[1024];
                                        _snprintf( (char *)new_buf, sizeof( new_buf )
, "USER %s\r\n", lpReq->UserName );
                                        // send username to server
                                        _RPTF0(_CRT_WARN, "OnSSHRead: sending USER cm
d to the server.\n");
                                        SSH_channel_send(lpReq->pvar, lpReq->pvar->fw
d_state.channels[localch].remote_num,
                                                new_buf, strlen((char *)new_buf));
                                        lpReq->state = ST_CONNECTING3;
                                }
                        } else { // should be ST_CONNECTING3
                                _RPTF0(_CRT_WARN, "OnSSHRead: sending server response
 to the client. (next cmd: PASS)\n");
                                ftp_msgf( lpReq, lpReq->ftp_resp_code, "%s", lpReq->f
tp_resp_str );
                                lpReq->ftp_resp_str[0] = '\0';
                                lpReq->state = ST_CONNECT;
                        }
                } else if( retcode == 2 ) {
                        ftp_msgf( lpReq, 530, "Proxy failed to connect. (server retur
ns invalid response)\r\n%s",
                                lpReq->ftp_resp_str );
                        // TODO: shutdown channel etc.
                        lpReq->state = ST_WAIT_CMD;
                }

                if( length > 0 ) {
                        // Winsock buffer full & SSH buffer still remains

                        // switch to blocking I/O
                        unsigned long do_block = 0;
                        WSAAsyncSelect(lpReq->socket, pvar->cv.HWin, 0, 0);
                        ioctlsocket(lpReq->socket, FIONBIO, &do_block);
                        // send remaining write buffer
                        send(lpReq->socket, (char *)lpReq->WrBuff, lpReq->WrDataSize,
```



```
 0);
                        lpReq->WrDataSize = 0;
                        // send remaining SSH read buffer
                        send(lpReq->socket, (char *)lpReq->SSHRdBuff, lpReq->SSHRdDat
aSize, 0);
                        lpReq->SSHRdDataSize = 0;

                        if( length > FTPWRBUFFSIZE )
                                send(lpReq->socket, (char *)data, length - FTPWRBUFFS
IZE, 0);

                        // turn on normal non blocking I/O
                        WSAAsyncSelect(lpReq->socket, pvar->cv.HWin,
                                WM_NETEVENT, FD_READ | FD_CLOSE | FD_WRITE);

                        // and buffers
                        memcpy(lpReq->WrBuff + lpReq->WrDataSize, data, length );
                        lpReq->WrDataSize = length;
                        WriteData(lpReq);
                }

        }
}
```

```
// stdafx.cpp : 標準インクルードファイルを含むソース ファイル
//              apopwrap.pch 生成されるプリコンパイル済ヘッダー
//              stdafx.obj 生成されるプリコンパイル済タイプ情報

#include "stdafx.h"

// TODO: STDAFX.H に含まれていて、このファイルに記述されていない
// ヘッダーファイルを追加してください。
```

```
// stdafx.h : 標準のシステム インクルード ファイル、
//            または参照回数が多く、かつあまり変更されない
//            プロジェクト専用のインクルード ファイルを記述します。
//

#if !defined(AFX_STDAFX_H__A9DB83DB_A9FD_11D0_BFD1_444553540000__INCLUDED_)
#define AFX_STDAFX_H__A9DB83DB_A9FD_11D0_BFD1_444553540000__INCLUDED_

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

#define WIN32_LEAN_AND_MEAN             // Windows ヘッダーから殆ど使用されないスタッ
フを除外します

#include <windows.h>


// TODO: プログラムで必要なヘッダー参照を追加してください。

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif // !defined(AFX_STDAFX_H__A9DB83DB_A9FD_11D0_BFD1_444553540000__INCLUDED_)
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

#include "ttxssh.h"
#include "util.h"

#include <io.h>
#include <fcntl.h>
#include <stdlib.h>
#include <errno.h>

#include "resource.h"
#include "keyfiles.h"

#define AUTH_START_USER_AUTH_ON_ERROR_END 1

#define MAX_AUTH_CONTROL IDC_SSHUSETIS

static void destroy_malloced_string(char FAR * FAR * str) {
  if (*str != NULL) {
    memset(*str, 0, strlen(*str));
    free(*str);
    *str = NULL;
  }
}

static int auth_types_to_control_IDs[] = {
  -1, IDC_SSHUSERHOSTS, IDC_SSHUSERSA, IDC_SSHUSEPASSWORD,
  IDC_SSHUSERHOSTS, IDC_SSHUSETIS, -1
};

static LRESULT CALLBACK password_wnd_proc(HWND control, UINT msg, WPARAM wParam, LPAR
AM lParam) {
  switch (msg) {
  case WM_CHAR:
    if ((GetKeyState(VK_CONTROL) & 0x8000) != 0) {
```



```
      char chars[] = { (char)wParam, 0 };

      SendMessage(control, EM_REPLACESEL, (WPARAM)TRUE, (LPARAM)(char FAR *)chars);
      return 0;
    }
  }

  return CallWindowProc((WNDPROC)GetWindowLong(control, GWL_USERDATA),
    control, msg, wParam, lParam);
}

static void init_password_control(HWND dlg) {
  HWND passwordControl = GetDlgItem(dlg, IDC_SSHPASSWORD);

  SetWindowLong(passwordControl, GWL_USERDATA,
    SetWindowLong(passwordControl, GWL_WNDPROC, (LONG)password_wnd_proc));

  SetFocus(passwordControl);
}

static void set_auth_options_status(HWND dlg, int controlID) {
  BOOL RSA_enabled = controlID == IDC_SSHUSERSA;
  BOOL rhosts_enabled = controlID == IDC_SSHUSERHOSTS;
  BOOL TIS_enabled = controlID == IDC_SSHUSETIS;
  int i;

  CheckRadioButton(dlg, IDC_SSHUSEPASSWORD, MAX_AUTH_CONTROL,
    controlID);

  EnableWindow(GetDlgItem(dlg, IDC_SSHPASSWORDCAPTION), !TIS_enabled);
  EnableWindow(GetDlgItem(dlg, IDC_SSHPASSWORD), !TIS_enabled);

  for (i = IDC_CHOOSERSAFILE; i <= IDC_RSAFILENAME; i++) {
    EnableWindow(GetDlgItem(dlg, i), RSA_enabled);
  }

  for (i = IDC_LOCALUSERNAMELABEL; i <= IDC_HOSTRSAFILENAME; i++) {
    EnableWindow(GetDlgItem(dlg, i), rhosts_enabled);
  }
}

# define LOGGING_MESSAGE "にログインしています"

static void init_auth_machine_banner(PTInstVar pvar, HWND dlg) {
  char buf[1024];

  strncpy(buf, SSH_get_host_name(pvar), sizeof(buf));
  if (strlen(SSH_get_host_name(pvar)) + strlen(LOGGING_MESSAGE) < sizeof(buf) - 2) {
    strcat(buf, LOGGING_MESSAGE);
  }
  SetDlgItemText(dlg, IDC_SSHAUTHBANNER, buf);
}

static void update_server_supported_types(PTInstVar pvar, HWND dlg) {
  int supported_methods = pvar->auth_state.supported_types;
  int cur_control = -1;
  int control;
  HWND focus = GetFocus();

  if (supported_methods == 0) {
    return;
  }

  for (control = IDC_SSHUSEPASSWORD; control <= MAX_AUTH_CONTROL; control++) {
    BOOL enabled = FALSE;
    int method;
```

```
    HWND item = GetDlgItem(dlg, control);

    if (item != NULL) {
      for (method = 0; method <= SSH_AUTH_MAX; method++) {
        if (auth_types_to_control_IDs[method] == control
          && (supported_methods & (1 << method)) != 0) {
          enabled = TRUE;
        }
      }

      EnableWindow(item, enabled);

      if (IsDlgButtonChecked(dlg, control)) {
        cur_control = control;
      }
    }
  }

  if (cur_control >= 0) {
    if (!IsWindowEnabled(GetDlgItem(dlg, cur_control))) {
      do {
        cur_control++;
        if (cur_control > MAX_AUTH_CONTROL) {
          cur_control = IDC_SSHUSEPASSWORD;
        }
      } while (!IsWindowEnabled(GetDlgItem(dlg, cur_control)));

      set_auth_options_status(dlg, cur_control);

      if (focus != NULL && !IsWindowEnabled(focus)) {
        SetFocus(GetDlgItem(dlg, cur_control));
      }
    }
  }
}

static void init_auth_dlg(PTInstVar pvar, HWND dlg) {
  int default_method = pvar->session_settings.DefaultAuthMethod;

  init_auth_machine_banner(pvar, dlg);
  init_password_control(dlg);

  if (pvar->auth_state.failed_method != SSH_AUTH_NONE) {
    /* must be retrying a failed attempt */
    SetDlgItemText(dlg, IDC_SSHAUTHBANNER2, "認証は失敗しました．再試行して下さい．")
;
    SetWindowText(dlg, "SSH認証の再試行中");
    default_method = pvar->auth_state.failed_method;
  }

  set_auth_options_status(dlg, auth_types_to_control_IDs[default_method]);

  if (default_method == SSH_AUTH_TIS) {
    /* we disabled the password control, so fix the focus */
    SetFocus(GetDlgItem(dlg, IDC_SSHUSETIS));
  }

  if (pvar->auth_state.user != NULL) {
    SetDlgItemText(dlg, IDC_SSHUSERNAME, pvar->auth_state.user);
    EnableWindow(GetDlgItem(dlg, IDC_SSHUSERNAME), FALSE);
    EnableWindow(GetDlgItem(dlg, IDC_SSHUSERNAMELABEL), FALSE);
  } else if (pvar->session_settings.DefaultUserName[0] != 0) {
    SetDlgItemText(dlg, IDC_SSHUSERNAME, pvar->session_settings.DefaultUserName);
  } else {
    SetFocus(GetDlgItem(dlg, IDC_SSHUSERNAME));
  }
```



```
  SetDlgItemText(dlg, IDC_RSAFILENAME, pvar->session_settings.DefaultRSAPrivateKeyFil
e);
  SetDlgItemText(dlg, IDC_HOSTRSAFILENAME, pvar->session_settings.DefaultRhostsHostPr
ivateKeyFile);
  SetDlgItemText(dlg, IDC_LOCALUSERNAME, pvar->session_settings.DefaultRhostsLocalUse
rName);

  update_server_supported_types(pvar, dlg);
}

static char FAR * alloc_control_text(HWND ctl) {
  int len = GetWindowTextLength(ctl);
  char * result = (char *)malloc(len + 1);

  if (result != NULL) {
    GetWindowText(ctl, result, len + 1);
    result[len] = 0;
  }

  return result;
}

static int get_key_file_name(HWND parent, char * title, char * def_filename, char FAR
 * buf, int bufsize) {
//static int get_key_file_name(HWND parent, char FAR * buf, int bufsize) {
#ifdef TERATERM32
  OPENFILENAME params;
//  char fullname_buf[2048] = "identity";
  char fullname_buf[2048] ;

  params.lStructSize = sizeof(OPENFILENAME);
  params.hwndOwner = parent;
  params.lpstrFilter = NULL;
  params.lpstrCustomFilter = NULL;
  params.nFilterIndex = 0;
  buf[0] = 0;
  strncpy(fullname_buf, def_filename, sizeof(fullname_buf));
  params.lpstrFile = fullname_buf;
  params.nMaxFile = sizeof(fullname_buf);
  params.lpstrFileTitle = NULL;
  params.lpstrInitialDir = NULL;
  params.lpstrTitle = title;
  params.Flags = OFN_FILEMUSTEXIST | OFN_PATHMUSTEXIST | OFN_HIDEREADONLY;
  params.lpstrDefExt = NULL;

  if (GetOpenFileName(&params) != 0) {
    copy_teraterm_dir_relative_path(buf, bufsize, fullname_buf);
    return 1;
  } else {
    return 0;
  }
#else
  return 0;
#endif
}

static void choose_RSA_key_file(HWND dlg) {
  char buf[1024];

  if (get_key_file_name(dlg, "RSA個人鍵ファイルの選択", "identity", buf, sizeof(buf))
) {
//  if (get_key_file_name(dlg, buf, sizeof(buf))) {
    SetDlgItemText(dlg, IDC_RSAFILENAME, buf);
  }
}
```

```
static void choose_host_RSA_key_file(HWND dlg) {
  char buf[1024];

  if (get_key_file_name(dlg, "RSAホスト鍵ファイルの選択", "ssh_host_key", buf, sizeof
(buf))) {
//  if (get_key_file_name(dlg, buf, sizeof(buf))) {
    SetDlgItemText(dlg, IDC_HOSTRSAFILENAME, buf);
  }
}

static BOOL end_auth_dlg(PTInstVar pvar, HWND dlg) {
  int method = SSH_AUTH_PASSWORD;
  char FAR * password = alloc_control_text(GetDlgItem(dlg, IDC_SSHPASSWORD));
  CRYPTKeyPair FAR * key_pair = NULL;

  if (IsDlgButtonChecked(dlg, IDC_SSHUSERSA)) {
    method = SSH_AUTH_RSA;
  } else if (IsDlgButtonChecked(dlg, IDC_SSHUSERHOSTS)) {
    if (GetWindowTextLength(GetDlgItem(dlg, IDC_HOSTRSAFILENAME)) > 0) {
      method = SSH_AUTH_RHOSTS_RSA;
    } else {
      method = SSH_AUTH_RHOSTS;
    }
  } else if (IsDlgButtonChecked(dlg, IDC_SSHUSETIS)) {
    method = SSH_AUTH_TIS;
  }

  if (method == SSH_AUTH_RSA || method == SSH_AUTH_RHOSTS_RSA) {
    char buf[2048];
    int file_ctl_ID = method == SSH_AUTH_RSA ? IDC_RSAFILENAME : IDC_HOSTRSAFILENAME;

    buf[0] = '\0';
    GetDlgItemText(dlg, file_ctl_ID, buf, sizeof(buf));
    if (buf[0] == 0) {
      notify_nonfatal_error(pvar, "RSA個人鍵のファイルを指定して下さい．");
      SetFocus(GetDlgItem(dlg, file_ctl_ID));
      destroy_malloced_string(&password);
      return FALSE;
    } else {
      BOOL invalid_passphrase = FALSE;

      key_pair = KEYFILES_read_private_key(pvar, buf, password,
        &invalid_passphrase, FALSE);

      if (key_pair == NULL) {
        if (invalid_passphrase) {
          HWND passwordCtl = GetDlgItem(dlg, IDC_SSHPASSWORD);

          SetFocus(passwordCtl);
          SendMessage(passwordCtl, EM_SETSEL, 0, -1);
        } else {
          SetFocus(GetDlgItem(dlg, file_ctl_ID));
        }
        destroy_malloced_string(&password);
        return FALSE;
      }
    }
  }

  /* from here on, we cannot fail, so just munge cur_cred in place */
  pvar->auth_state.cur_cred.method = method;
  pvar->auth_state.cur_cred.key_pair = key_pair;
  /* we can't change the user name once it's set. It may already have
     been sent to the server, and it can only be sent once. */
  if (pvar->auth_state.user == NULL) {
```



```
    pvar->auth_state.user = alloc_control_text(GetDlgItem(dlg, IDC_SSHUSERNAME));
  }
  if (method == SSH_AUTH_PASSWORD) {
    pvar->auth_state.cur_cred.password = password;
  } else {
    destroy_malloced_string(&password);
  }
  if (method == SSH_AUTH_RHOSTS || method == SSH_AUTH_RHOSTS_RSA) {
    if (pvar->session_settings.DefaultAuthMethod != SSH_AUTH_RHOSTS) {
      notify_nonfatal_error(pvar,
        "デフォルトの認証方式に設定されていないため(サーバへ接続しようとするポートが
特権ポートとは限らないために)、rhosts認証は失敗する可能性があります．\n"
        "TTSSHでrhosts認証を利用するには、接続する前にセットアップメニューから\"SSH認
証...\"を選択し、"
        "デフォルト認証方式にrhosts認証を設定して下さい．");
    }

    pvar->auth_state.cur_cred.rhosts_client_user =
      alloc_control_text(GetDlgItem(dlg, IDC_LOCALUSERNAME));
  }
  pvar->auth_state.auth_dialog = NULL;

  GetDlgItemText(dlg, IDC_RSAFILENAME, pvar->session_settings.DefaultRSAPrivateKeyFil
e, sizeof(pvar->session_settings.DefaultRSAPrivateKeyFile));
  GetDlgItemText(dlg, IDC_HOSTRSAFILENAME, pvar->session_settings.DefaultRhostsHostPr
ivateKeyFile, sizeof(pvar->session_settings.DefaultRhostsHostPrivateKeyFile));
  GetDlgItemText(dlg, IDC_LOCALUSERNAME, pvar->session_settings.DefaultRhostsLocalUse
rName, sizeof(pvar->session_settings.DefaultRhostsLocalUserName));

  SSH_notify_user_name(pvar);
  SSH_notify_cred(pvar);

  EndDialog(dlg, 1);
  return TRUE;
}

static BOOL CALLBACK auth_dlg_proc(HWND dlg, UINT msg, WPARAM wParam, LPARAM lParam)
{
  PTInstVar pvar;

  switch (msg) {
  case WM_INITDIALOG:
    pvar = (PTInstVar)lParam;
    pvar->auth_state.auth_dialog = dlg;
    SetWindowLong(dlg, DWL_USER, lParam);

    init_auth_dlg(pvar, dlg);
    return FALSE; /* because we set the focus */

  case WM_COMMAND:
    pvar = (PTInstVar)GetWindowLong(dlg, DWL_USER);

    switch (LOWORD(wParam)) {
    case IDOK:
      return end_auth_dlg(pvar, dlg);

    case IDCANCEL: /* kill the connection */
      pvar->auth_state.auth_dialog = NULL;
      notify_closed_connection(pvar);
      EndDialog(dlg, 0);
      return TRUE;

    case IDC_SSHUSEPASSWORD:
    case IDC_SSHUSERSA:
    case IDC_SSHUSERHOSTS:
    case IDC_SSHUSETIS:
```

```
      set_auth_options_status(dlg, LOWORD(wParam));
      return TRUE;

    case IDC_CHOOSERSAFILE:
      choose_RSA_key_file(dlg);
      return TRUE;

    case IDC_CHOOSEHOSTRSAFILE:
      choose_host_RSA_key_file(dlg);
      return TRUE;

    default:
      return FALSE;
    }

  default:
    return FALSE;
  }
}

char FAR * AUTH_get_user_name(PTInstVar pvar) {
  return pvar->auth_state.user;
}

int AUTH_set_supported_auth_types(PTInstVar pvar, int types) {
  char buf[1024];

  _snprintf(buf, sizeof(buf), "Server reports supported authentication method mask =
%d", types);
  buf[sizeof(buf) - 1] = 0;
  notify_verbose_message(pvar, buf, LOG_LEVEL_VERBOSE);

  types &= (1 << SSH_AUTH_PASSWORD) | (1 << SSH_AUTH_RSA)
    | (1 << SSH_AUTH_RHOSTS_RSA) | (1 << SSH_AUTH_RHOSTS)
    | (1 << SSH_AUTH_TIS);
  pvar->auth_state.supported_types = types;

  if (types == 0) {
    notify_fatal_error(pvar, "このサーバはTTSSHでサポートしているどの認証方法にも対応
していません．\n"
          "接続を切断します．");
    return 0;
  } else {
    if (pvar->auth_state.auth_dialog != NULL) {
      update_server_supported_types(pvar, pvar->auth_state.auth_dialog);
    }

    return 1;
  }
}

static void start_user_auth(PTInstVar pvar) {
  PostMessage(pvar->NotificationWindow, WM_COMMAND, (WPARAM)ID_SSHAUTH, (LPARAM)NULL)
;
  pvar->auth_state.cur_cred.method = SSH_AUTH_NONE;
}

static void try_default_auth(PTInstVar pvar) {
  if (pvar->session_settings.TryDefaultAuth) {
    switch (pvar->session_settings.DefaultAuthMethod) {
    case SSH_AUTH_RSA: {
      BOOL invalid_passphrase;
      char password[] = "";

      pvar->auth_state.cur_cred.key_pair
        = KEYFILES_read_private_key(pvar, pvar->session_settings.DefaultRSAPrivateKey
```



```
File,
            password, &invalid_passphrase, TRUE);
      if (pvar->auth_state.cur_cred.key_pair == NULL) {
        return;
      } else {
        pvar->auth_state.cur_cred.method = SSH_AUTH_RSA;
      }
      break;
    }

    case SSH_AUTH_RHOSTS:
      if (pvar->session_settings.DefaultRhostsHostPrivateKeyFile[0] != 0) {
        BOOL invalid_passphrase;
        char password[] = "";

        pvar->auth_state.cur_cred.key_pair
          = KEYFILES_read_private_key(pvar, pvar->session_settings.DefaultRhostsHostP
rivateKeyFile,
              password, &invalid_passphrase, TRUE);
        if (pvar->auth_state.cur_cred.key_pair == NULL) {
          return;
        } else {
          pvar->auth_state.cur_cred.method = SSH_AUTH_RHOSTS_RSA;
        }
      } else {
        pvar->auth_state.cur_cred.method = SSH_AUTH_RHOSTS;
      }

      pvar->auth_state.cur_cred.rhosts_client_user =
        _strdup(pvar->session_settings.DefaultRhostsLocalUserName);
      break;

    case SSH_AUTH_PASSWORD:
      pvar->auth_state.cur_cred.password = _strdup("");
      pvar->auth_state.cur_cred.method = SSH_AUTH_PASSWORD;
      break;

    case SSH_AUTH_TIS:
    default:
      return;
    }

    pvar->auth_state.user = _strdup(pvar->session_settings.DefaultUserName);
  }
}

void AUTH_notify_end_error(PTInstVar pvar) {
  if ((pvar->auth_state.flags & AUTH_START_USER_AUTH_ON_ERROR_END) != 0) {
    start_user_auth(pvar);
    pvar->auth_state.flags &= ~AUTH_START_USER_AUTH_ON_ERROR_END;
  }
}

void AUTH_advance_to_next_cred(PTInstVar pvar) {
  pvar->auth_state.failed_method = pvar->auth_state.cur_cred.method;

  if (pvar->auth_state.cur_cred.method == SSH_AUTH_NONE) {
    try_default_auth(pvar);

    if (pvar->auth_state.cur_cred.method == SSH_AUTH_NONE) {
      if (pvar->err_msg != NULL) {
        pvar->auth_state.flags |= AUTH_START_USER_AUTH_ON_ERROR_END;
      } else {
        start_user_auth(pvar);
      }
    }
```

```
  } else {
    start_user_auth(pvar);
  }
}

static void init_TIS_dlg(PTInstVar pvar, HWND dlg) {
  init_auth_machine_banner(pvar, dlg);
  init_password_control(dlg);

  if (pvar->auth_state.TIS_prompt != NULL) {
    if (strlen(pvar->auth_state.TIS_prompt) > 10000) {
      pvar->auth_state.TIS_prompt[10000] = 0;
    }
    SetDlgItemText(dlg, IDC_SSHAUTHBANNER2, pvar->auth_state.TIS_prompt);
    destroy_malloced_string(&pvar->auth_state.TIS_prompt);
  }
}

static BOOL end_TIS_dlg(PTInstVar pvar, HWND dlg) {
  char FAR * password = alloc_control_text(GetDlgItem(dlg, IDC_SSHPASSWORD));

  pvar->auth_state.cur_cred.method = SSH_AUTH_TIS;
  pvar->auth_state.cur_cred.password = password;
  pvar->auth_state.auth_dialog = NULL;

  SSH_notify_cred(pvar);

  EndDialog(dlg, 1);
  return TRUE;
}

static BOOL CALLBACK TIS_dlg_proc(HWND dlg, UINT msg, WPARAM wParam, LPARAM lParam) {
  PTInstVar pvar;

  switch (msg) {
  case WM_INITDIALOG:
    pvar = (PTInstVar)lParam;
    pvar->auth_state.auth_dialog = dlg;
    SetWindowLong(dlg, DWL_USER, lParam);

    init_TIS_dlg(pvar, dlg);
    return FALSE; /* because we set the focus */

  case WM_COMMAND:
    pvar = (PTInstVar)GetWindowLong(dlg, DWL_USER);

    switch (LOWORD(wParam)) {
    case IDOK:
      return end_TIS_dlg(pvar, dlg);

    case IDCANCEL: /* kill the connection */
      pvar->auth_state.auth_dialog = NULL;
      notify_closed_connection(pvar);
      EndDialog(dlg, 0);
      return TRUE;

    default:
      return FALSE;
    }

  default:
    return FALSE;
  }
}

void AUTH_do_cred_dialog(PTInstVar pvar) {
```



```
  if (pvar->auth_state.auth_dialog == NULL) {
    HWND cur_active = GetActiveWindow();
    DLGPROC dlg_proc;
    LPCTSTR dialog_template;

    switch (pvar->auth_state.mode) {
    case TIS_AUTH_MODE:
      dialog_template = MAKEINTRESOURCE(IDD_SSHTISAUTH);
      dlg_proc = TIS_dlg_proc;
      break;
    case GENERIC_AUTH_MODE:
    default:
      dialog_template = MAKEINTRESOURCE(IDD_SSHAUTH);
      dlg_proc = auth_dlg_proc;
    }

    if (!DialogBoxParam(hInst, dialog_template,
        cur_active != NULL ? cur_active : pvar->NotificationWindow,
        dlg_proc, (LPARAM)pvar) == -1) {
      notify_fatal_error(pvar, "認証ダイアログボックスを開けませんでした．\n"
        "接続を切断します．");
    }
  }
}

static void init_default_auth_dlg(PTInstVar pvar, HWND dlg) {
        switch (pvar->settings.DefaultAuthMethod) {
  case SSH_AUTH_RSA:
    CheckRadioButton(dlg, IDC_SSHUSEPASSWORD, MAX_AUTH_CONTROL, IDC_SSHUSERSA);
    break;
  case SSH_AUTH_RHOSTS:
  case SSH_AUTH_RHOSTS_RSA:
    CheckRadioButton(dlg, IDC_SSHUSEPASSWORD, MAX_AUTH_CONTROL, IDC_SSHUSERHOSTS);
    break;
  case SSH_AUTH_TIS:
    CheckRadioButton(dlg, IDC_SSHUSEPASSWORD, MAX_AUTH_CONTROL, IDC_SSHUSETIS);
    break;
  case SSH_AUTH_PASSWORD:
  default:
    CheckRadioButton(dlg, IDC_SSHUSEPASSWORD, MAX_AUTH_CONTROL, IDC_SSHUSEPASSWORD);
  }
  SetDlgItemText(dlg, IDC_SSHUSERNAME, pvar->settings.DefaultUserName);
  SetDlgItemText(dlg, IDC_RSAFILENAME, pvar->settings.DefaultRSAPrivateKeyFile);
  SetDlgItemText(dlg, IDC_HOSTRSAFILENAME, pvar->settings.DefaultRhostsHostPrivateKey
File);
  SetDlgItemText(dlg, IDC_LOCALUSERNAME, pvar->settings.DefaultRhostsLocalUserName);
}

static BOOL end_default_auth_dlg(PTInstVar pvar, HWND dlg) {
  if (IsDlgButtonChecked(dlg, IDC_SSHUSERSA)) {
    pvar->settings.DefaultAuthMethod = SSH_AUTH_RSA;
  } else if (IsDlgButtonChecked(dlg, IDC_SSHUSERHOSTS)) {
//    if (GetWindowTextLength(GetDlgItem(dlg, IDC_HOSTRSAFILENAME)) > 0) {
//      pvar->settings.DefaultAuthMethod = SSH_AUTH_RHOSTS_RSA;
//    } else {
      pvar->settings.DefaultAuthMethod = SSH_AUTH_RHOSTS;
//    }
  } else if (IsDlgButtonChecked(dlg, IDC_SSHUSETIS)) {
    pvar->settings.DefaultAuthMethod = SSH_AUTH_TIS;
  } else {
    pvar->settings.DefaultAuthMethod = SSH_AUTH_PASSWORD;
  }

  GetDlgItemText(dlg, IDC_SSHUSERNAME, pvar->settings.DefaultUserName, sizeof(pvar->s
ettings.DefaultUserName));
  GetDlgItemText(dlg, IDC_RSAFILENAME, pvar->settings.DefaultRSAPrivateKeyFile, sizeo
```

```
f(pvar->settings.DefaultRSAPrivateKeyFile));
  GetDlgItemText(dlg, IDC_HOSTRSAFILENAME, pvar->settings.DefaultRhostsHostPrivateKey
File, sizeof(pvar->settings.DefaultRhostsHostPrivateKeyFile));
  GetDlgItemText(dlg, IDC_LOCALUSERNAME, pvar->settings.DefaultRhostsLocalUserName, s
izeof(pvar->settings.DefaultRhostsLocalUserName));

  EndDialog(dlg, 1);
  return TRUE;
}

static BOOL CALLBACK default_auth_dlg_proc(HWND dlg, UINT msg, WPARAM wParam, LPARAM
lParam) {
  PTInstVar pvar;

  switch (msg) {
  case WM_INITDIALOG:
    pvar = (PTInstVar)lParam;
    SetWindowLong(dlg, DWL_USER, lParam);

    init_default_auth_dlg(pvar, dlg);
    return TRUE; /* because we do not set the focus */

  case WM_COMMAND:
    pvar = (PTInstVar)GetWindowLong(dlg, DWL_USER);

    switch (LOWORD(wParam)) {
    case IDOK:
      return end_default_auth_dlg(pvar, dlg);

    case IDCANCEL:
      EndDialog(dlg, 0);
      return TRUE;

    case IDC_CHOOSERSAFILE:
      choose_RSA_key_file(dlg);
      return TRUE;

    case IDC_CHOOSEHOSTRSAFILE:
      choose_host_RSA_key_file(dlg);
      return TRUE;

    default:
      return FALSE;
    }

  default:
    return FALSE;
  }
}

void AUTH_init(PTInstVar pvar) {
  pvar->auth_state.failed_method = SSH_AUTH_NONE;
  pvar->auth_state.auth_dialog = NULL;
  pvar->auth_state.user = NULL;
  pvar->auth_state.flags = 0;
  pvar->auth_state.TIS_prompt = NULL;
  pvar->auth_state.supported_types = 0;
  pvar->auth_state.cur_cred.method = SSH_AUTH_NONE;
  pvar->auth_state.cur_cred.password = NULL;
  pvar->auth_state.cur_cred.rhosts_client_user = NULL;
  pvar->auth_state.cur_cred.key_pair = NULL;
  AUTH_set_generic_mode(pvar);
}

void AUTH_set_generic_mode(PTInstVar pvar) {
  pvar->auth_state.mode = GENERIC_AUTH_MODE;
```



```
  destroy_malloced_string(&pvar->auth_state.TIS_prompt);
}

void AUTH_set_TIS_mode(PTInstVar pvar, char FAR * prompt, int len) {
  if (pvar->auth_state.cur_cred.method == SSH_AUTH_TIS) {
    pvar->auth_state.mode = TIS_AUTH_MODE;

    destroy_malloced_string(&pvar->auth_state.TIS_prompt);
    pvar->auth_state.TIS_prompt = malloc(len + 1);
    memcpy(pvar->auth_state.TIS_prompt, prompt, len);
    pvar->auth_state.TIS_prompt[len] = 0;
  } else {
    AUTH_set_generic_mode(pvar);
  }
}

void AUTH_do_default_cred_dialog(PTInstVar pvar) {
  HWND cur_active = GetActiveWindow();

  if (DialogBoxParam(hInst, MAKEINTRESOURCE(IDD_SSHAUTHSETUP),
    cur_active != NULL ? cur_active : pvar->NotificationWindow, default_auth_dlg_proc
, (LPARAM)pvar) == -1) {
    notify_nonfatal_error(pvar, "設定ダイアログボックスを開けませんでした．");
  }
}

void AUTH_destroy_cur_cred(PTInstVar pvar) {
  destroy_malloced_string(&pvar->auth_state.cur_cred.password);
  destroy_malloced_string(&pvar->auth_state.cur_cred.rhosts_client_user);
  if (pvar->auth_state.cur_cred.key_pair != NULL) {
    CRYPT_free_key_pair(pvar->auth_state.cur_cred.key_pair);
    pvar->auth_state.cur_cred.key_pair = NULL;
  }
}

static char FAR * get_auth_method_name(SSHAuthMethod auth) {
  switch (auth) {
  case SSH_AUTH_PASSWORD: return "パスワード";
  case SSH_AUTH_RSA: return "RSA";
  case SSH_AUTH_RHOSTS: return "rhosts";
  case SSH_AUTH_RHOSTS_RSA: return "RSAホスト認証を用いたrhosts";
  case SSH_AUTH_TIS: return "チャレンジ/レスポンス (TIS)";
  default:
    return "不明の方式による";
  }
}

void AUTH_get_auth_info(PTInstVar pvar, char FAR * dest, int len) {
  if (pvar->auth_state.user == NULL) {
    strncpy(dest, "なし", len);
  } else if (pvar->auth_state.cur_cred.method != SSH_AUTH_NONE) {
    _snprintf(dest, len, "ユーザ '%s', %s認証", pvar->auth_state.user,
      get_auth_method_name(pvar->auth_state.cur_cred.method));
  } else {
    _snprintf(dest, len, "ユーザ '%s', %s認証", pvar->auth_state.user,
      get_auth_method_name(pvar->auth_state.failed_method));
  }

  dest[len - 1] = 0;
}

void AUTH_notify_disconnecting(PTInstVar pvar) {
  if (pvar->auth_state.auth_dialog != NULL) {
    PostMessage(pvar->auth_state.auth_dialog, WM_COMMAND, IDCANCEL, 0);
      /* the main window might not go away if it's not enabled. (see vtwin.cpp) */
    EnableWindow(pvar->NotificationWindow, TRUE);
```

```
  }
}

void AUTH_end(PTInstVar pvar) {
  destroy_malloced_string(&pvar->auth_state.user);
  destroy_malloced_string(&pvar->auth_state.TIS_prompt);

  AUTH_destroy_cur_cred(pvar);
}
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
This code is copyright (C) 1998-1999 Robert O'Callahan.
See LICENSE.TXT for the license.
*/

#ifndef __AUTH_H
#define __AUTH_H

#include <stdio.h>

typedef struct {
  SSHAuthMethod method;
  char FAR * password;
  char FAR * rhosts_client_user;
  struct _CRYPTKeyPair FAR * key_pair;
} AUTHCred;

typedef enum { GENERIC_AUTH_MODE, TIS_AUTH_MODE } AuthMode;

typedef struct {
  char FAR * user;
  AUTHCred cur_cred;
  SSHAuthMethod failed_method;
  int flags;
  int supported_types;
  HWND auth_dialog;

  AuthMode mode;
  char FAR * TIS_prompt;
} AUTHState;

void AUTH_init(PTInstVar pvar);
char FAR * AUTH_get_user_name(PTInstVar pvar);
```



```
int AUTH_set_supported_auth_types(PTInstVar pvar, int types);
void AUTH_set_generic_mode(PTInstVar pvar);
void AUTH_set_TIS_mode(PTInstVar pvar, char FAR * prompt, int len);
void AUTH_advance_to_next_cred(PTInstVar pvar);
void AUTH_do_cred_dialog(PTInstVar pvar);
void AUTH_do_default_cred_dialog(PTInstVar pvar);
void AUTH_destroy_cur_cred(PTInstVar pvar);
void AUTH_get_auth_info(PTInstVar pvar, char FAR * dest, int len);
void AUTH_notify_disconnecting(PTInstVar pvar);
void AUTH_notify_end_error(PTInstVar pvar);
void AUTH_end(PTInstVar pvar);

#define AUTH_get_cur_cred(pvar) (&(pvar)->auth_state.cur_cred)

#endif
```

```
#include <winsock2.h>
#include "ftpwrap.h"


// Winsock async operation handle


void CommOpen(HWND HW, PTInstVar pvar) {
        WORD COMFlag;
        int Err;
        char ErrMsg[21];
        char P[50];

        MSG Msg;
        char HEntBuff[MAXGETHOSTSTRUCT];
        u_long addr;
        SOCKADDR_IN saddr;

        BOOL InvalidHost;
        BOOL BBuf;

        BOOL bHostMustResolve = FALSE;
        int i;

/* initialize ComVar */
        pvar->cv.InBuffCount = 0;
        pvar->cv.InPtr = 0;
        pvar->cv.OutBuffCount = 0;
        pvar->cv.OutPtr = 0;
        pvar->cv.HWin = HW;
        pvar->cv.Ready = FALSE;
        pvar->cv.Open = FALSE;
        pvar->cv.RetryCount = 0;
        pvar->socket = INVALID_SOCKET;
//      pvar->cv.ComID = INVALID_HANDLE_VALUE;
        pvar->cv.CanSend = TRUE;
        pvar->cv.RRQ = FALSE;

        if (strlen(pvar->HostName)==0) {
                PostMessage(pvar->cv.HWin, WM_USER_COMMNOTIFY, 0, FD_CLOSE);
                return;
        }

    TTXOpenTCP(pvar); /* TTPLUG */
        pvar->cv.Open = TRUE;

        for( i = 0; i < lstrlen( pvar->HostName ); i++ ) {
                if( ( pvar->HostName[i] < '.' ) || ( pvar->HostName[i] == '/' ) ||
                        ( pvar->HostName[i] > '9' ) ) {
                        if( ( i > 0 ) && ( pvar->HostName[i-1] == '0' ) &&
                                ( ( pvar->HostName[i] == 'X' ) || ( pvar->HostName[i]
 == 'x' ) ) )
                                // hex なら数値
                                continue;
                        if( ( ( pvar->HostName[i] >= 'a' ) && ( pvar->HostName[i] <=
'f' ) ) ||
                                ( ( pvar->HostName[i] >= 'A' ) && ( pvar->HostName[i]
 <= 'F' ) ) ) {
                                if( ( i > 1 ) &&
                                        ( ( pvar->HostName[i-1] == 'X' ) || ( pvar->H
ostName[i-1] == 'x' ) ) )
                                        continue;
                                if( ( i > 2 ) &&
                                        ( ( pvar->HostName[i-2] == 'X' ) || ( pvar->H
ostName[i-2] == 'x' ) ) )
                                        continue;
```



```
                        }
                        bHostMustResolve = TRUE;
                        break;
                }
        }
        if( ! bHostMustResolve ) {
                addr = inet_addr(pvar->HostName);
                InvalidHost = (addr == 0xffffffff);
        } else {
                pvar->HAsync = WSAAsyncGetHostByName(HW,WM_USER_GETHOST,
                        pvar->HostName,HEntBuff,sizeof(HEntBuff));
                if (pvar->HAsync == 0)
                        InvalidHost = TRUE;
                else {
                        pvar->cv.ComPort = 1; // set "getting host" flag
                        //  (see CVTWindow::OnSysCommand())
                        do {
                                if (GetMessage(&Msg,0,0,0))
                                {
                                        if ((Msg.hwnd==HW) &&
                                                ((Msg.message == WM_SYSCOMMAND) &&
                                                ((Msg.wParam & 0xfff0) == SC_CLOSE) |
|
                                                (Msg.message == WM_COMMAND) &&
                                                (LOWORD(Msg.wParam) == ID_MENUITEM_EX
IT) ||
                                                (Msg.message == WM_CLOSE)))     { /*
Exit when the user closes Tera Term */
                                                WSACancelAsyncRequest(pvar->HAsync);
                                                pvar->HAsync = 0;
                                                pvar->cv.ComPort = 0; // clear "getti
ng host" flag
                                                PostMessage(HW,Msg.message,Msg.wParam
,Msg.lParam);
                                                return;
                                        }
                                        if (Msg.message != WM_USER_GETHOST)
                                        { /* Prosess messages */
                                                TranslateMessage(&Msg);
                                                DispatchMessage(&Msg);
                                        }
                                } else {
                                        return;
                                }
                        } while (Msg.message!=WM_USER_GETHOST);
                        pvar->cv.ComPort = 0; // clear "getting host" flag
                        pvar->HAsync = 0;
                        InvalidHost = WSAGETASYNCERROR(Msg.lParam) != 0;
                        if (! InvalidHost) {
                                if (((PHOSTENT)HEntBuff)->h_addr_list != NULL)
                                        memcpy(&addr,
                                        ((PHOSTENT)HEntBuff)->h_addr_list[0],sizeof(a
ddr));
                                else
                                        InvalidHost = TRUE;
                        }
                }
        }
        if (InvalidHost) {
                if (pvar->cv.NoMsg==0)
                        MessageBox(pvar->cv.HWin,"ホスト名が無効です","error",
                        MB_TASKMODAL | MB_ICONEXCLAMATION);
        } else {
                pvar->socket = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
                if (pvar->socket==INVALID_SOCKET) {
                        InvalidHost = TRUE;
```

```
                        if (pvar->cv.NoMsg==0)
                                MessageBox(pvar->cv.HWin,"cannot connect","error",
                                MB_TASKMODAL | MB_ICONEXCLAMATION);
                } else {
                        BBuf = TRUE;
                        setsockopt(pvar->socket,(int)SOL_SOCKET,SO_OOBINLINE,(char FA
R *)&BBuf,sizeof(BBuf));

                        TTXWSAAsyncSelect(pvar, pvar->socket,pvar->cv.HWin,WM_USER_CO
MMNOTIFY, FD_CONNECT);
                        saddr.sin_family = AF_INET;
                        saddr.sin_port = htons(pvar->TCPPort);
                        saddr.sin_addr.s_addr = addr;
                        memset(saddr.sin_zero,0,8);

                        Err = TTXconnect(pvar, pvar->socket,(LPSOCKADDR)&saddr,sizeof
(saddr));
                        if (Err!=0 ) Err = WSAGetLastError();
                        if (Err==WSAEWOULDBLOCK ) {
                                /* Do nothing */
                        } else if (Err!=0 )
                                PostMessage(pvar->cv.HWin, WM_USER_COMMOPEN,0,
                                MAKELONG(FD_CONNECT,Err));
                }
        }

        if (InvalidHost) {
                PostMessage(pvar->cv.HWin, WM_USER_COMMNOTIFY, 0, FD_CLOSE);
                if ( pvar->cv.Open ) {
                        if ( pvar->socket!=INVALID_SOCKET ) closesocket(pvar->socket)
;
//      FreeWinsock();
                }
                PostMessage(pvar->cv.HWin, WM_USER_NOTIFY_CONNRESULT, (WPARAM)2, (LPA
RAM)pvar);
                return;
        }
}

void CommStart(PTInstVar pvar, LONG lParam) {
  char ErrMsg[31];
  char Temp[20];
  char Temp2[3];

  if (! pvar->cv.Open ) return;
  if ( pvar->cv.Ready ) return;
      ErrMsg[0] = 0;
      switch (HIWORD(lParam)) {
        case WSAECONNREFUSED:
          strcpy(ErrMsg,"接続が拒否されました"); break;
        case WSAENETUNREACH:
          strcpy(ErrMsg,"ネットワークに到達できませんでした"); break;
        case WSAETIMEDOUT:
          strcpy(ErrMsg,"接続がタイムアウトしました"); break;
        default:
          strcpy(ErrMsg,"cannot connect");
      }
      if (HIWORD(lParam)>0)
      {
        if (pvar->cv.NoMsg==0)
          MessageBox(pvar->cv.HWin,ErrMsg,"error",
            MB_TASKMODAL | MB_ICONEXCLAMATION);
        PostMessage(pvar->cv.HWin, WM_USER_COMMNOTIFY, 0, FD_CLOSE);
        PostMessage(pvar->cv.HWin, WM_USER_NOTIFY_CONNRESULT, (WPARAM)3, (LPARAM)pvar
);
        return;
```



```
      }
      TTXWSAAsyncSelect(pvar,pvar->socket,pvar->cv.HWin,WM_USER_COMMNOTIFY, FD_READ |
 FD_WRITE | FD_CLOSE);
//      TCPIPClosed = FALSE;
  pvar->cv.Ready = TRUE;
}

BOOL CommCanClose(PTInstVar pvar)
{ // check if data remains in buffer
  if (! pvar->cv.Open) return TRUE;
  if (pvar->cv.InBuffCount>0) return FALSE;
  return TRUE;
}

void CommClose(PTInstVar pvar)
{
  WORD COMFlag;

  if ( ! pvar->cv.Open ) return;
  pvar->cv.Open = FALSE;

  /* disable event message posting & flush buffer */
  pvar->cv.RRQ = FALSE;
  pvar->cv.Ready = FALSE;
  pvar->cv.InPtr = 0;
  pvar->cv.InBuffCount = 0;
  pvar->cv.OutPtr = 0;
  pvar->cv.OutBuffCount = 0;

  /* close port & release resources */
      if (pvar->HAsync!=0)
        WSACancelAsyncRequest(pvar->HAsync);
      pvar->HAsync = 0;
      if ( pvar->socket!=INVALID_SOCKET )
                closesocket(pvar->socket);
      pvar->socket = INVALID_SOCKET;
      TTXCloseTCP(pvar); /* TTPLUG */
        PostMessage(pvar->cv.HWin, WM_USER_NOTIFY_CONNRESULT, (WPARAM)4, (LPARAM)pvar
);

}

void CommProcRRQ(PTInstVar pvar)
{

  if ( ! pvar->cv.Ready ) return;
  /* disable receive request */
//      if (! TCPIPClosed)
//      TTXWSAAsyncSelect(pvar->socket,pvar->cv.HWin,WM_USER_COMMNOTIFY, FD_OOB | FD_
CLOSE);
  pvar->cv.RRQ = TRUE;
  CommReceive(pvar);
}

void CommReceive(PTInstVar pvar)
{
  DWORD C;
  DWORD DErr;

  if (! pvar->cv.Ready || ! pvar->cv.RRQ ||
      (pvar->cv.InBuffCount>=InBuffSize)) return;

  /* Compact buffer */
  if ((pvar->cv.InBuffCount>0) && (pvar->cv.InPtr>0))
  {
    memmove(pvar->cv.InBuff,&(pvar->cv.InBuff[pvar->cv.InPtr]),pvar->cv.InBuffCount);
```

```
    pvar->cv.InPtr = 0;
  }

  if (pvar->cv.InBuffCount<InBuffSize)
  {
        C = TTXrecv(pvar,pvar->socket, (char *)&(pvar->cv.InBuff[pvar->cv.InBuffCount
]),
                InBuffSize-pvar->cv.InBuffCount, 0);
        if (C == SOCKET_ERROR)
        {
          C = 0;
          WSAGetLastError();
        }
        pvar->cv.InBuffCount = pvar->cv.InBuffCount + C;
  }

  if (pvar->cv.InBuffCount==0)
  {
/*
        if (! TCPIPClosed)
          TTXWSAAsyncSelect(pvar,pvar->socket,pvar->cv.HWin,
            WM_USER_COMMNOTIFY, FD_READ | FD_WRITE | FD_CLOSE);
*/
    pvar->cv.RRQ = FALSE;
  }
}

void CommSend(PTInstVar pvar)
{
  int delay;
  BYTE LineEnd;
  int C, D, Max;
  DWORD DErr;

  if ((! pvar->cv.Open) || (! pvar->cv.Ready))
  {
    pvar->cv.OutBuffCount = 0;
    return;
  }

  if ((pvar->cv.OutBuffCount == 0) || (! pvar->cv.CanSend)) return;

  /* Max num of bytes to be written */
//      if (TCPIPClosed) pvar->cv.OutBuffCount = 0;
      Max = pvar->cv.OutBuffCount;

  if ( Max<=0 ) return;
  if ( Max > pvar->cv.OutBuffCount ) Max = pvar->cv.OutBuffCount;

  C = Max;
  delay = 0;

  /* Write to comm driver/Winsock */
      D = TTXsend(pvar,pvar->socket, (const char *)&(pvar->cv.OutBuff[pvar->cv.OutPtr
]), C, 0);
      if ( D==SOCKET_ERROR ) /* if error occurs */
      {
        WSAGetLastError(); /* Clear error */
        D = 0;
      }

  pvar->cv.OutBuffCount = pvar->cv.OutBuffCount - D;
  if ( pvar->cv.OutBuffCount==0 )
    pvar->cv.OutPtr = 0;
  else
    pvar->cv.OutPtr = pvar->cv.OutPtr + D;
```



```
}
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

#define WINDOWS

#include "ttxssh.h"
#include "util.h"

#include "rand.h"
#include "bn.h"
#include "md5.h"
#include "err.h"

#define do_crc(buf, len) (~(uint32)crc32(0xFFFFFFFF, (buf), (len)))
#define get_uint32(buf) get_uint32_MSBfirst((buf))

#define DEATTACK_OK             0
#define DEATTACK_DETECTED       1

/*
 * $Id: deattack.c,v 1.9 1998/06/27 00:02:22 futo Exp $ Cryptographic attack
 * detector for ssh - source code (C)1998 CORE-SDI, Buenos Aires Argentina
 * Ariel Futoransky(futo@core-sdi.com) <http://www.core-sdi.com>
 */

/* SSH Constants */
#define SSH_BLOCKSIZE 8

/* Hashing constants */
#define HASH_MINSIZE     8*2048
#define HASH_ENTRYSIZE   4
#define HASH_FACTOR(x)   ((x)*3/2)
#define HASH_UNUSEDCHAR  0xff
#define HASH_UNUSED      0xffffffff
#define HASH_IV          0xfffffffe
```



```
#define HASH_MINBLOCKS  (7*SSH_BLOCKSIZE)

/* Hash function (Input keys are cipher results) */
#define HASH(x) get_uint32(x)

#define CMP(a,b) memcmp(a, b, SSH_BLOCKSIZE)

static void crc_update(uint32 FAR * a, uint32 b) {
  b ^= *a;
  *a = do_crc((unsigned char FAR *)&b, sizeof(b));
}

/* check_crc
   detects if a block is used in a particular pattern
*/

static int check_crc(unsigned char FAR * S, unsigned char FAR * buf,
  uint32 len, unsigned char FAR * IV) {
  uint32 crc;
  unsigned char FAR * c;

  crc = 0;
  if (IV && !CMP(S, IV)) {
    crc_update(&crc, 1);
    crc_update(&crc, 0);
  }
  for (c = buf; c < buf + len; c += SSH_BLOCKSIZE) {
    if (!CMP(S, c)) {
      crc_update(&crc, 1);
      crc_update(&crc, 0);
    } else {
      crc_update(&crc, 0);
      crc_update(&crc, 0);
    }
  }

  return crc == 0;
}


/*
detect_attack
Detects a crc32 compensation attack on a packet
*/
static int detect_attack(CRYPTDetectAttack FAR * statics,
  unsigned char FAR * buf, uint32 len, unsigned char * FAR IV) {
  uint32 FAR * h = statics->h;
  uint32 n = statics->n;
  uint32 i, j;
  uint32 l;
  unsigned char FAR * c;
  unsigned char FAR * d;

  for (l = n; l < HASH_FACTOR(len / SSH_BLOCKSIZE); l = l << 2) {
  }

  if (h == NULL) {
    n = l;
    h = (uint32 FAR *)malloc(n*HASH_ENTRYSIZE);
  } else {
    if (l > n) {
      n = l;
      h = (uint32 FAR *)realloc(h, n*HASH_ENTRYSIZE);
    }
  }
```

```
  statics->h = h;
  statics->n = n;

  if (len <= HASH_MINBLOCKS) {
    for (c = buf; c < buf + len; c += SSH_BLOCKSIZE) {
      if (IV && (!CMP(c, IV))) {
        if ((check_crc(c, buf, len, IV)))
          return DEATTACK_DETECTED;
        else
          break;
      }
      for (d = buf; d < c; d += SSH_BLOCKSIZE) {
        if (!CMP(c, d)) {
          if ((check_crc(c, buf, len, IV)))
            return DEATTACK_DETECTED;
          else
            break;
        }
      }
    }
    return (DEATTACK_OK);
  }
  memset(h, HASH_UNUSEDCHAR, n*HASH_ENTRYSIZE);

  if (IV) {
    h[HASH(IV) & (n - 1)] = HASH_IV;
  }

  for (c = buf, j = 0; c < (buf + len); c += SSH_BLOCKSIZE, j++) {
    for (i = HASH(c) & (n - 1); h[i] != HASH_UNUSED; i = (i + 1) & (n - 1)) {
      if (h[i] == HASH_IV) {
        if (!CMP(c, IV)) {
          if (check_crc(c, buf, len, IV))
            return DEATTACK_DETECTED;
          else
            break;
        }
      } else if (!CMP(c, buf + h[i]*SSH_BLOCKSIZE)) {
        if (check_crc(c, buf, len, IV))
          return DEATTACK_DETECTED;
        else
          break;
      }
    }
    h[i] = j;
  }

  return DEATTACK_OK;
}

BOOL CRYPT_detect_attack(PTInstVar pvar, unsigned char FAR * buf, int bytes) {
  if (SSHv1(pvar)) {
    switch (pvar->crypt_state.sender_cipher) {
    case SSH_CIPHER_NONE:
      return FALSE;
    case SSH_CIPHER_IDEA:
      return detect_attack(&pvar->crypt_state.detect_attack_statics, buf, bytes,
        pvar->crypt_state.dec.cIDEA.ivec) == DEATTACK_DETECTED;
    default:
      return detect_attack(&pvar->crypt_state.detect_attack_statics, buf, bytes,
        NULL) == DEATTACK_DETECTED;
    }
  } else {
    return FALSE;
  }
}
```



```
static void no_encrypt(PTInstVar pvar, unsigned char FAR * buf, int bytes) {
}

static void c3DES_encrypt(PTInstVar pvar, unsigned char FAR * buf, int bytes) {
  Cipher3DESState FAR * encryptstate = &pvar->crypt_state.enc.c3DES;

  des_ncbc_encrypt((des_cblock FAR *)buf, (des_cblock FAR *)buf, bytes,
        encryptstate->k1, &encryptstate->ivec1, DES_ENCRYPT);
  des_ncbc_encrypt((des_cblock FAR *)buf, (des_cblock FAR *)buf, bytes,
          encryptstate->k2, &encryptstate->ivec2, DES_DECRYPT);
  des_ncbc_encrypt((des_cblock FAR *)buf, (des_cblock FAR *)buf, bytes,
          encryptstate->k3, &encryptstate->ivec3, DES_ENCRYPT);
}

static void c3DES_decrypt(PTInstVar pvar, unsigned char FAR * buf, int bytes) {
  Cipher3DESState FAR * decryptstate = &pvar->crypt_state.dec.c3DES;

  des_ncbc_encrypt((des_cblock FAR *)buf, (des_cblock FAR *)buf, bytes,
          decryptstate->k3, &decryptstate->ivec3, DES_DECRYPT);
  des_ncbc_encrypt((des_cblock FAR *)buf, (des_cblock FAR *)buf, bytes,
          decryptstate->k2, &decryptstate->ivec2, DES_ENCRYPT);
  des_ncbc_encrypt((des_cblock FAR *)buf, (des_cblock FAR *)buf, bytes,
          decryptstate->k1, &decryptstate->ivec1, DES_DECRYPT);
}

static void cDES_encrypt(PTInstVar pvar, unsigned char FAR * buf, int bytes) {
  CipherDESState FAR * encryptstate = &pvar->crypt_state.enc.cDES;

  des_ncbc_encrypt((des_cblock FAR *)buf, (des_cblock FAR *)buf, bytes,
        encryptstate->k, &encryptstate->ivec, DES_ENCRYPT);
}

static void cDES_decrypt(PTInstVar pvar, unsigned char FAR * buf, int bytes) {
  CipherDESState FAR * decryptstate = &pvar->crypt_state.dec.cDES;

  des_ncbc_encrypt((des_cblock FAR *)buf, (des_cblock FAR *)buf, bytes,
          decryptstate->k, &decryptstate->ivec, DES_DECRYPT);
}

static void cIDEA_encrypt(PTInstVar pvar, unsigned char FAR * buf, int bytes) {
  CipherIDEAState FAR * encryptstate = &pvar->crypt_state.enc.cIDEA;
  int num = 0;

  idea_cfb64_encrypt(buf, buf, bytes, &encryptstate->k, encryptstate->ivec,
        &num, IDEA_ENCRYPT);
}

static void cIDEA_decrypt(PTInstVar pvar, unsigned char FAR * buf, int bytes) {
  CipherIDEAState FAR * decryptstate = &pvar->crypt_state.dec.cIDEA;
  int num = 0;

  idea_cfb64_encrypt(buf, buf, bytes, &decryptstate->k, decryptstate->ivec,
        &num, IDEA_DECRYPT);
}

static void flip_endianness(unsigned char FAR * cbuf, int bytes) {
  uint32 FAR * buf = (uint32 FAR *)cbuf;
  int count = bytes/4;

  while (count > 0) {
    uint32 w = *buf;

    *buf = ((w << 24) & 0xFF000000) | ((w << 8) & 0x00FF0000)
      | ((w >> 8) & 0x0000FF00) | ((w >> 24) & 0x000000FF);
    count--;
```

```
    buf++;
  }
}

static void cBlowfish_encrypt(PTInstVar pvar, unsigned char FAR * buf, int bytes) {
  CipherBlowfishState FAR * encryptstate = &pvar->crypt_state.enc.cBlowfish;

  flip_endianness(buf, bytes);
  BF_cbc_encrypt(buf, buf, bytes, &encryptstate->k, encryptstate->ivec,
    BF_ENCRYPT);
  flip_endianness(buf, bytes);
}

static void cBlowfish_decrypt(PTInstVar pvar, unsigned char FAR * buf, int bytes) {
  CipherBlowfishState FAR * decryptstate = &pvar->crypt_state.dec.cBlowfish;

  flip_endianness(buf, bytes);
  BF_cbc_encrypt(buf, buf, bytes, &decryptstate->k, decryptstate->ivec,
    BF_DECRYPT);
  flip_endianness(buf, bytes);
}

static void cRC4_encrypt(PTInstVar pvar, unsigned char FAR * buf, int bytes) {
  CipherRC4State FAR * encryptstate = &pvar->crypt_state.enc.cRC4;
  int num = 0;

  RC4(&encryptstate->k, bytes, buf, buf);
}

static void cRC4_decrypt(PTInstVar pvar, unsigned char FAR * buf, int bytes) {
  CipherRC4State FAR * decryptstate = &pvar->crypt_state.dec.cRC4;
  int num = 0;

  RC4(&decryptstate->k, bytes, buf, buf);
}

void CRYPT_set_random_data(PTInstVar pvar, unsigned char FAR * buf, int bytes) {
  RAND_bytes(buf, bytes);
}

void CRYPT_initialize_random_numbers(PTInstVar pvar) {
  RAND_screen();
}

static BIGNUM FAR * get_bignum(unsigned char FAR * bytes) {
  int bits = get_ushort16_MSBfirst(bytes);

  return BN_bin2bn(bytes + 2, (bits + 7)/8, NULL);
}

static RSA FAR * make_key(PTInstVar pvar,
                          int bits, unsigned char FAR * exp, unsigned char FAR * mod)
 {
  RSA FAR * key = RSA_new();

  if (key != NULL) {
    key->e = get_bignum(exp);
    key->n = get_bignum(mod);
  }

  if (key == NULL || key->e == NULL || key->n == NULL) {
    notify_fatal_error(pvar, "RSA鍵のセットアップ中にエラーが発生しました．");

    if (key != NULL) {
      if (key->e != NULL) {
        BN_free(key->e);
```



```
      }
      if (key->n != NULL) {
        BN_free(key->n);
      }
      RSA_free(key);
    }

    return NULL;
  } else {
    return key;
  }
}

void CRYPT_set_server_cookie(PTInstVar pvar, unsigned char FAR * cookie) {
  if (SSHv1(pvar)) {
    memcpy(pvar->crypt_state.server_cookie, cookie, SSH_COOKIE_LENGTH);
  } else {
    memcpy(pvar->crypt_state.server_cookie, cookie, SSH2_COOKIE_LENGTH);
  }
}

void CRYPT_set_client_cookie(PTInstVar pvar, unsigned char FAR * cookie) {
  if (SSHv2(pvar)) {
    memcpy(pvar->crypt_state.client_cookie, cookie, SSH2_COOKIE_LENGTH);
  }
}

BOOL CRYPT_set_server_RSA_key(PTInstVar pvar,
  int bits, unsigned char FAR * exp, unsigned char FAR * mod) {
  pvar->crypt_state.server_key.RSA_key = make_key(pvar, bits, exp, mod);

  return pvar->crypt_state.server_key.RSA_key != NULL;
}

BOOL CRYPT_set_host_RSA_key(PTInstVar pvar,
  int bits, unsigned char FAR * exp, unsigned char FAR * mod) {
  pvar->crypt_state.host_key.RSA_key = make_key(pvar, bits, exp, mod);

  return pvar->crypt_state.host_key.RSA_key != NULL;
}

BOOL CRYPT_set_supported_ciphers(PTInstVar pvar, int sender_ciphers,
  int receiver_ciphers) {
  const int cipher_mask =
      (1 << SSH_CIPHER_IDEA) | (1 << SSH_CIPHER_DES)
    | (1 << SSH_CIPHER_3DES) | (1 << SSH_CIPHER_RC4)
    | (1 << SSH_CIPHER_BLOWFISH);

  sender_ciphers &= cipher_mask;
  receiver_ciphers &= cipher_mask;
  pvar->crypt_state.supported_sender_ciphers = sender_ciphers;
  pvar->crypt_state.supported_receiver_ciphers = receiver_ciphers;

  if (sender_ciphers == 0) {
    notify_fatal_error(pvar, "サーバはTTSSHがサポートしているいずれの暗号アルゴリズム
もサポートしていないため、"
      "TTSSHからサーバへの安全な通信路を確立できません．接続を切断します．");
    return FALSE;
  } else if (receiver_ciphers == 0) {
    notify_fatal_error(pvar, "サーバはTTSSHがサポートしているいずれの暗号アルゴリズム
もサポートしていないため、"
      "サーバからTTSSHへの安全な通信路を確立できません．接続を切断します．");
    return FALSE;
  } else {
    return TRUE;
  }
```

```
}

int CRYPT_get_decryption_block_size(PTInstVar pvar) {
  return 8;
}

int CRYPT_get_encryption_block_size(PTInstVar pvar) {
  return 8;
}

int CRYPT_get_receiver_MAC_size(PTInstVar pvar) {
  return 0;
}

BOOL CRYPT_verify_receiver_MAC(PTInstVar pvar, uint32 sequence_number,
  char FAR * data, int len, char FAR * MAC) {
  return TRUE;
}

int CRYPT_get_sender_MAC_size(PTInstVar pvar) {
  return 0;
}

void CRYPT_build_sender_MAC(PTInstVar pvar, uint32 sequence_number,
  char FAR * data, int len, char FAR * MAC) {
}

static int choose_cipher(PTInstVar pvar, int supported) {
  int i;

  for (i = 0; pvar->session_settings.CipherOrder[i] != 0; i++) {
    int cipher = pvar->session_settings.CipherOrder[i] - '0';

    if (cipher == SSH_CIPHER_NONE) {
      break;
    } else if ((supported & (1 << cipher)) != 0) {
      return cipher;
    }
  }

  return SSH_CIPHER_NONE;
}

BOOL CRYPT_choose_ciphers(PTInstVar pvar) {
  pvar->crypt_state.sender_cipher = choose_cipher(pvar,
    pvar->crypt_state.supported_sender_ciphers);
  pvar->crypt_state.receiver_cipher = choose_cipher(pvar,
    pvar->crypt_state.supported_receiver_ciphers);

  if (pvar->crypt_state.sender_cipher == SSH_CIPHER_NONE
    || pvar->crypt_state.receiver_cipher == SSH_CIPHER_NONE) {
    notify_fatal_error(pvar,
      "TTSSHとサーバ両方ともサポートしている暗号アルゴリズムが全て無効になっています
．\n"
    "このサーバと通信するためには、TTSSH設定ダイアログボックスから、他の暗号アルゴリ
ズムを有効にしてから再接続して下さい．\n"
    "接続を切断します．");
    return FALSE;
  } else {
    return TRUE;
  }
}

int CRYPT_get_encrypted_session_key_len(PTInstVar pvar) {
  int server_key_bits = BN_num_bits(pvar->crypt_state.server_key.RSA_key->n);
  int host_key_bits = BN_num_bits(pvar->crypt_state.host_key.RSA_key->n);
```



```
  int server_key_bytes = (server_key_bits + 7)/8;
  int host_key_bytes = (host_key_bits + 7)/8;

  if (server_key_bits < host_key_bits) {
    return host_key_bytes;
  } else {
    return server_key_bytes;
  }
}

int CRYPT_choose_session_key(PTInstVar pvar, unsigned char FAR * encrypted_key_buf) {
  int server_key_bits = BN_num_bits(pvar->crypt_state.server_key.RSA_key->n);
  int host_key_bits = BN_num_bits(pvar->crypt_state.host_key.RSA_key->n);
  int server_key_bytes = (server_key_bits + 7)/8;
  int host_key_bytes = (host_key_bits + 7)/8;
  int encrypted_key_bytes;
  int bit_delta;

  if (server_key_bits < host_key_bits) {
    encrypted_key_bytes = host_key_bytes;
    bit_delta = host_key_bits - server_key_bits;
  } else {
    encrypted_key_bytes = server_key_bytes;
    bit_delta = server_key_bits - host_key_bits;
  }

  if (bit_delta < 128 || server_key_bits < 512 || host_key_bits < 512) {
    notify_fatal_error(pvar, "サーバのRSA鍵は強度に問題があります．安全な通信路を確立
することができません．");
    return 0;
  } else {
    /* following Goldberg's code, I'm using MD5(servkey->n || hostkey->n || cookie)
       for the session ID, rather than the one specified in the RFC */
    int session_buf_len = server_key_bytes + host_key_bytes + 8;
    char FAR * session_buf = (char FAR *)malloc(session_buf_len);
    char session_id[16];
    int i;

    BN_bn2bin(pvar->crypt_state.host_key.RSA_key->n, session_buf);
    BN_bn2bin(pvar->crypt_state.server_key.RSA_key->n, session_buf + host_key_bytes);
    memcpy(session_buf + server_key_bytes + host_key_bytes, pvar->crypt_state.server_
cookie, 8);
    MD5(session_buf, session_buf_len, session_id);

    free(session_buf);

    RAND_bytes(pvar->crypt_state.sender_cipher_key, SSH_SESSION_KEY_LENGTH);
    memcpy(pvar->crypt_state.receiver_cipher_key,
      pvar->crypt_state.sender_cipher_key, SSH_SESSION_KEY_LENGTH);

    memcpy(encrypted_key_buf + encrypted_key_bytes - SSH_SESSION_KEY_LENGTH,
      pvar->crypt_state.sender_cipher_key, SSH_SESSION_KEY_LENGTH);
    for (i = 0; i < sizeof(session_id); i++) {
      encrypted_key_buf[encrypted_key_bytes - SSH_SESSION_KEY_LENGTH + i]
        ^= session_id[i];
    }

    if (host_key_bits > server_key_bits) {
          if (RSA_public_encrypt(SSH_SESSION_KEY_LENGTH,
        encrypted_key_buf + encrypted_key_bytes - SSH_SESSION_KEY_LENGTH,
        encrypted_key_buf + encrypted_key_bytes - server_key_bytes,
              pvar->crypt_state.server_key.RSA_key, RSA_PKCS1_PADDING) < 0) return 0;

          if (RSA_public_encrypt(server_key_bytes,
        encrypted_key_buf + encrypted_key_bytes - server_key_bytes,
        encrypted_key_buf,
```

```
              pvar->crypt_state.host_key.RSA_key, RSA_PKCS1_PADDING) < 0) return 0;
    } else {
           if (RSA_public_encrypt(SSH_SESSION_KEY_LENGTH,
        encrypted_key_buf + encrypted_key_bytes - SSH_SESSION_KEY_LENGTH,
        encrypted_key_buf + encrypted_key_bytes - host_key_bytes,
              pvar->crypt_state.host_key.RSA_key, RSA_PKCS1_PADDING) < 0) return 0;

          if (RSA_public_encrypt(host_key_bytes,
        encrypted_key_buf + encrypted_key_bytes - host_key_bytes,
        encrypted_key_buf,
              pvar->crypt_state.server_key.RSA_key, RSA_PKCS1_PADDING) < 0) return 0;
    }
  }

  return 1;
}

int CRYPT_generate_RSA_challenge_response(PTInstVar pvar, unsigned char FAR * challen
ge,
  int challenge_len, unsigned char FAR * response) {
  int server_key_bits = BN_num_bits(pvar->crypt_state.server_key.RSA_key->n);
  int host_key_bits = BN_num_bits(pvar->crypt_state.host_key.RSA_key->n);
  int server_key_bytes = (server_key_bits + 7)/8;
  int host_key_bytes = (host_key_bits + 7)/8;
  int session_buf_len = server_key_bytes + host_key_bytes + 8;
  char FAR * session_buf = (char FAR *)malloc(session_buf_len);
  char decrypted_challenge[48];
  int decrypted_challenge_len;

  decrypted_challenge_len =
    RSA_private_decrypt(challenge_len, challenge, challenge,
      AUTH_get_cur_cred(pvar)->key_pair->RSA_key, RSA_PKCS1_PADDING);
  if (decrypted_challenge_len < 0) {
    free(session_buf);
    return 0;
  }
  if (decrypted_challenge_len >= SSH_RSA_CHALLENGE_LENGTH) {
    memcpy(decrypted_challenge, challenge + decrypted_challenge_len - SSH_RSA_CHALLEN
GE_LENGTH,
      SSH_RSA_CHALLENGE_LENGTH);
  } else {
    memset(decrypted_challenge, 0, SSH_RSA_CHALLENGE_LENGTH - decrypted_challenge_len
);
    memcpy(decrypted_challenge + SSH_RSA_CHALLENGE_LENGTH - decrypted_challenge_len,
      challenge, decrypted_challenge_len);
  }

  BN_bn2bin(pvar->crypt_state.host_key.RSA_key->n, session_buf);
  BN_bn2bin(pvar->crypt_state.server_key.RSA_key->n, session_buf + host_key_bytes);
  memcpy(session_buf + server_key_bytes + host_key_bytes, pvar->crypt_state.server_co
okie, 8);
  MD5(session_buf, session_buf_len, decrypted_challenge + 32);

  free(session_buf);

  MD5(decrypted_challenge, 48, response);

  return 1;
}

static void c3DES_init(char FAR * session_key, Cipher3DESState FAR * state) {
  des_set_key((des_cblock FAR *)session_key, state->k1);
  des_set_key((des_cblock FAR *)(session_key + 8), state->k2);
  des_set_key((des_cblock FAR *)(session_key + 16), state->k3);
  memset(state->ivec1, 0, 8);
  memset(state->ivec2, 0, 8);
```



```
  memset(state->ivec3, 0, 8);
}

static void cDES_init(char FAR * session_key, CipherDESState FAR * state) {
  des_set_key((des_cblock FAR *)session_key, state->k);
  memset(state->ivec, 0, 8);
}

static void cIDEA_init(char FAR * session_key, CipherIDEAState FAR * state) {
  idea_set_encrypt_key(session_key, &state->k);
  memset(state->ivec, 0, 8);
}

static void cBlowfish_init(char FAR * session_key, CipherBlowfishState FAR * state) {
  BF_set_key(&state->k, 32, session_key);
  memset(state->ivec, 0, 8);
}

BOOL CRYPT_start_encryption(PTInstVar pvar) {
  char FAR * encryption_key = pvar->crypt_state.sender_cipher_key;
  char FAR * decryption_key = pvar->crypt_state.receiver_cipher_key;
  BOOL isOK = TRUE;

  switch (pvar->crypt_state.sender_cipher) {
  case SSH_CIPHER_3DES: {
    c3DES_init(encryption_key, &pvar->crypt_state.enc.c3DES);
    pvar->crypt_state.encrypt = c3DES_encrypt;
    break;
  }
  case SSH_CIPHER_IDEA: {
    cIDEA_init(encryption_key, &pvar->crypt_state.enc.cIDEA);
    pvar->crypt_state.encrypt = cIDEA_encrypt;
    break;
  }
  case SSH_CIPHER_DES: {
    cDES_init(encryption_key, &pvar->crypt_state.enc.cDES);
    pvar->crypt_state.encrypt = cDES_encrypt;
    break;
  }
  case SSH_CIPHER_RC4: {
    RC4_set_key(&pvar->crypt_state.enc.cRC4.k, 16, encryption_key + 16);
    pvar->crypt_state.encrypt = cRC4_encrypt;
    break;
  }
  case SSH_CIPHER_BLOWFISH: {
    cBlowfish_init(encryption_key, &pvar->crypt_state.enc.cBlowfish);
    pvar->crypt_state.encrypt = cBlowfish_encrypt;
    break;
  }
  default:
    isOK = FALSE;
  }

  switch (pvar->crypt_state.receiver_cipher) {
  case SSH_CIPHER_3DES: {
    c3DES_init(decryption_key, &pvar->crypt_state.dec.c3DES);
    pvar->crypt_state.decrypt = c3DES_decrypt;
    break;
  }
  case SSH_CIPHER_IDEA: {
    cIDEA_init(decryption_key, &pvar->crypt_state.dec.cIDEA);
    pvar->crypt_state.decrypt = cIDEA_decrypt;
    break;
  }
  case SSH_CIPHER_DES: {
    cDES_init(decryption_key, &pvar->crypt_state.dec.cDES);
```

```
    pvar->crypt_state.decrypt = cDES_decrypt;
    break;
  }
  case SSH_CIPHER_RC4: {
    RC4_set_key(&pvar->crypt_state.dec.cRC4.k, 16, decryption_key);
    pvar->crypt_state.decrypt = cRC4_decrypt;
    break;
  }
  case SSH_CIPHER_BLOWFISH: {
    cBlowfish_init(decryption_key, &pvar->crypt_state.dec.cBlowfish);
    pvar->crypt_state.decrypt = cBlowfish_decrypt;
    break;
  }
  default:
    isOK = FALSE;
  }

  if (!isOK) {
    notify_fatal_error(pvar, "暗号アルゴリズムが選択されていません．"); /* should nev
er get here! */
    return FALSE;
  } else {
    memset(encryption_key, 0, CRYPT_KEY_LENGTH);
    memset(decryption_key, 0, CRYPT_KEY_LENGTH);
    return TRUE;
  }
}

void CRYPT_init(PTInstVar pvar) {
  pvar->crypt_state.encrypt = no_encrypt;
  pvar->crypt_state.decrypt = no_encrypt;
  pvar->crypt_state.sender_cipher = SSH_CIPHER_NONE;
  pvar->crypt_state.receiver_cipher = SSH_CIPHER_NONE;
  pvar->crypt_state.server_key.RSA_key = NULL;
  pvar->crypt_state.host_key.RSA_key = NULL;

  pvar->crypt_state.detect_attack_statics.h = NULL;
  pvar->crypt_state.detect_attack_statics.n = HASH_MINSIZE/HASH_ENTRYSIZE;
}

static char FAR * get_cipher_name(int cipher) {
  switch (cipher) {
  case SSH_CIPHER_NONE: return "なし";
  case SSH_CIPHER_3DES: return "3DES (168ビット鍵)";
  case SSH_CIPHER_DES:  return "DES (56ビット鍵)";
  case SSH_CIPHER_IDEA: return "IDEA (128ビット鍵)";
  case SSH_CIPHER_RC4:  return "RC4 (128ビット鍵)";
  case SSH_CIPHER_BLOWFISH:
                        return "Blowfish (256ビット鍵)";
  default:              return "不明";
  }
}

void CRYPT_get_cipher_info(PTInstVar pvar, char FAR * dest, int len) {
  _snprintf(dest, len, "TTSSH→サーバ %s / サーバ→TTSSH %s",
    get_cipher_name(pvar->crypt_state.sender_cipher),
    get_cipher_name(pvar->crypt_state.receiver_cipher));
  dest[len - 1] = 0;
}

void CRYPT_get_server_key_info(PTInstVar pvar, char FAR * dest, int len) {
  if (pvar->crypt_state.server_key.RSA_key == NULL || pvar->crypt_state.host_key.RSA_
key == NULL) {
    strncpy(dest, "なし", len);
  } else {
    _snprintf(dest, len, "サーバ %dビット鍵 / ホスト %dビット鍵",
```



```
      BN_num_bits(pvar->crypt_state.server_key.RSA_key->n),
      BN_num_bits(pvar->crypt_state.host_key.RSA_key->n));
  }

  dest[len - 1] = 0;
}

static void destroy_public_key(CRYPTPublicKey FAR * key) {
  if (key->RSA_key != NULL) {
    RSA_free(key->RSA_key);
    key->RSA_key = NULL;
  }
}

void CRYPT_free_public_key(CRYPTPublicKey FAR * key) {
  destroy_public_key(key);
  free(key);
}

void CRYPT_end(PTInstVar pvar) {
  destroy_public_key(&pvar->crypt_state.host_key);
  destroy_public_key(&pvar->crypt_state.server_key);

  if (pvar->crypt_state.detect_attack_statics.h != NULL) {
    memset(pvar->crypt_state.detect_attack_statics.h, 0,
      pvar->crypt_state.detect_attack_statics.n*HASH_ENTRYSIZE);
    free(pvar->crypt_state.detect_attack_statics.h);
  }

  memset(pvar->crypt_state.sender_cipher_key, 0, sizeof(pvar->crypt_state.sender_ciph
er_key));
  memset(pvar->crypt_state.receiver_cipher_key, 0, sizeof(pvar->crypt_state.receiver_
cipher_key));
  memset(pvar->crypt_state.server_cookie, 0, sizeof(pvar->crypt_state.server_cookie))
;
  memset(pvar->crypt_state.client_cookie, 0, sizeof(pvar->crypt_state.client_cookie))
;
  memset(&pvar->crypt_state.enc, 0, sizeof(pvar->crypt_state.enc));
  memset(&pvar->crypt_state.dec, 0, sizeof(pvar->crypt_state.dec));
}

int CRYPT_passphrase_decrypt(int cipher, char FAR * passphrase, char FAR * buf,
  int bytes) {
  unsigned char passphrase_key[16];

  MD5(passphrase, strlen(passphrase), passphrase_key);

  switch (cipher) {
  case SSH_CIPHER_3DES: {
    Cipher3DESState state;

    des_set_key((des_cblock FAR *)passphrase_key, state.k1);
    des_set_key((des_cblock FAR *)(passphrase_key + 8), state.k2);
    des_set_key((des_cblock FAR *)passphrase_key, state.k3);
    memset(state.ivec1, 0, 8);
    memset(state.ivec2, 0, 8);
    memset(state.ivec3, 0, 8);
    des_ncbc_encrypt((des_cblock FAR *)buf, (des_cblock FAR *)buf, bytes,
      state.k3, &state.ivec3, DES_DECRYPT);
    des_ncbc_encrypt((des_cblock FAR *)buf, (des_cblock FAR *)buf, bytes,
      state.k2, &state.ivec2, DES_ENCRYPT);
    des_ncbc_encrypt((des_cblock FAR *)buf, (des_cblock FAR *)buf, bytes,
      state.k1, &state.ivec1, DES_DECRYPT);
    break;
  }
```

```
  case SSH_CIPHER_IDEA: {
    CipherIDEAState state;
    int num = 0;

    cIDEA_init(passphrase_key, &state);
    idea_cfb64_encrypt(buf, buf, bytes, &state.k, state.ivec,
          &num, IDEA_DECRYPT);
    break;
  }

  case SSH_CIPHER_DES: {
    CipherDESState state;

    cDES_init(passphrase_key, &state);
    des_ncbc_encrypt((des_cblock FAR *)buf, (des_cblock FAR *)buf, bytes,
           state.k, &state.ivec, DES_DECRYPT);
    break;
  }

  case SSH_CIPHER_RC4: {
    CipherRC4State state;
    int num = 0;

    RC4_set_key(&state.k, 16, passphrase_key);
    RC4(&state.k, bytes, buf, buf);
    break;
  }

  case SSH_CIPHER_BLOWFISH: {
    CipherBlowfishState state;

    BF_set_key(&state.k, 16, passphrase_key);
    memset(state.ivec, 0, 8);
    flip_endianness(buf, bytes);
    BF_cbc_encrypt(buf, buf, bytes, &state.k, state.ivec, BF_DECRYPT);
    flip_endianness(buf, bytes);
    break;
  }

  case SSH_CIPHER_NONE:
    break;

  default:
    memset(passphrase_key, 0, sizeof(passphrase_key));
    return 0;
  }

  memset(passphrase_key, 0, sizeof(passphrase_key));
  return 1;
}

void CRYPT_free_key_pair(CRYPTKeyPair FAR * key_pair) {
  RSA_free(key_pair->RSA_key);
  free(key_pair);
}
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
This code is copyright (C) 1998-1999 Robert O'Callahan.
See LICENSE.TXT for the license.
*/

#ifndef __CRYPT_H
#define __CRYPT_H

#include "rsa.h"
#include "des.h"
#include "idea.h"
#include "rc4.h"
#include "blowfish.h"

#define SSH_SESSION_KEY_LENGTH    32
#define SSH_RSA_CHALLENGE_LENGTH  32
#define SSH_COOKIE_LENGTH         8
#define SSH2_COOKIE_LENGTH         16

#define CRYPT_KEY_LENGTH     32
#define COOKIE_LENGTH        16

typedef struct {
  des_key_schedule k1;
  des_key_schedule k2;
  des_key_schedule k3;
  des_cblock ivec1;
  des_cblock ivec2;
  des_cblock ivec3;
} Cipher3DESState;

typedef struct {
  IDEA_KEY_SCHEDULE k;
```



```
  unsigned char ivec[8];
} CipherIDEAState;

typedef struct {
  des_key_schedule k;
  des_cblock ivec;
} CipherDESState;

typedef struct {
  RC4_KEY k;
} CipherRC4State;

typedef struct {
  BF_KEY k;
  unsigned char ivec[8];
} CipherBlowfishState;

typedef struct {
  uint32 FAR * h;
  uint32 n;
} CRYPTDetectAttack;

typedef struct {
  RSA FAR * RSA_key;
} CRYPTPublicKey;

typedef struct _CRYPTKeyPair {
  RSA FAR * RSA_key;
} CRYPTKeyPair;

typedef union {
  Cipher3DESState c3DES;
  CipherIDEAState cIDEA;
  CipherDESState cDES;
  CipherRC4State cRC4;
  CipherBlowfishState cBlowfish;
} CRYPTCipherState;

typedef void (* CRYPTCryptFun)(PTInstVar pvar, unsigned char FAR * buf, int bytes);

typedef struct {
  CRYPTDetectAttack detect_attack_statics;

  CRYPTPublicKey server_key;
  CRYPTPublicKey host_key;

  char server_cookie[COOKIE_LENGTH];
  char client_cookie[COOKIE_LENGTH];

  int supported_sender_ciphers;
  int supported_receiver_ciphers;
  int sender_cipher;
  int receiver_cipher;
  char sender_cipher_key[CRYPT_KEY_LENGTH];
  char receiver_cipher_key[CRYPT_KEY_LENGTH];
  CRYPTCryptFun encrypt;
  CRYPTCryptFun decrypt;
  CRYPTCipherState enc;
  CRYPTCipherState dec;
} CRYPTState;

void CRYPT_init(PTInstVar pvar);
/* this function is called during 'slack time' while we wait for a response
   from the server. Therefore we have some time available to do some
   moderately expensive computations. */
void CRYPT_initialize_random_numbers(PTInstVar pvar);
```

```
void CRYPT_set_random_data(PTInstVar pvar, unsigned char FAR * buf, int bytes);
void CRYPT_end(PTInstVar pvar);

void CRYPT_get_cipher_info(PTInstVar pvar, char FAR * dest, int len);
void CRYPT_get_server_key_info(PTInstVar pvar, char FAR * dest, int len);

void CRYPT_set_server_cookie(PTInstVar pvar, unsigned char FAR * cookie);
void CRYPT_set_client_cookie(PTInstVar pvar, unsigned char FAR * cookie);
#define CRYPT_get_server_cookie(pvar) ((pvar)->crypt_state.server_cookie)

void CRYPT_free_key_pair(CRYPTKeyPair FAR * key_pair);
void CRYPT_free_public_key(CRYPTPublicKey FAR * key);

BOOL CRYPT_set_server_RSA_key(PTInstVar pvar,
  int bits, unsigned char FAR * exp, unsigned char FAR * mod);
BOOL CRYPT_set_host_RSA_key(PTInstVar pvar,
  int bits, unsigned char FAR * exp, unsigned char FAR * mod);
int CRYPT_get_encrypted_session_key_len(PTInstVar pvar);
int CRYPT_choose_session_key(PTInstVar pvar, unsigned char FAR * encrypted_key_buf);
BOOL CRYPT_start_encryption(PTInstVar pvar);
int CRYPT_generate_RSA_challenge_response(PTInstVar pvar, unsigned char FAR * challen
ge,
                                          int challenge_len, unsigned char FAR * res
ponse);

int CRYPT_get_receiver_MAC_size(PTInstVar pvar);
BOOL CRYPT_verify_receiver_MAC(PTInstVar pvar, uint32 sequence_number,
  char FAR * data, int len, char FAR * MAC);
int CRYPT_get_sender_MAC_size(PTInstVar pvar);
void CRYPT_build_sender_MAC(PTInstVar pvar, uint32 sequence_number,
  char FAR * data, int len, char FAR * MAC);

BOOL CRYPT_set_supported_ciphers(PTInstVar pvar, int sender_ciphers, int receiver_cip
hers);
BOOL CRYPT_choose_ciphers(PTInstVar pvar);
#define CRYPT_get_sender_cipher(pvar) ((pvar)->crypt_state.sender_cipher)
#define CRYPT_get_receiver_cipher(pvar) ((pvar)->crypt_state.receiver_cipher)
int CRYPT_get_decryption_block_size(PTInstVar pvar);
int CRYPT_get_encryption_block_size(PTInstVar pvar);
#define CRYPT_encrypt(pvar, buf, bytes) \
    ((pvar)->crypt_state.encrypt((pvar), (buf), (bytes)))
#define CRYPT_decrypt(pvar, buf, bytes) \
    ((pvar)->crypt_state.decrypt((pvar), (buf), (bytes)))

BOOL CRYPT_detect_attack(PTInstVar pvar, unsigned char FAR * buf, int bytes);
int CRYPT_passphrase_decrypt(int cipher, char FAR * passphrase, char FAR * buf, int l
en);

#endif
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
This code is copyright (C) 1998-1999 Robert O'Callahan.
See LICENSE.TXT for the license.
*/

#include "ttxssh.h"
#include "util.h"
#include "resource.h"
#include "matcher.h"

#include "bn.h"

#include <fcntl.h>
#include <io.h>
#include <errno.h>
#include <sys/stat.h>

static char FAR * FAR * parse_multi_path(char FAR * buf) {
  int i;
  int ch;
  int num_paths = 1;
  char FAR * FAR * result;
  int last_path_index;

  for (i = 0; (ch = buf[i]) != 0; i++) {
    if (ch == ';') {
      num_paths++;
    }
  }

  result = (char FAR * FAR *)malloc(sizeof(char FAR *)*(num_paths + 1));

  last_path_index = 0;
```



```
  num_paths = 0;
  for (i = 0; (ch = buf[i]) != 0; i++) {
    if (ch == ';') {
      buf[i] = 0;
      result[num_paths] = _strdup(buf + last_path_index);
      num_paths++;
      buf[i] = ch;
      last_path_index = i + 1;
    }
  }
  if (i > last_path_index) {
    result[num_paths] = _strdup(buf + last_path_index);
    num_paths++;
  }
  result[num_paths] = NULL;
  return result;
}

void HOSTS_init(PTInstVar pvar) {
  pvar->hosts_state.prefetched_hostname = NULL;
  pvar->hosts_state.key_exp = NULL;
  pvar->hosts_state.key_mod = NULL;
  pvar->hosts_state.hosts_dialog = NULL;
  pvar->hosts_state.file_names = NULL;
}

void HOSTS_open(PTInstVar pvar) {
  pvar->hosts_state.file_names = parse_multi_path(pvar->session_settings.KnownHostsFi
les);
}

static int begin_read_file(PTInstVar pvar, char FAR * name, int suppress_errors) {
  int fd;
  int length;
  int amount_read;
  char buf[2048];

  get_teraterm_dir_relative_name(buf, sizeof(buf), name);
  fd = _open(buf, _O_RDONLY | _O_SEQUENTIAL | _O_BINARY);
  if (fd == -1) {
    if (!suppress_errors) {
      if (errno == ENOENT) {
        notify_nonfatal_error(pvar, "known_hostsファイルの読み込みエラーが発生しまし
た．\n"
          "指定されたファイルが見つかりませんでした．");
      } else {
        notify_nonfatal_error(pvar, "known_hostsファイルの読み込みエラーが発生しまし
た．");
      }
    }
    return 0;
  }

  length = (int)_lseek(fd, 0, SEEK_END);
  _lseek(fd, 0, SEEK_SET);

  if (length >= 0 && length < 0x7FFFFFFF) {
    pvar->hosts_state.file_data = malloc(length + 1);
    if (pvar->hosts_state.file_data == NULL) {
      if (!suppress_errors) {
        notify_nonfatal_error(pvar, "known_hostsファイルを読み込むのに必要なメモリを
確保できませんでした．");
      }
      _close(fd);
      return 0;
    }
```

```
  } else {
    if (!suppress_errors) {
      notify_nonfatal_error(pvar, "known_hostsファイルの読み込みエラーが発生しました
. ");
    }
    _close(fd);
    return 0;
  }

  amount_read = _read(fd, pvar->hosts_state.file_data, length);
  pvar->hosts_state.file_data[length] = 0;

  _close(fd);

  if (amount_read != length) {
    if (!suppress_errors) {
      notify_nonfatal_error(pvar, "known_hostsファイルの読み込みエラーが発生しました
. ");
    }
    free(pvar->hosts_state.file_data);
    pvar->hosts_state.file_data = NULL;
    return 0;
  } else {
    return 1;
  }
}

static int end_read_file(PTInstVar pvar, int suppress_errors) {
  free(pvar->hosts_state.file_data);
  pvar->hosts_state.file_data = NULL;
  return 1;
}

static int begin_read_host_files(PTInstVar pvar, int suppress_errors) {
  pvar->hosts_state.file_num = 0;
  pvar->hosts_state.file_data = NULL;
  return 1;
}

static int eat_spaces(char FAR * data) {
  int index = 0;
  int ch;

  while ((ch = data[index]) == ' ' || ch == '\t') {
    index++;
  }
  return index;
}

static int eat_digits(char FAR * data) {
  int index = 0;
  int ch;

  while ((ch = data[index]) >= '0' && ch <= '9') {
    index++;
  }
  return index;
}

static int eat_to_end_of_line(char FAR * data) {
  int index = 0;
  int ch;

  while ((ch = data[index]) != '\n' && ch != '\r' && ch != 0) {
    index++;
  }
```



```
  while ((ch = data[index]) == '\n' || ch == '\r') {
    index++;
  }

  return index;
}

static int eat_to_end_of_pattern(char FAR * data) {
  int index = 0;
  int ch;

  while (ch = data[index], is_pattern_char(ch)) {
    index++;
  }

  return index;
}

static char FAR * parse_bignum(char FAR * data) {
  uint32 digits = 0;
  BIGNUM * num = BN_new();
  BIGNUM * billion = BN_new();
  BIGNUM * digits_num = BN_new();
  char FAR * result;
  int ch;
  int leftover_digits = 1;
  BN_CTX *bn_ctx = BN_CTX_new();

  BN_set_word(num, 0);
  BN_set_word(billion, 1000000000L);

  BN_CTX_init( bn_ctx );

  while ((ch = *data) >= '0' && ch <= '9') {
    if (leftover_digits == 1000000000L) {
      BN_set_word(digits_num, digits);
      BN_mul(num, num, billion, bn_ctx);
      BN_add(num, num, digits_num);
      leftover_digits = 1;
      digits = 0;
    }

    digits = digits*10 + ch - '0';
    leftover_digits *= 10;
    data++;
  }

  BN_set_word(digits_num, digits);
  BN_set_word(billion, leftover_digits);
  BN_mul(num, num, billion, bn_ctx);
  BN_add(num, num, digits_num);

  result = (char FAR *)malloc(2 + BN_num_bytes(num));
  set_ushort16_MSBfirst(result, BN_num_bits(num));
  BN_bn2bin(num, result + 2);

  BN_free(digits_num);
  BN_free(num);
  BN_free(billion);

  return result;
}

static int check_host_key(PTInstVar pvar, char FAR * hostname,
                          char FAR * data) {
```

```
  int index = eat_spaces(data);
  int matched = 0;
  int keybits = 0;

  if (data[index] == '#') {
    return index + eat_to_end_of_line(data + index);
  }

  /* if we find an empty line, then it won't have any patterns matching the hostname
     and so we skip it */
  index--;
  do {
    int negated;

    index++;
    negated = data[index] == '!';

    if (negated) {
      index++;
      if (match_pattern(data + index, hostname)) {
        return index + eat_to_end_of_line(data + index);
      }
    } else if (match_pattern(data + index, hostname)) {
      matched = 1;
    }

    index += eat_to_end_of_pattern(data + index);
  } while (data[index] == ',');

  if (!matched) {
    return index + eat_to_end_of_line(data + index);
  } else {
    index += eat_spaces(data + index);

    pvar->hosts_state.key_bits = atoi(data + index);
    index += eat_digits(data + index);
    index += eat_spaces(data + index);

    pvar->hosts_state.key_exp = parse_bignum(data + index);
    index += eat_digits(data + index);
    index += eat_spaces(data + index);

    pvar->hosts_state.key_mod = parse_bignum(data + index);

    if (pvar->hosts_state.key_bits < 0 || pvar->hosts_state.key_exp == NULL
      || pvar->hosts_state.key_mod == NULL) {
      pvar->hosts_state.key_bits = 0;
      free(pvar->hosts_state.key_exp);
      free(pvar->hosts_state.key_mod);
    }

    return index + eat_to_end_of_line(data + index);
  }
}

static int read_host_key(PTInstVar pvar, char FAR * hostname, int suppress_errors) {
  int i;

  for (i = 0; hostname[i] != 0; i++) {
    int ch = hostname[i];

    if (!is_pattern_char(ch) || ch == '*' || ch == '?') {
      if (!suppress_errors) {
        notify_fatal_error(pvar, "ホスト名に無効な文字が含まれています．\n"
          "セッションを終了します．");
      }
```



```
      return 0;
    }
  }

  if (i == 0) {
    if (!suppress_errors) {
      notify_fatal_error(pvar, "ホスト名が空です．\n"
        "セッションを終了します．");
    }
    return 0;
  }

  pvar->hosts_state.key_bits = 0;
  free(pvar->hosts_state.key_exp);
  pvar->hosts_state.key_exp = NULL;
  free(pvar->hosts_state.key_mod);
  pvar->hosts_state.key_mod = NULL;

  do {
    if (pvar->hosts_state.file_data == NULL
      || pvar->hosts_state.file_data[pvar->hosts_state.file_data_index] == 0) {
      char FAR * filename;
      int keep_going = 1;

      if (pvar->hosts_state.file_data != NULL) {
        end_read_file(pvar, suppress_errors);
      }

      do {
        filename = pvar->hosts_state.file_names[pvar->hosts_state.file_num];

        if (filename == NULL) {
          return 1;
        } else {
          pvar->hosts_state.file_num++;

          if (filename[0] != 0) {
            if (begin_read_file(pvar, filename, suppress_errors)) {
              pvar->hosts_state.file_data_index = 0;
              keep_going = 0;
            }
          }
        }
      } while (keep_going);
    }

    pvar->hosts_state.file_data_index +=
      check_host_key(pvar, hostname, pvar->hosts_state.file_data + pvar->hosts_state.
file_data_index);
  } while (pvar->hosts_state.key_bits == 0);

  return 1;
}

static void finish_read_host_files(PTInstVar pvar, int suppress_errors) {
  if (pvar->hosts_state.file_data != NULL) {
    end_read_file(pvar, suppress_errors);
  }
}

void HOSTS_prefetch_host_key(PTInstVar pvar, char FAR * hostname) {
  if (!begin_read_host_files(pvar, 1)) {
    return;
  }

  if (!read_host_key(pvar, hostname, 1)) {
```

```
    return;
  }

  free(pvar->hosts_state.prefetched_hostname);
  pvar->hosts_state.prefetched_hostname = _strdup(hostname);

  finish_read_host_files(pvar, 1);
}

static BOOL equal_mp_ints(unsigned char FAR * num1, unsigned char FAR * num2) {
  if (num1 == NULL || num2 == NULL) {
    return FALSE;
  } else {
    uint32 bytes = (get_ushort16_MSBfirst(num1) + 7)/8;

    if (bytes != (get_ushort16_MSBfirst(num2) + 7)/8) {
      return FALSE; /* different byte lengths */
    } else {
      return memcmp(num1 + 2, num2 + 2, bytes) == 0;
    }
  }
}

static BOOL match_key(PTInstVar pvar,
                      int bits, unsigned char FAR * exp, unsigned char FAR * mod) {
  /* just check for equal exponent and modulus */
  return equal_mp_ints(exp, pvar->hosts_state.key_exp)
    && equal_mp_ints(mod, pvar->hosts_state.key_mod);
}

static void init_hosts_dlg(PTInstVar pvar, HWND dlg) {
  char buf[1024];
  char buf2[2048];
  int i, j;
  int ch;

  GetDlgItemText(dlg, IDC_HOSTWARNING, buf, sizeof(buf));
  for (i = 0; (ch = buf[i]) != 0 && ch != '#'; i++) {
    buf2[i] = ch;
  }
  if (sizeof(buf2) - i - 1 > 0) {
    strncpy(buf2 + i, pvar->hosts_state.prefetched_hostname, sizeof(buf2) - i - 1);
  }
  j = i + strlen(buf2 + i);
  for (; buf[i] == '#'; i++) {
  }
  if (sizeof(buf2) - j - 1 > 0) {
    strncpy(buf2 + j, buf + i, sizeof(buf2) - j - 1);
  }
  buf2[sizeof(buf2) - 1] = 0;

  SetDlgItemText(dlg, IDC_HOSTWARNING, buf2);
}

static int print_mp_int(char FAR * buf, unsigned char FAR * mp) {
  int i = 0, j, k;
  BIGNUM * num = BN_new();
  int ch;

  BN_bin2bn(mp + 2, (get_ushort16_MSBfirst(mp)+7)/8, num);

  do {
    buf[i] = (char)((BN_div_word(num, 10)) + '0');
    i++;
  } while (!BN_is_zero(num));
```



```
  /* we need to reverse the digits */
  for (j = 0, k = i - 1; j < k; j++, k--) {
    ch = buf[j];
    buf[j] = buf[k];
    buf[k] = ch;
  }

  buf[i] = 0;
  return i;
}

static char FAR * format_host_key(PTInstVar pvar) {
  int host_len = strlen(pvar->hosts_state.prefetched_hostname);
  char FAR * result = (char FAR *)malloc(host_len
    + 50 + get_ushort16_MSBfirst(pvar->hosts_state.key_exp)/3
    + get_ushort16_MSBfirst(pvar->hosts_state.key_mod)/3);
  int index;

  strcpy(result, pvar->hosts_state.prefetched_hostname);
  index = host_len;

  sprintf(result + index, " %d ", pvar->hosts_state.key_bits);
  index += strlen(result + index);
  index += print_mp_int(result + index, pvar->hosts_state.key_exp);
  result[index] = ' ';
  index++;
  index += print_mp_int(result + index, pvar->hosts_state.key_mod);
  strcpy(result + index, " \r\n");

  return result;
}

static void add_host_key(PTInstVar pvar) {
  char FAR * name = pvar->hosts_state.file_names[0];

  if (name == NULL || name[0] == 0) {
    notify_nonfatal_error(pvar, "known-hostsファイルが指定されていないため、ホストと
鍵の組を追加できませんでした．\n"
      "Teratermを再起動して、TTSSH設定ダイアログボックスから読み書き可能known-hostsフ
ァイルを指定して下さい．");
  } else {
    char FAR * keydata = format_host_key(pvar);
    int length = strlen(keydata);
    int fd = _open(name, _O_APPEND | _O_CREAT | _O_WRONLY | _O_SEQUENTIAL | _O_BINARY
,
      _S_IREAD | _S_IWRITE);
    int amount_written;
    int close_result;

    if (fd == -1) {
      if (errno == EACCES) {
        notify_nonfatal_error(pvar, "ホスト鍵の書き込み中にエラーが発生しました．\n"
          "known_hostsファイルへの書き込み権限がありません．");
      } else {
        notify_nonfatal_error(pvar, "ホスト鍵の書き込み中にエラーが発生しました．\n"
          "ホスト鍵を書き込めませんでした．");
      }
      return;
    }

    amount_written = _write(fd, keydata, length);
    free(keydata);
    close_result = _close(fd);

    if (amount_written != length || close_result == -1) {
      notify_nonfatal_error(pvar, "ホスト鍵の書き込み中にエラーが発生しました．\n"
```

```
        "ホスト鍵を書き込めませんでした．");
    }
  }
}

static BOOL CALLBACK hosts_dlg_proc(HWND dlg, UINT msg, WPARAM wParam, LPARAM lParam)
 {
  PTInstVar pvar;

  switch (msg) {
  case WM_INITDIALOG:
    pvar = (PTInstVar)lParam;
    pvar->hosts_state.hosts_dialog = dlg;
    SetWindowLong(dlg, DWL_USER, lParam);

    init_hosts_dlg(pvar, dlg);
    return TRUE; /* because we do not set the focus */

  case WM_COMMAND:
    pvar = (PTInstVar)GetWindowLong(dlg, DWL_USER);

    switch (LOWORD(wParam)) {
    case IDC_CONTINUE:
      if (IsDlgButtonChecked(dlg, IDC_ADDTOKNOWNHOSTS)) {
        add_host_key(pvar);
      }
      SSH_notify_host_OK(pvar);

      pvar->hosts_state.hosts_dialog = NULL;

      EndDialog(dlg, 1);
      return TRUE;

    case IDCANCEL: /* kill the connection */
      pvar->hosts_state.hosts_dialog = NULL;
      notify_closed_connection(pvar);
      EndDialog(dlg, 0);
      return TRUE;

    default:
      return FALSE;
    }

  default:
    return FALSE;
  }
}

static char FAR * copy_mp_int(char FAR * num) {
  int len = (get_ushort16_MSBfirst(num) + 7)/8 + 2;
  char FAR * result = (char FAR *)malloc(len);

  if (result != NULL) {
    memcpy(result, num, len);
  }

  return result;
}

void HOSTS_do_unknown_host_dialog(HWND wnd, PTInstVar pvar) {
  if (pvar->hosts_state.hosts_dialog == NULL) {
    HWND cur_active = GetActiveWindow();

    DialogBoxParam(hInst, MAKEINTRESOURCE(IDD_SSHUNKNOWNHOST),
      cur_active != NULL ? cur_active : wnd, hosts_dlg_proc, (LPARAM)pvar);
  }
```



```
}

void HOSTS_do_different_host_dialog(HWND wnd, PTInstVar pvar) {
  if (pvar->hosts_state.hosts_dialog == NULL) {
    HWND cur_active = GetActiveWindow();

    DialogBoxParam(hInst, MAKEINTRESOURCE(IDD_SSHDIFFERENTHOST),
      cur_active != NULL ? cur_active : wnd, hosts_dlg_proc, (LPARAM)pvar);
  }
}

BOOL HOSTS_check_host_key(PTInstVar pvar, char FAR * hostname,
                          int bits, unsigned char FAR * exp, unsigned char FAR * mod)
{
  int found_different_key = 0;

  if (pvar->hosts_state.prefetched_hostname != NULL
    && stricmp(pvar->hosts_state.prefetched_hostname, hostname) == 0
    && match_key(pvar, bits, exp, mod)) {
    SSH_notify_host_OK(pvar);
    return TRUE;
  }

  if (begin_read_host_files(pvar, 0)) {
    do {
      if (!read_host_key(pvar, hostname, 0)) {
        break;
      }

      if (pvar->hosts_state.key_bits > 0) {
        if (match_key(pvar, bits, exp, mod)) {
          finish_read_host_files(pvar, 0);
          SSH_notify_host_OK(pvar);
          return TRUE;
        } else {
          found_different_key = 1;
        }
      }
    } while (pvar->hosts_state.key_bits > 0);

    finish_read_host_files(pvar, 0);
  }

  pvar->hosts_state.key_bits = bits;
  pvar->hosts_state.key_exp = copy_mp_int(exp);
  pvar->hosts_state.key_mod = copy_mp_int(mod);
  free(pvar->hosts_state.prefetched_hostname);
  pvar->hosts_state.prefetched_hostname = _strdup(hostname);

  if (found_different_key) {
    PostMessage(pvar->NotificationWindow, WM_COMMAND, ID_SSHDIFFERENTHOST, 0);
  } else {
    PostMessage(pvar->NotificationWindow, WM_COMMAND, ID_SSHUNKNOWNHOST, 0);
  }

  return TRUE;
}

void HOSTS_notify_disconnecting(PTInstVar pvar) {
  if (pvar->hosts_state.hosts_dialog != NULL) {
    PostMessage(pvar->hosts_state.hosts_dialog, WM_COMMAND, IDCANCEL, 0);
    /* the main window might not go away if it's not enabled. (see vtwin.cpp) */
    EnableWindow(pvar->NotificationWindow, TRUE);
  }
}
```

```
void HOSTS_end(PTInstVar pvar) {
  int i;

  free(pvar->hosts_state.prefetched_hostname);
  free(pvar->hosts_state.key_exp);
  free(pvar->hosts_state.key_mod);

  if (pvar->hosts_state.file_names != NULL) {
    for (i = 0; pvar->hosts_state.file_names[i] != NULL; i++) {
      free(pvar->hosts_state.file_names[i]);
    }
    free(pvar->hosts_state.file_names);
  }
}
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
This code is copyright (C) 1998-1999 Robert O'Callahan.
See LICENSE.TXT for the license.
*/

#ifndef __HOSTS_H
#define __HOSTS_H

typedef struct {
  char FAR * prefetched_hostname;
  int key_bits;
  /* The key exponent and modulus, in SSH mp_int format */
  unsigned char FAR * key_exp;
  unsigned char FAR * key_mod;

  int file_num;
  char FAR * FAR * file_names;
  int file_data_index;
  char FAR * file_data;

  HWND hosts_dialog;
} HOSTSState;

void HOSTS_init(PTInstVar pvar);
void HOSTS_open(PTInstVar pvar);
void HOSTS_prefetch_host_key(PTInstVar pvar, char FAR * hostname);
BOOL HOSTS_check_host_key(PTInstVar pvar, char FAR * hostname,
                          int bits, unsigned char FAR * exp, unsigned char FAR * mod);
void HOSTS_do_unknown_host_dialog(HWND wnd, PTInstVar pvar);
void HOSTS_do_different_host_dialog(HWND wnd, PTInstVar pvar);
void HOSTS_notify_disconnecting(PTInstVar pvar);
void HOSTS_end(PTInstVar pvar);
```



```
#endif
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
This code is copyright (C) 1998-1999 Robert O'Callahan.
See LICENSE.TXT for the license.
*/

#include "ttxssh.h"

#include <io.h>
#include <fcntl.h>
#include <stdlib.h>
#include <errno.h>
#include <rsa.h>

static char ID_string[] = "SSH PRIVATE KEY FILE FORMAT 1.1\n";

static BIGNUM FAR * get_bignum(unsigned char FAR * bytes) {
  int bits = get_ushort16_MSBfirst(bytes);

  return BN_bin2bn(bytes + 2, (bits + 7)/8, NULL);
}

/* normalize the RSA key by precomputing various bits of it.
   This code is adapted from libeay's rsa_gen.c
   It's needed to work around "issues" with LIBEAY/RSAREF.
   If this function returns 0, then something went wrong and the
   key must be discarded. */
static BOOL normalize_key(RSA FAR * key) {
  BOOL OK = FALSE;
        BIGNUM * r = BN_new();
  BN_CTX * ctx = BN_CTX_new();

        if (BN_cmp(key->p, key->q) < 0) {
                BIGNUM * tmp = key->p;
```



```
                key->p = key->q;
                key->q = tmp;
        }

  if (r != NULL && ctx != NULL) {
          key->dmp1 = BN_new();
          key->dmq1 = BN_new();
          key->iqmp = BN_mod_inverse(NULL, key->q, key->p, ctx);

    if (key->dmp1 != NULL && key->dmq1 != NULL && key->iqmp != NULL) {
          OK = BN_sub(r, key->p, BN_value_one())
        && BN_mod(key->dmp1, key->d, r, ctx)
        && BN_sub(r, key->q, BN_value_one())
              && BN_mod(key->dmq1, key->d, r, ctx);
    }
  }

  BN_free(r);
        BN_CTX_free(ctx);

  return OK;
}

static RSA FAR * read_RSA_private_key(PTInstVar pvar,
  char FAR * relative_name, char FAR * passphrase,
  BOOL FAR * invalid_passphrase, BOOL is_auto_login) {
  char filename[2048];
  int fd;
  unsigned int length, amount_read;
  unsigned char * keyfile_data;
  unsigned int index;
  int cipher;
  RSA FAR * key;
  unsigned int E_index, N_index, D_index, U_index, P_index, Q_index = 0;

  *invalid_passphrase = FALSE;

  get_teraterm_dir_relative_name(filename, sizeof(filename), relative_name);

  fd = _open(filename, _O_RDONLY | _O_SEQUENTIAL | _O_BINARY);
  if (fd == -1) {
    if (errno == ENOENT) {
      notify_nonfatal_error(pvar, "鍵ファイルの読み込み中にエラーが発生しました．\n"
        "指定されたファイルは見つかりませんでした．");
    } else {
      notify_nonfatal_error(pvar, "鍵ファイルの読み込み中にエラーが発生しました．");
    }
    return NULL;
  }

  length = (int)_lseek(fd, 0, SEEK_END);
  _lseek(fd, 0, SEEK_SET);

  if (length >= 0 && length < 0x7FFFFFFF) {
    keyfile_data = malloc(length + 1);
    if (keyfile_data == NULL) {
      notify_nonfatal_error(pvar, "鍵ファイルを読み込むのに必要なメモリを確保できませ
んでした．");
      _close(fd);
      return NULL;
    }
  } else {
    notify_nonfatal_error(pvar, "鍵ファイルの読み込み中にエラーが発生しました．");
    _close(fd);
    return NULL;
```

```
  }

  amount_read = _read(fd, keyfile_data, length);
  /* terminate it with 0 so that the strncmp below is guaranteed not to
     crash */
  keyfile_data[length] = 0;

  _close(fd);

  if (amount_read != length) {
    notify_nonfatal_error(pvar, "鍵ファイルの読み込み中にエラーが発生しました．");
    free(keyfile_data);
    return NULL;
  }

  if (strcmp(keyfile_data, ID_string) != 0) {
    notify_nonfatal_error(pvar, "指定された鍵ファイルはSSH個人鍵ではありません．");
    free(keyfile_data);
    return NULL;
  }

  index = NUM_ELEM(ID_string);

  if (length < index + 9) {
    notify_nonfatal_error(pvar, "指定された鍵ファイルは途切れており、有効な鍵ファイル
ではありません．");
    free(keyfile_data);
    return NULL;
  }

  cipher = keyfile_data[index];
  /* skip reserved int32, public key bits int32 */
  index += 9;
  /* skip public key e and n mp_ints */
  if (length < index + 2) {
    notify_nonfatal_error(pvar, "指定された鍵ファイルは途切れており、有効な鍵ファイル
ではありません．");
    free(keyfile_data);
    return NULL;
  }
  N_index = index;
  index += (get_ushort16_MSBfirst(keyfile_data + index) + 7)/8 + 2;
  if (length < index + 2) {
    notify_nonfatal_error(pvar, "指定された鍵ファイルは途切れており、有効な鍵ファイル
ではありません．");
    free(keyfile_data);
    return NULL;
  }
  E_index = index;
  index += (get_ushort16_MSBfirst(keyfile_data + index) + 7)/8 + 2;
  /* skip comment */
  if (length < index + 4) {
    notify_nonfatal_error(pvar, "指定された鍵ファイルは途切れており、有効な鍵ファイル
ではありません．");
    free(keyfile_data);
    return NULL;
  }
  index += get_uint32_MSBfirst(keyfile_data + index) + 4;

  if (length < index + 6) {
    notify_nonfatal_error(pvar, "指定された鍵ファイルは途切れており、有効な鍵ファイル
ではありません．");
    free(keyfile_data);
    return NULL;
  }
  if (cipher != SSH_CIPHER_NONE) {
```



```
    if ((length - index) % 8 != 0) {
      notify_nonfatal_error(pvar, "指定された鍵ファイルをパスフレーズで復号化できませ
ん．\n"
        "鍵ファイルの大きさが不正です．");
      free(keyfile_data);
      return NULL;
    }
    if (!CRYPT_passphrase_decrypt(cipher, passphrase, keyfile_data + index, length -
index)) {
      notify_nonfatal_error(pvar, "指定された鍵ファイルをパスフレーズで復号化できませ
ん．\n"
        "暗号化に使用されているアルゴリズムをサポートしていません．");
      free(keyfile_data);
      return NULL;
    }
  }

  if (keyfile_data[index] != keyfile_data[index + 2]
    || keyfile_data[index + 1] != keyfile_data[index + 3]) {
    *invalid_passphrase = TRUE;
    notify_nonfatal_error(pvar, is_auto_login
      ? "指定されている鍵ファイルは空パスフレーズで復号できませんでした．\n"
      "自動ログインを行うには，空のパスフレーズで暗号化した鍵ファイルが必要です．\n"
      "注意！: この場合，鍵は容易に盗まれる可能性がありますので，扱いには充分な注意が
必要です．"
      : "指定されている鍵ファイルはパスフレーズで復号できませんでした．\n"
      "パスフレーズが違います．");
    memset(keyfile_data, 0, length);
    free(keyfile_data);
    return NULL;
  }
  index += 4;

  D_index = index;
  if (length >= D_index + 2) {
    U_index = D_index + (get_ushort16_MSBfirst(keyfile_data + D_index) + 7)/8 + 2;
    if (length >= U_index + 2) {
      P_index = U_index + (get_ushort16_MSBfirst(keyfile_data + U_index) + 7)/8 + 2;
      if (length >= P_index + 2) {
        Q_index = P_index + (get_ushort16_MSBfirst(keyfile_data + P_index) + 7)/8 + 2
;
      }
    }
  }
  if (Q_index == 0 || length < Q_index + (get_ushort16_MSBfirst(keyfile_data + Q_inde
x) + 7)/8 + 2) {
    notify_nonfatal_error(pvar, "指定された鍵ファイルは途切れており、有効な鍵ファイル
ではありません．");
    memset(keyfile_data, 0, length);
    free(keyfile_data);
    return NULL;
  }

  key = RSA_new();
  key->n = get_bignum(keyfile_data + N_index);
  key->e = get_bignum(keyfile_data + E_index);
  key->d = get_bignum(keyfile_data + D_index);
  key->p = get_bignum(keyfile_data + P_index);
  key->q = get_bignum(keyfile_data + Q_index);

  if (!normalize_key(key)) {
    notify_nonfatal_error(pvar, "暗号ライブラリでエラーが発生しました．\n"
      "格納されている鍵情報が正しくない可能性があります．");

    RSA_free(key);
    key = NULL;
```

```
  }

  memset(keyfile_data, 0, length);
  free(keyfile_data);
  return key;
}

CRYPTKeyPair FAR * KEYFILES_read_private_key(PTInstVar pvar, char FAR * relative_name
,
  char FAR * passphrase, BOOL FAR * invalid_passphrase,
  BOOL is_auto_login) {
  RSA FAR * RSA_key = read_RSA_private_key(pvar, relative_name,
    passphrase, invalid_passphrase, is_auto_login);

  if (RSA_key == NULL) {
    return FALSE;
  } else {
    CRYPTKeyPair FAR * result = (CRYPTKeyPair FAR *)malloc(sizeof(CRYPTKeyPair));

    result->RSA_key = RSA_key;
    return result;
  }
}
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
This code is copyright (C) 1998-1999 Robert O'Callahan.
See LICENSE.TXT for the license.
*/

#ifndef __KEYFILES_H
#define __KEYFILES_H

CRYPTKeyPair FAR * KEYFILES_read_private_key(PTInstVar pvar, char FAR * relative_name
,
  char FAR * passphrase, BOOL FAR * invalid_passphrase,
  BOOL is_auto_login);

#endif
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
This code is copyright (C) 1998-1999 Robert O'Callahan.
See LICENSE.TXT for the license.
*/

#include <windows.h>
#include <memory.h>

#include "matcher.h"

/* This is the "slow" wildcard matcher, but at least it is worst-case quadratic
   in the length of the string (as opposed to exponential).
   Since it's also best-case quadratic, it should only be used for unusual
   cases (2 or more *'s in the pattern). */
static BOOL wildcard_match_pattern(char const FAR * pattern, int patternLen,
                                   char const FAR * str, int strLen) {
  char * canBeAt = (char *)malloc(sizeof(char)*(strLen + 1));
  int i, result;

  canBeAt[0] = 1;
  memset(canBeAt + 1, 0, strLen);

  for (; patternLen > 0; patternLen--, pattern++) {
    char patternCh = *pattern;

    if (patternCh == '*') {
      char * firstPossible = (char *)memchr(canBeAt, 1, strLen + 1);

      if (firstPossible != NULL) {
        memset(firstPossible + 1, 1, canBeAt + strLen - firstPossible);
      }
    } else if (patternCh == '?') {
      memmove(canBeAt + 1, canBeAt, strLen);
```



```
      canBeAt[0] = 0;
    } else {
      for (i = strLen; i > 0; i--) {
        canBeAt[i] = canBeAt[i - 1] && str[i - 1] == patternCh;
      }
      canBeAt[0] = 0;
    }
  }

  result = canBeAt[strLen];

  free(canBeAt);

  return result;
}

/*
Before entry, ensure that str does not contain any invalid characters
(at least including '*', '?', ',', ' ' and '\n').
*/
BOOL match_pattern(char const FAR * pattern, char const FAR * str) {
  char patternCh = *pattern;
  char strCh = *str;

  while (strCh != 0 && (strCh == patternCh || patternCh == '?')) {
    str++;
    pattern++;
    strCh = *str;
    patternCh = *pattern;
  }

  if (strCh == 0) {
    while (patternCh == '*') {
      pattern++;
      patternCh = *pattern;
    }

    return !is_pattern_char(patternCh);
  } else if (patternCh == '*') {
    char nextPatternCh = pattern[1];

    if (!is_pattern_char(nextPatternCh)) {
      return TRUE;
    } else {
      int minStrLen = 0;
      int patternLen = 1;
      int strLen;

      patternCh = pattern[patternLen];
      while (is_pattern_char(patternCh)) {
        if (patternCh != '*') {
          minStrLen++;
        }
        patternLen++;
        patternCh = pattern[patternLen];
      }

      strLen = strlen(str);
      if (strLen < minStrLen) {
        return FALSE;
      } else {
        patternCh = pattern[patternLen - 1];
        strCh = str[strLen - 1];
        while (patternLen > 1 && (strCh == patternCh || patternCh == '?')) {
          strLen--;
          patternLen--;
```

```
          strCh = str[strLen - 1];
          patternCh = pattern[patternLen - 1];
        }

        if (patternLen == 1) { /* single * matches whatever's left in str */
          return TRUE;
        } else if (patternCh != '*') {
          return FALSE;
        } else {
          return wildcard_match_pattern(pattern, patternLen, str, strLen);
        }
      }
    }
  } else {
    return FALSE;
  }
}
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
This code is copyright (C) 1998-1999 Robert O'Callahan.
See LICENSE.TXT for the license.
*/

#ifndef __MATCHER_H
#define __MATCHER_H

#define is_pattern_char(ch) ((ch) != 0 && (ch) != ' ' && (ch) != ',' && (ch) != '\n'
&& (ch) != '\r' && (ch) != '\t')

/*
Matches a string against a glob pattern. The pattern is a string of
pattern characters (everything except 0, space, comma, \n, \r, and \t)
followed by one or more non-pattern characters. ? and * in the pattern
are metacharacters interpreted in the normal way.

Before entry, ensure that str does not contain any non-pattern characters
and that it does not contain any pattern metacharacters.

This function is optimized for matching many patterns against one string,
such as occurs when searching an ssh_known_hosts file for a host key.
*/
BOOL match_pattern(char const FAR * pattern, char const FAR * str);

#endif
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
This code is copyright (C) 1998-1999 Robert O'Callahan.
See LICENSE.TXT for the license.
*/

#include "ttxssh.h"
#include "util.h"

#define READAMOUNT 60000

void PKT_init(PTInstVar pvar) {
  buf_create(&pvar->pkt_state.buf, &pvar->pkt_state.buflen);
  pvar->pkt_state.datastart = 0;
  pvar->pkt_state.datalen = 0;
  pvar->pkt_state.seen_server_ID = FALSE;
  pvar->pkt_state.seen_newline = FALSE;
  pvar->pkt_state.predecrypted_packet = FALSE;
}

/* Read some data, leave no more than up_to_amount bytes in the buffer,
   return the number of bytes read or -1 on error or blocking. */
static int recv_data(PTInstVar pvar, unsigned long up_to_amount) {
  int amount_read;

  /* Shuffle data to the start of the buffer */
  if (pvar->pkt_state.datastart != 0) {
    memmove(pvar->pkt_state.buf,
      pvar->pkt_state.buf + pvar->pkt_state.datastart,
      pvar->pkt_state.datalen);
    pvar->pkt_state.datastart = 0;
  }

  buf_ensure_size(&pvar->pkt_state.buf, &pvar->pkt_state.buflen, up_to_amount);
```



```
  amount_read = recv(
    pvar->socket,
    pvar->pkt_state.buf + pvar->pkt_state.datalen,
    up_to_amount - pvar->pkt_state.datalen, 0);

  if (amount_read > 0) {
    /* Update seen_newline if necessary */
    if (!pvar->pkt_state.seen_server_ID && !pvar->pkt_state.seen_newline) {
      int i;

      for (i = 0; i < amount_read; i++) {
        if (pvar->pkt_state.buf[pvar->pkt_state.datalen + i] == '\n') {
          pvar->pkt_state.seen_newline = 1;
        }
      }
    }

    pvar->pkt_state.datalen += amount_read;
  }

  return amount_read;
}

/* This function does two things:
   -- reads data from the sshd and feeds the SSH protocol packets to ssh.c
   -- copies any available decrypted session data into the application buffer
*/
int PKT_recv(PTInstVar pvar, char FAR * buf, int buflen) {
  int amount_in_buf = 0;
  BOOL connection_closed = FALSE;

  while (SSH_is_any_payload(pvar) ? buflen > 0 : !connection_closed) {
    if (SSH_is_any_payload(pvar)) {
      /* ssh.c has some session data for us to give to Teraterm. */
      int grabbed = SSH_extract_payload(pvar, buf, buflen);

      amount_in_buf += grabbed;
      buf += grabbed;
      buflen -= grabbed;
    } else if (!pvar->pkt_state.seen_server_ID &&
      (pvar->pkt_state.seen_newline || pvar->pkt_state.datalen >= 255)) {
      /* We're looking for the initial ID string and either we've seen the
         terminating newline, or we've exceeded the limit at which we should see
         a newline. */
      unsigned int i;

      for (i = 0; pvar->pkt_state.buf[i] != '\n' && i < pvar->pkt_state.datalen; i++)
 {
      }
      if (pvar->pkt_state.buf[i] == '\n') {
        i++;
      }

      if (SSH_handle_server_ID(pvar, pvar->pkt_state.buf, i)) {
        pvar->pkt_state.seen_server_ID = 1;
      }

      pvar->pkt_state.datastart += i;
      pvar->pkt_state.datalen -= i;
    } else if (pvar->pkt_state.seen_server_ID
      && pvar->pkt_state.datalen >= (unsigned int)SSH_get_min_packet_size(pvar)) {
      char FAR * data = pvar->pkt_state.buf + pvar->pkt_state.datastart;
      uint32 padding;
      uint32 pktsize;
      uint32 total_packet_size;
```

```
      if (!pvar->pkt_state.predecrypted_packet) {
        SSH_predecrpyt_packet(pvar, data);
        pvar->pkt_state.predecrypted_packet = TRUE;
      }

      if (SSHv1(pvar)) {
        uint32 realpktsize = get_uint32_MSBfirst(data);

        padding = 8 - (realpktsize % 8);
        pktsize = realpktsize + padding;
      } else {
        pktsize = get_uint32_MSBfirst(data);
        padding = (unsigned char)data[4];
      }

      total_packet_size = pktsize + 4 + SSH_get_clear_MAC_size(pvar);

      if (total_packet_size <= pvar->pkt_state.datalen) {
        /* the data must be 4 byte aligned. */
        SSH_handle_packet(pvar, data, pktsize, padding);
        pvar->pkt_state.predecrypted_packet = FALSE;

        pvar->pkt_state.datastart += total_packet_size;
        pvar->pkt_state.datalen -= total_packet_size;
      } else if (total_packet_size > 4*1024*1024) {
        notify_fatal_error(pvar, "サイズが大きすぎるパケットを受信したため、接続を閉
じます．");
      } else {
        int amount_read = recv_data(pvar, max(total_packet_size, READAMOUNT));

        if (amount_read == SOCKET_ERROR) {
          if (amount_in_buf == 0) {
            return SOCKET_ERROR;
          } else {
            return amount_in_buf;
          }
        } else {
          if (amount_read == 0) {
            connection_closed = TRUE;
          }
        }
      }
    } else {
      int amount_read = recv_data(pvar, READAMOUNT);

      if (amount_read == SOCKET_ERROR) {
        if (amount_in_buf == 0) {
          return SOCKET_ERROR;
        } else {
          return amount_in_buf;
        }
      } else if (amount_read == 0) {
        connection_closed = TRUE;
      }
    }

    if (pvar->fatal_error) {
      return amount_in_buf;
    }
  }

  if (SSH_is_any_payload(pvar)) {

    PostMessage(pvar->NotificationWindow, WM_USER_COMMNOTIFY,
      pvar->socket, MAKELPARAM(FD_READ, 0));
```



```
  }

  return amount_in_buf;
}

void PKT_end(PTInstVar pvar) {
  buf_destroy(&pvar->pkt_state.buf, &pvar->pkt_state.buflen);
}
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
This code is copyright (C) 1998-1999 Robert O'Callahan.
See LICENSE.TXT for the license.
*/

#ifndef __PKT_H
#define __PKT_H

typedef struct {
  char FAR * buf;
  unsigned long buflen;
  unsigned long datastart;
  unsigned long datalen;
  BOOL seen_server_ID;
  BOOL seen_newline;
  BOOL predecrypted_packet;
} PKTState;

void PKT_init(PTInstVar pvar);
int PKT_recv(PTInstVar pvar, char FAR * buf, int buflen);
void PKT_end(PTInstVar pvar);

#endif
```

```
//
// REQLIST.C -- Linked list of active requests
//

#include "ftpwrap.h"

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

//
// Head of linked list
// Internal linkage
//
static PTInstVar lpHead = NULL;
static PREQUEST lpFTPHead = NULL;

//
// Internal function
//
static void FreeRequest(PTInstVar lpReq);
static void FTPFreeRequest(PREQUEST lpReq);

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


PTInstVar       GetNewRequest(VOID)
{
        PTInstVar lpReq = NULL;
        PTInstVar lpThis = NULL;

        //
        // Allocate memory from the heap for the new
        // REQUEST structure
        //
        lpThis = (PTInstVar)malloc(sizeof(TInstVar));
        if (lpThis == NULL)
                return(NULL);

        // If the list is empty
        if (lpHead == NULL)
        {
                // Put this one at the head
                lpHead = lpThis;
    }
        else
        {
                // Walk to the end of the list
                lpReq = lpHead;
                while(lpReq->lpNext)
                        lpReq = lpReq->lpNext;

                // And append this one to the end
                lpReq->lpNext = lpThis;
    }

        //
        // Fill in the REQUEST structure
        //
        ZeroMemory( lpThis, sizeof(TInstVar) );
        lpThis->lpNext = NULL;
        // And return a pointer to it
        return (lpThis);
}

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

//
```



```
// GetRequest()
//
// Returns the REQUEST that Socket belongs to
//
PTInstVar GetRequest(SOCKET socket)
{
        PTInstVar lpReq;

        //
        // Walk through the list looking
        // for this socket
        //
        lpReq = lpHead;
        while(lpReq != NULL)
        {
                if (lpReq->socket == socket)
                        break;
                lpReq = lpReq->lpNext;
        }

        return(lpReq);
}

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

//
// DelRequest()
//
// Delete the REQUEST from the list
// and free associated memory
//
void DelRequest(PTInstVar lpThis)
{
        PTInstVar lpReq;
        BOOL bRet = FALSE;

        //
        // Search for "this"
        //

        // If This is the first member
        if (lpThis == lpHead)
        {
                // This->Next becomes the head
                lpHead = lpThis->lpNext;
        }
        else
        {
                // Search the list
                for (lpReq = lpHead; lpReq; lpReq = lpReq->lpNext)
                {
                        // If This is next...
                        if (lpReq->lpNext == lpThis)
                        {
                                // Set the pointer to skip "this"
                                lpReq->lpNext = lpThis->lpNext;
                                break;
                        }
                }
        }
        // free the memory
        FreeRequest(lpThis);
}

////////////////////////////////////////////////////////
```

```
//
// GetFirstRequest()
//
// Return the head of the list
// even if it is NULL
//
PTInstVar       GetFirstRequest(void)
{
        return(lpHead);
}

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

//
// GetNextRequest()
//
// Return the next REQUEST after This
//
PTInstVar       GetNextRequest(PTInstVar lpThis)
{
        return(lpThis->lpNext);
}

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

//
// DelAllRequests()
//
// Delete all of the REQUESTs in the list
// and free all memory
//
void DelAllRequests(void)
{
        PTInstVar lpReq;
        PTInstVar lpNext;

        //
        // Walk through the list
        // freeing memory
        //
        for (lpReq = lpHead; lpReq; )
        {
                lpNext = lpReq->lpNext;
                FreeRequest(lpReq);
                lpReq = lpNext;
        }
        lpHead = NULL;
}

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

static void FreeRequest(PTInstVar lpReq)
{
        free(lpReq);
}



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


PREQUEST        FTPGetNewRequest(VOID)
{
        PREQUEST lpReq = NULL;
```



```
        PREQUEST lpThis = NULL;

        //
        // Allocate memory from the heap for the new
        // REQUEST structure
        //
        lpThis = (PREQUEST)malloc(sizeof(REQUEST));
        if (lpThis == NULL)
                return(NULL);

        // If the list is empty
        if (lpFTPHead == NULL)
        {
                // Put this one at the head
                lpFTPHead = lpThis;
    }
        else
        {
                // Walk to the end of the list
                lpReq = lpFTPHead;
                while(lpReq->lpNext)
                        lpReq = lpReq->lpNext;

                // And append this one to the end
                lpReq->lpNext = lpThis;
    }

        //
        // Fill in the REQUEST structure
        //
        ZeroMemory( lpThis, sizeof(REQUEST) );
        lpThis->RdDataSize = 0;
        lpThis->WrDataSize = 0;
        lpThis->lpNext = NULL;
        lpThis->lastremoteport = -1;
        lpThis->RdBuff = (unsigned char *)malloc(FTPRDBUFFSIZE);
        if (lpThis->RdBuff == NULL) {
                free(lpThis);
                return(NULL);
        }
        lpThis->WrBuff = (unsigned char *)malloc(FTPWRBUFFSIZE);
        if (lpThis->WrBuff == NULL) {
                free(lpThis->RdBuff);
                free(lpThis);
                return(NULL);
        }
        lpThis->SSHRdBuff = (unsigned char *)malloc(SSHRDBUFFSIZE);
        if (lpThis->SSHRdBuff == NULL) {
                free(lpThis->RdBuff);
                free(lpThis->WrBuff);
                free(lpThis);
                return(NULL);
        }

        // And return a pointer to it
        return (lpThis);
}

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

//
// GetRequest()
//
// Returns the REQUEST that Socket belongs to
//
PREQUEST FTPGetRequest(SOCKET socket)
```

```
{
        PREQUEST lpReq;

        //
        // Walk through the list looking
        // for this socket
        //
        lpReq = lpFTPHead;
        while(lpReq != NULL)
        {
                if (lpReq->socket == socket)
                        break;
                lpReq = lpReq->lpNext;
        }

        return(lpReq);
}

PREQUEST FTPGetRequestByPVar(PTInstVar pvar)
{
        PREQUEST lpReq;

        //
        // Walk through the list looking
        // for this socket
        //
        lpReq = lpFTPHead;
        while(lpReq != NULL)
        {
                if (lpReq->pvar == pvar)
                        break;
                lpReq = lpReq->lpNext;
        }

        return(lpReq);
}

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

//
// DelRequest()
//
// Delete the REQUEST from the list
// and free associated memory
//
void FTPDelRequest(PREQUEST lpThis)
{
        PREQUEST lpReq;
        BOOL bRet = FALSE;

        //
        // Search for "this"
        //

        // If This is the first member
        if (lpThis == lpFTPHead)
        {
                // This->Next becomes the head
                lpFTPHead = lpThis->lpNext;
        }
        else
        {
                // Search the list
                for (lpReq = lpFTPHead; lpReq; lpReq = lpReq->lpNext)
                {
                        // If This is next...
```



```
                        if (lpReq->lpNext == lpThis)
                        {
                                // Set the pointer to skip "this"
                                lpReq->lpNext = lpThis->lpNext;
                                break;
                        }
                }
        }
        // free the memory
        FTPFreeRequest(lpThis);
}

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

//
// GetFirstRequest()
//
// Return the head of the list
// even if it is NULL
//
PREQUEST        FTPGetFirstRequest(void)
{
        return(lpFTPHead);
}

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

//
// GetNextRequest()
//
// Return the next REQUEST after This
//
PREQUEST        FTPGetNextRequest(PREQUEST lpThis)
{
        return(lpThis->lpNext);
}

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

//
// DelAllRequests()
//
// Delete all of the REQUESTs in the list
// and free all memory
//
void FTPDelAllRequests(void)
{
        PREQUEST lpReq;
        PREQUEST lpNext;

        //
        // Walk through the list
        // freeing memory
        //
        for (lpReq = lpFTPHead; lpReq; )
        {
                lpNext = lpReq->lpNext;
                FTPFreeRequest(lpReq);
                lpReq = lpNext;
        }
        lpFTPHead = NULL;
}

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

static void FTPFreeRequest(PREQUEST lpReq) {
```

```
        free(lpReq->SSHRdBuff);
        free(lpReq->RdBuff);
        free(lpReq->WrBuff);
        free(lpReq);
}
```

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by mailwraprc.rc
//

#define IDI_SECURETT                    100

#define IDD_ABOUTDIALOG                 101
#define IDD_SSHAUTH                     102
#define IDD_HOSTDLG                     600
#define IDD_SSHSETUP                    103
#define IDD_SSHUNKNOWNHOST              104
#define IDD_SSHDIFFERENTHOST            105
#define IDD_SSHAUTHSETUP                106
#define IDD_SSHFWDSETUP                 107
#define IDD_SSHFWDEDIT                  108
#define IDD_SSHTISAUTH                  109

#define IDC_ABOUTTEXT                   1000

#define IDC_SSHAUTHBANNER               1010
#define IDC_SSHAUTHBANNER2              1011

#define IDC_SSHUSERNAMELABEL            1020
#define IDC_SSHUSERNAME                 1021

#define IDC_SSHPASSWORDCAPTION          1030
#define IDC_SSHPASSWORD                 1031

#define IDC_SSHUSEPASSWORD              1040
#define IDC_SSHUSERSA                   1041
#define IDC_SSHUSERHOSTS                1042
#define IDC_SSHUSETIS                   1043

#define IDC_CHOOSERSAFILE               1050
#define IDC_RSAFILENAME                 1051

#define IDC_LOCALUSERNAMELABEL          1060
#define IDC_LOCALUSERNAME               1061
#define IDC_CHOOSEHOSTRSAFILE           1062
#define IDC_HOSTRSAFILENAME             1063

#define IDC_HOSTNAMELABEL               601
#define IDC_HOSTNAME                    602
#define IDC_SERVICELABEL                603
#define IDC_HOSTTELNET                  604
#define IDC_HOSTSSH                     605
#define IDC_HOSTOTHER                   606
#define IDC_HOSTTCPPORTLABEL            607
#define IDC_HOSTTCPPORT                 608
#define IDC_HOSTCOMLABEL                609
#define IDC_HOSTCOM                     610
#define IDC_HOSTTCPIP                   611
#define IDC_HOSTSERIAL                  612
#define IDC_HOSTHELP                    699

#define IDC_SSHCOMPRESSIONLEVEL         1070
#define IDC_SSHCIPHERPREFS              1071
#define IDC_SSHMOVECIPHERUP             1072
#define IDC_SSHMOVECIPHERDOWN           1073
#define IDC_CHOOSEREADWRITEFILE         1074
#define IDC_READWRITEFILENAME           1075
#define IDC_CHOOSEREADONLYFILE          1076
#define IDC_READONLYFILENAME            1077

#define IDC_ADDTOKNOWNHOSTS             1080
```



```
#define IDC_HOSTWARNING                 1081

#define IDC_CONTINUE                    1090

#define IDC_SSHFWDLIST                  1100
#define IDC_ADD                         1101
#define IDC_EDIT                        1102
#define IDC_REMOVE                      1103
#define IDC_SSHFWDX11                   1104

#define IDC_SSHFWDLOCALTOREMOTE         1110
#define IDC_SSHFWDREMOTETOLOCAL         1111
#define IDC_SSHLTRFROMPORT              1112
#define IDC_SSHLTRTOHOST                1113
#define IDC_SSHLTRTOPORT                1114
#define IDC_SSHRTLFROMPORT              1115
#define IDC_SSHRTLTOHOST                1116
#define IDC_SSHRTLTOPORT                1117


#define WSABASEERR              10000
/*
 * Windows Sockets definitions of regular Microsoft C error constants
 */
#define WSAEINTR                (WSABASEERR+4)
#define WSAEBADF                (WSABASEERR+9)
#define WSAEACCES               (WSABASEERR+13)
#define WSAEFAULT               (WSABASEERR+14)
#define WSAEINVAL               (WSABASEERR+22)
#define WSAEMFILE               (WSABASEERR+24)

/*
 * Windows Sockets definitions of regular Berkeley error constants
 */
#define WSAEWOULDBLOCK          (WSABASEERR+35)
#define WSAEINPROGRESS          (WSABASEERR+36)
#define WSAEALREADY             (WSABASEERR+37)
#define WSAENOTSOCK             (WSABASEERR+38)
#define WSAEDESTADDRREQ         (WSABASEERR+39)
#define WSAEMSGSIZE             (WSABASEERR+40)
#define WSAEPROTOTYPE           (WSABASEERR+41)
#define WSAENOPROTOOPT          (WSABASEERR+42)
#define WSAEPROTONOSUPPORT      (WSABASEERR+43)
#define WSAESOCKTNOSUPPORT      (WSABASEERR+44)
#define WSAEOPNOTSUPP           (WSABASEERR+45)
#define WSAEPFNOSUPPORT         (WSABASEERR+46)
#define WSAEAFNOSUPPORT         (WSABASEERR+47)
#define WSAEADDRINUSE           (WSABASEERR+48)
#define WSAEADDRNOTAVAIL        (WSABASEERR+49)
#define WSAENETDOWN             (WSABASEERR+50)
#define WSAENETUNREACH          (WSABASEERR+51)
#define WSAENETRESET            (WSABASEERR+52)
#define WSAECONNABORTED         (WSABASEERR+53)
#define WSAECONNRESET           (WSABASEERR+54)
#define WSAENOBUFS              (WSABASEERR+55)
#define WSAEISCONN              (WSABASEERR+56)
#define WSAENOTCONN             (WSABASEERR+57)
#define WSAESHUTDOWN            (WSABASEERR+58)
#define WSAETOOMANYREFS         (WSABASEERR+59)
#define WSAETIMEDOUT            (WSABASEERR+60)
#define WSAECONNREFUSED         (WSABASEERR+61)
#define WSAELOOP                (WSABASEERR+62)
#define WSAENAMETOOLONG         (WSABASEERR+63)
#define WSAEHOSTDOWN            (WSABASEERR+64)
#define WSAEHOSTUNREACH         (WSABASEERR+65)
#define WSAENOTEMPTY            (WSABASEERR+66)
```

```
#define WSAEPROCLIM             (WSABASEERR+67)
#define WSAEUSERS               (WSABASEERR+68)
#define WSAEDQUOT               (WSABASEERR+69)
#define WSAESTALE               (WSABASEERR+70)
#define WSAEREMOTE              (WSABASEERR+71)

#define WSAEDISCON              (WSABASEERR+101)

/*
 * Extended Windows Sockets error constant definitions
 */
#define WSASYSNOTREADY          (WSABASEERR+91)
#define WSAVERNOTSUPPORTED      (WSABASEERR+92)
#define WSANOTINITIALISED       (WSABASEERR+93)

/* Authoritative Answer: Host not found */
#define WSAHOST_NOT_FOUND       (WSABASEERR+1001)
#define HOST_NOT_FOUND          WSAHOST_NOT_FOUND

/* Non-Authoritative: Host not found, or SERVERFAIL */
#define WSATRY_AGAIN            (WSABASEERR+1002)
#define TRY_AGAIN               WSATRY_AGAIN

/* Non recoverable errors, FORMERR, REFUSED, NOTIMP */
#define WSANO_RECOVERY          (WSABASEERR+1003)
#define NO_RECOVERY             WSANO_RECOVERY

/* Valid name, no data record of requested type */
#define WSANO_DATA              (WSABASEERR+1004)
#define NO_DATA                 WSANO_DATA

/* no address, look for MX record */
#define WSANO_ADDRESS           WSANO_DATA


#define IDS_ERR_PORTNUM                 2001
#define IDS_ERR_EXECMULTIPLE            2002
#define IDS_STATBAR_IDLE                2003
#define IDS_APPNAME                     2004
#define IDS_ERR_DEFAULT                 2005
#define IDS_CAPTION_ERR                 2006
#define IDS_ERR_WINSOCKVER              2007
#define IDS_STATBAR_STOP                2008
#define IDS_INFO_STARTSVC               2009
#define IDS_INFO_ENDSVC                 2010
#define IDS_EVENT_ACCEPT                2011
#define IDS_EVENT_CONNECT               2012
#define IDS_EVENT_DISCONNECT            2013
#define IDS_ERR_TOOSHORTRES             2014
#define IDS_ERR_NOTRES                  2015
#define IDS_ERR_NOTAUTHSTATE            2016
#define IDS_ERR_NOTTRANSSTATE           2017
#define IDS_ERR_NOTIMESTAMP             2018
#define IDS_ERR_NOUSER                  2019
#define IDS_ERR_NOPASSWORD              2020
#define IDS_ERR_INVALIDCMD              2021
#define IDS_EVENT_RESTART               2022
#define IDS_POPMENU_RUN                 2023
#define IDS_POPMENU_STOP                2024
#define IDS_POPMENU_CFG                 2025
#define IDS_POPMENU_SHOW                2026
#define IDS_POPMENU_EXIT                2027
#define IDS_ERR_NOOPIEDLL               2028
#define IDS_ERR_NOOPIEGEN               2029
#define IDS_ERR_OTPGEN                  2030
#define IDS_ERR_NOOTPCHALLENGE          2031
```



```
#define IDS_ERR_CANTGETPRIVILEDGEDPORT  2032
#define IDS_ERR_NOSSLEAYDLL             2033
#define IDS_ERR_NOLIBEAYDLL             2034
#define IDS_ERR_NOSSLPROC               2035
#define IDS_ERR_SSLERROR                2036
#define IDS_ERR_SSLINIT                 2037
#define IDS_MSG_SSLCIPHER               2038
#define IDR_MENU1                       2101
#define IDD_DIALOG1                     2104
#define IDD_VERSIONDLG                  2104
#define IDI_ICON1                       2105
#define IDI_ICON2                       2106
#define IDR_ACCELERATOR1                2107
#define IDD_SETUPDLG                    2108
#define IDI_ICON3                       2109
#define IDD_LOGSETUPDLG                 2111
#define IDC_HAND                        2112
#define IDC_REMOTEPORT_STD              3000
#define IDC_REMOTEPORT_OTHER            3001
#define IDC_REMOTEPORT                  3002
#define IDC_REJECTREMOTE                3003
#define IDC_LOCALPORT_STD               3004
#define IDC_LOCALPORT_OTHER             3005
#define IDC_LOCALPORT                   3006
#define IDC_REMOTEHOST_IP               3007
#define IDC_REMOTEHOST_NAME             3008
#define IDC_REMOTEIP                    3009
#define IDC_REMOTENAME                  3010
#define IDC_KEROCHAN                    3011
#define IDC_EMAILADDR                   3012
#define IDC_URI                         3013
#define IDC_PROTOPOP3                   3015
#define IDC_PROTOIMAP4                  3016
#define IDC_AUTHMETHOD                  3017
#define IDC_SLIDER1                     3018
#define IDC_CHECK1                      3019
#define IDC_CHECK2                      3020
#define IDC_BUTTON1                     3021
#define IDC_EDIT1                       3022
#define IDC_BUTTON3                     3024
#define IDC_BUTTON4                     3026
#define IDC_BUTTON5                     3027
#define IDC_BUTTON6                     3028
#define IDC_BUTTON7                     3029
#define ID_MENUITEM_STARTSVC            40001
#define ID_MENUITEM_ENDSVC              40002
#define ID_MENUITEM_EXIT                40003
#define ID_MENUITEM_HELP                40004
#define ID_MENUITEM_VERSION             40005
#define ID_MENUITEM_AUTORUN             40008
#define ID_MENUITEM_AUTOSTARTSVC        40009
#define ID_MENUITEM_SETUP               40010
#define ID_ACCEL40011                   40011
#define ID_ACCEL40012                   40012
#define ID_MENUITEM_LOGSETUP            40013

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        113
#define _APS_NEXT_COMMAND_VALUE         40014
#define _APS_NEXT_CONTROL_VALUE         1030
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

```
// SSH Connection Manager

#include "ftpwrap.h"
#include "ttxssh.h"


DWORD SCM_connect(HWND hwnd, PREQUEST lpReq) {
        PTInstVar pvar;

#define READ_STD_STRING_OPTION(name1,name2) \
  strncpy(pvar->settings.name1, name2, sizeof(pvar->settings.name1))

        pvar = GetNewRequest();
        lpReq->pvar = pvar;

//  pvar->settings = *pvar->ts_SSH;
        pvar->fatal_error = FALSE;
        pvar->showing_err = FALSE;
        pvar->err_msg = NULL;

        init_TTSSH(pvar);

        pvar->TCPPort = 22;

        pvar->settings.CompressionLevel = 4;
        pvar->settings.Enabled = TRUE;
        strncpy(pvar->HostName, lpReq->HostName, sizeof( pvar->HostName ) );
        READ_STD_STRING_OPTION(DefaultUserName,lpReq->UserName);
        READ_STD_STRING_OPTION(DefaultForwarding,"R10001:localhost:1;R10002:localhost
:2;R10003:localhost:3;R10004:localhost:4;R10005:localhost:5;R10006:localhost:6;R10007
:localhost:7;R10008:localhost:8");
        READ_STD_STRING_OPTION(DefaultRhostsLocalUserName,lpReq->UserName);
        READ_STD_STRING_OPTION(DefaultRhostsHostPrivateKeyFile,"ssh_host_key");
        READ_STD_STRING_OPTION(DefaultRSAPrivateKeyFile,"identity");
        READ_STD_STRING_OPTION(CipherOrder,"302615");
        READ_STD_STRING_OPTION(KnownHostsFiles,"ssh_known_hosts");
        pvar->settings.TryDefaultAuth = FALSE;
        pvar->settings.DefaultAuthMethod = SSH_AUTH_RHOSTS;
//      pvar->settings.DefaultAuthMethod = SSH_AUTH_RSA;
        pvar->settings.LogLevel = 100;
        pvar->settings.WriteBufferSize = 2097152;
        pvar->settings.LocalForwardingIdentityCheck = TRUE;

        CommOpen(hwnd,pvar);

        return 0;
}
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

#include "ttxssh.h"
#include "util.h"

#include <bn.h>
#include <limits.h>

static char ssh_ttymodes[] = "\x01\x03\x02\x1c\x03\x08\x04\x15\x05\x04";

static void try_send_credentials(PTInstVar pvar);
static void prep_compression(PTInstVar pvar);

static int get_predecryption_amount(PTInstVar pvar) {
  static int small_block_decryption_sizes[] = { 5, 5, 6, 6, 8 };

  if (SSHv1(pvar)) {
    return 0;
  } else {
    int block_size = CRYPT_get_decryption_block_size(pvar);

    if (block_size < 5) {
      return small_block_decryption_sizes[block_size];
    } else {
      return block_size;
    }
  }
}

/* Get up to 'limit' bytes into the payload buffer.
   'limit' is counted from the start of the payload data.
   Returns the amount of data in the payload buffer, or
   -1 if there is an error.
   We can return more than limit in some cases. */
static int buffer_packet_data(PTInstVar pvar, int limit) {
```



```
  if (pvar->ssh_state.payloadlen >= 0) {
    return pvar->ssh_state.payloadlen;
  } else {
    int cur_decompressed_bytes = pvar->ssh_state.decompress_stream.next_out -
      pvar->ssh_state.postdecompress_inbuf;

    while (limit > cur_decompressed_bytes) {
      int result;

      pvar->ssh_state.payload = pvar->ssh_state.postdecompress_inbuf + 1;
      if (pvar->ssh_state.postdecompress_inbuflen == cur_decompressed_bytes) {
        buf_ensure_size(&pvar->ssh_state.postdecompress_inbuf,
          &pvar->ssh_state.postdecompress_inbuflen,
          min(limit, cur_decompressed_bytes*2));
      }

      pvar->ssh_state.decompress_stream.next_out
        = pvar->ssh_state.postdecompress_inbuf + cur_decompressed_bytes;
      pvar->ssh_state.decompress_stream.avail_out
        = min(limit, pvar->ssh_state.postdecompress_inbuflen)
            - cur_decompressed_bytes;

      result = inflate(&pvar->ssh_state.decompress_stream, Z_SYNC_FLUSH);
      cur_decompressed_bytes = pvar->ssh_state.decompress_stream.next_out -
        pvar->ssh_state.postdecompress_inbuf;

      switch (result) {
      case Z_OK:
        break;
      case Z_BUF_ERROR:
        pvar->ssh_state.payloadlen = cur_decompressed_bytes;
        return cur_decompressed_bytes;
      default:
        notify_fatal_error(pvar, "受信したパケットの圧縮データが不正です．");
        return -1;
      }
    }

    return cur_decompressed_bytes;
  }
}

/* For use by the protocol processing code.
   Gets N bytes of uncompressed payload. Returns FALSE if data not available
   and a fatal error has been signaled.
   The data is available in the payload buffer. This buffer address
   can change during a call to grab_payload, so take care!
   The payload pointer is set to point to the first byte of the actual data
   (after the packet type byte).
*/
static BOOL grab_payload(PTInstVar pvar, int num_bytes) {
  /* Accept maximum of 4MB of payload data */
  int in_buffer = buffer_packet_data(pvar, 4*1024*1024);

  if (in_buffer < 0) {
    return FALSE;
  } else {
    pvar->ssh_state.payload_grabbed += num_bytes;
    if (pvar->ssh_state.payload_grabbed > in_buffer) {
      notify_fatal_error(pvar, "受信したパケットは途中で切れています．");
      return FALSE;
    } else {
      return TRUE;
    }
  }
}
```

```
static BOOL grab_payload_limited(PTInstVar pvar, int num_bytes) {
  int in_buffer;

  pvar->ssh_state.payload_grabbed += num_bytes;
  in_buffer = buffer_packet_data(pvar, pvar->ssh_state.payload_grabbed);

  if (in_buffer < 0) {
    return FALSE;
  } else {
    if (pvar->ssh_state.payload_grabbed > in_buffer) {
      notify_fatal_error(pvar, "受信したパケットは途中で切れています．");
      return FALSE;
    } else {
      return TRUE;
    }
  }
}

#define get_payload_uint32(pvar, offset) get_uint32_MSBfirst((pvar)->ssh_state.payloa
d + (offset))
#define get_uint32(buf) get_uint32_MSBfirst((buf))
#define set_uint32(buf, v) set_uint32_MSBfirst((buf), (v))
#define get_mpint_len(pvar, offset) ((get_ushort16_MSBfirst((pvar)->ssh_state.payload
 + (offset)) + 7) >> 3)
#define get_ushort16(buf) get_ushort16_MSBfirst((buf))

#define do_crc(buf, len) (~(uint32)crc32(0xFFFFFFFF, (buf), (len)))

/* Decrypt the payload, checksum it, eat the padding, get the packet type
   and return it.
   'data' points to the start of the packet --- its length field.
   'len' is the length of the
   payload + padding (+ length of CRC for SSHv1). 'padding' is the length
   of the padding alone. */
static int prep_packet(PTInstVar pvar, char FAR * data, int len, int padding) {
  pvar->ssh_state.payload = data + 4;
  pvar->ssh_state.payloadlen = len;

  if (SSHv1(pvar)) {
    if (CRYPT_detect_attack(pvar, pvar->ssh_state.payload, len)) {
      notify_fatal_error(pvar, "'CORE insertion attack' 攻撃を検知しました．接続を中
断します．");
      return SSH_MSG_NONE;
    }

    CRYPT_decrypt(pvar, pvar->ssh_state.payload, len);
    /* PKT guarantees that the data is always 4-byte aligned */
    if (do_crc(pvar->ssh_state.payload, len - 4) != get_uint32_MSBfirst(pvar->ssh_sta
te.payload + len - 4)) {
      notify_fatal_error(pvar, "データの破損を検知しました．接続を切断します．");
      return SSH_MSG_NONE;
    }

    pvar->ssh_state.payload += padding;
    pvar->ssh_state.payloadlen -= padding + 4;
  } else {
    int already_decrypted = get_predecryption_amount(pvar);

    CRYPT_decrypt(pvar, data + already_decrypted, len - already_decrypted);

    if (!CRYPT_verify_receiver_MAC(pvar, pvar->ssh_state.receiver_sequence_number,
      data, len + 4, data + len + 4)) {
      notify_fatal_error(pvar, "データの破損を検知しました．接続を切断します．");
      return SSH_MSG_NONE;
    }
```



```
    pvar->ssh_state.payload++;
    pvar->ssh_state.payloadlen -= padding + 1;
  }

  pvar->ssh_state.payload_grabbed = 0;

  if (pvar->ssh_state.decompressing) {
    if (pvar->ssh_state.decompress_stream.avail_in != 0) {
      notify_nonfatal_error(pvar, "内部エラー: パケットを完全に展開できませんでした．
\n"
        "バグレポートをお願いします．");
    }

    pvar->ssh_state.decompress_stream.next_in = pvar->ssh_state.payload;
    pvar->ssh_state.decompress_stream.avail_in = pvar->ssh_state.payloadlen;
    pvar->ssh_state.decompress_stream.next_out = pvar->ssh_state.postdecompress_inbuf
;
    pvar->ssh_state.payloadlen = -1;
  } else {
    pvar->ssh_state.payload++;
  }

  if (!grab_payload_limited(pvar, 1)) {
    return SSH_MSG_NONE;
  }

  pvar->ssh_state.receiver_sequence_number++;

  return pvar->ssh_state.payload[-1];
}

/* Create a packet to be sent. The SSH protocol packet type is in 'type';
   'len' contains the length of the packet payload, in bytes (this
   does not include the space for any of the packet headers or padding,
   or for the packet type byte).
   Returns a pointer to the payload data area, a region of length 'len',
   to be filled by the caller. */
static unsigned char FAR * begin_send_packet(PTInstVar pvar, int type, int len) {
  unsigned char FAR * buf;

  pvar->ssh_state.outgoing_packet_len = len + 1;

  if (pvar->ssh_state.compressing) {
    buf_ensure_size(&pvar->ssh_state.precompress_outbuf, &pvar->ssh_state.precompress
_outbuflen,
      1 + len);
    buf = pvar->ssh_state.precompress_outbuf;
  } else {
    /* For SSHv2,
       Encrypted_length is 4(packetlength) + 1(paddinglength) + 1(packettype)
                           + len(payload) + 4(minpadding), rounded up to nearest block_
size
       We only need a reasonable upper bound for the buffer size */
    buf_ensure_size(&pvar->ssh_state.outbuf, &pvar->ssh_state.outbuflen,
      len + 30 + CRYPT_get_sender_MAC_size(pvar) + CRYPT_get_encryption_block_size(pv
ar));
    buf = pvar->ssh_state.outbuf + 12;
  }

  buf[0] = (unsigned char)type;
  return buf + 1;
}

#define finish_send_packet(pvar) finish_send_packet_special((pvar), 0)
```

```
static BOOL send_packet_blocking(PTInstVar pvar, char FAR * data, int len) {
  u_long do_block = 0;

  if (WSAAsyncSelect(pvar->socket, pvar->NotificationWindow,
        0, 0) == SOCKET_ERROR
    || ioctlsocket(pvar->socket, FIONBIO, &do_block) == SOCKET_ERROR
    || send(pvar->socket, data, len, 0) != len
    || WSAAsyncSelect(pvar->socket, pvar->NotificationWindow,
         pvar->notification_msg, pvar->notification_events) == SOCKET_ERROR) {
          char str[512], errstr[256];
          LoadString(hInst, WSAGetLastError(), errstr, sizeof(errstr));
          wsprintf(str, "SSHパケットを送信中に通信エラーが発生しました．(%s)\n"
       "接続を閉じます．", errstr);
      notify_fatal_error(pvar, str);
    return FALSE;
  } else {
    return TRUE;
  }
}

/* if skip_compress is true, then the data has already been compressed
   into outbuf + 12 */
static void finish_send_packet_special(PTInstVar pvar, int skip_compress) {
  int len = pvar->ssh_state.outgoing_packet_len;
  char FAR * data;
  int data_length;

  if (pvar->ssh_state.compressing) {
    if (!skip_compress) {
      buf_ensure_size(&pvar->ssh_state.outbuf, &pvar->ssh_state.outbuflen,
        len + (len >> 6) + 50 + CRYPT_get_sender_MAC_size(pvar));
      pvar->ssh_state.compress_stream.next_in = pvar->ssh_state.precompress_outbuf;
      pvar->ssh_state.compress_stream.avail_in = len;
      pvar->ssh_state.compress_stream.next_out = pvar->ssh_state.outbuf + 12;
      pvar->ssh_state.compress_stream.avail_out = pvar->ssh_state.outbuflen - 12;

      if (deflate(&pvar->ssh_state.compress_stream, Z_SYNC_FLUSH) != Z_OK) {
        notify_fatal_error(pvar, "データを圧縮中にエラーが発生しました．\n"
          "接続を閉じます．");
        return;
      }
    }

    len = pvar->ssh_state.outbuflen - 12 - pvar->ssh_state.compress_stream.avail_out;
  }

  if (SSHv1(pvar)) {
    int padding = 8 - ((len + 4) % 8);

    data = pvar->ssh_state.outbuf + 8 - padding;
    data_length = padding + len + 8;

    set_uint32(data, len + 4);
    if (CRYPT_get_receiver_cipher(pvar) != SSH_CIPHER_NONE) {
      CRYPT_set_random_data(pvar, data + 4, padding);
    } else {
      memset(data + 4, 0, padding);
    }
    set_uint32(data + data_length - 4, do_crc(data + 4, data_length - 8));
    CRYPT_encrypt(pvar, data + 4, data_length - 4);
  } else {
    int block_size = CRYPT_get_encryption_block_size(pvar);
    int encryption_size;
    int padding;

    if (block_size < 8) {
```



```
      block_size = 8;
    }
    encryption_size = ((len + 8)/block_size + 1)*block_size;
    data = pvar->ssh_state.outbuf + 7;
    data_length = encryption_size + CRYPT_get_sender_MAC_size(pvar);

    set_uint32(data, encryption_size - 4);
    padding = encryption_size - len - 5;
    data[4] = (unsigned char)padding;
    CRYPT_set_random_data(pvar, data + 5 + len, padding);
    CRYPT_build_sender_MAC(pvar, pvar->ssh_state.sender_sequence_number,
      data, encryption_size, data + encryption_size);
    CRYPT_encrypt(pvar, data, encryption_size);
  }

  send_packet_blocking(pvar, data, data_length);

  pvar->ssh_state.sender_sequence_number++;
}

static void destroy_packet_buf(PTInstVar pvar) {
  memset(pvar->ssh_state.outbuf, 0, pvar->ssh_state.outbuflen);
  if (pvar->ssh_state.compressing) {
    memset(pvar->ssh_state.precompress_outbuf, 0, pvar->ssh_state.precompress_outbufl
en);
  }
}

/* The handlers are added to the queue for each message. When one of the
   handlers fires, if it returns FALSE, then all handlers in the set are
   removed from their queues. */
static void enque_handlers(PTInstVar pvar, int num_msgs, const int FAR * messages,
  const SSHPacketHandler FAR * handlers) {
  SSHPacketHandlerItem FAR * first_item;
  SSHPacketHandlerItem FAR * last_item = NULL;
  int i;

  for (i = 0; i < num_msgs; i++) {
    SSHPacketHandlerItem FAR * item = (SSHPacketHandlerItem FAR *)malloc(sizeof(SSHPa
cketHandlerItem));
    SSHPacketHandlerItem FAR * cur_item = pvar->ssh_state.packet_handlers[messages[i]
];

    item->handler = handlers[i];

    if (cur_item == NULL) {
      pvar->ssh_state.packet_handlers[messages[i]] = item;
      item->next_for_message = item;
      item->last_for_message = item;
      item->active_for_message = messages[i];
    } else {
      item->next_for_message = cur_item;
      item->last_for_message = cur_item->last_for_message;
      cur_item->last_for_message->next_for_message = item;
      cur_item->last_for_message = item;
      item->active_for_message = -1;
    }

    if (last_item != NULL) {
      last_item->next_in_set = item;
    } else {
      first_item = item;
    }
    last_item = item;
  }
```

```
  if (last_item != NULL) {
    last_item->next_in_set = first_item;
  }
}

static SSHPacketHandler get_handler(PTInstVar pvar, int message) {
  SSHPacketHandlerItem FAR * cur_item = pvar->ssh_state.packet_handlers[message];

  if (cur_item == NULL) {
    return NULL;
  } else {
    return cur_item->handler;
  }
}

/* Called only by SSH_handle_packet */
static void deque_handlers(PTInstVar pvar, int message) {
  SSHPacketHandlerItem FAR * cur_item = pvar->ssh_state.packet_handlers[message];
  SSHPacketHandlerItem FAR * first_item_in_set = cur_item;

  do {
    SSHPacketHandlerItem FAR * next_in_set = cur_item->next_in_set;

    if (cur_item->active_for_message >= 0) {
      SSHPacketHandlerItem FAR * replacement = cur_item->next_for_message;

      if (replacement == cur_item) {
        replacement = NULL;
      } else {
        replacement->active_for_message = cur_item->active_for_message;
      }
      pvar->ssh_state.packet_handlers[cur_item->active_for_message] = replacement;
    }
    cur_item->next_for_message->last_for_message = cur_item->last_for_message;
    cur_item->last_for_message->next_for_message = cur_item->next_for_message;

    free(cur_item);
    cur_item = next_in_set;
  } while (cur_item != first_item_in_set);
}

static void enque_handler(PTInstVar pvar, int message, SSHPacketHandler handler) {
  enque_handlers(pvar, 1, &message, &handler);
}

static void chop_newlines(char FAR * buf) {
  int len = strlen(buf);

  while (len > 0 && (buf[len - 1] == '\n' || buf[len - 1] == '\r')) {
    buf[len - 1] = 0;
    len--;
  }
}

/********************/
/* Message handlers */
/********************/

static BOOL handle_forwarding_success(PTInstVar pvar) {
  return FALSE;
}

static BOOL handle_forwarding_failure(PTInstVar pvar) {
  return FALSE;
}
```



```
static void enque_forwarding_request_handlers(PTInstVar pvar) {
  static const int msgs[] = { SSH_SMSG_SUCCESS, SSH_SMSG_FAILURE };
  static const SSHPacketHandler handlers[]
    = { handle_forwarding_success, handle_forwarding_failure };

  enque_handlers(pvar, 2, msgs, handlers);
}

static BOOL handle_auth_failure(PTInstVar pvar) {
  notify_verbose_message(pvar, "Authentication failed", LOG_LEVEL_VERBOSE);

  AUTH_set_generic_mode(pvar);
  AUTH_advance_to_next_cred(pvar);
  pvar->ssh_state.status_flags &= ~STATUS_DONT_SEND_CREDENTIALS;
  try_send_credentials(pvar);
  return FALSE;
}

static BOOL handle_rsa_auth_refused(PTInstVar pvar) {
  AUTH_destroy_cur_cred(pvar);
  return handle_auth_failure(pvar);
}

static BOOL handle_TIS_challenge(PTInstVar pvar) {
  if (grab_payload(pvar, 4)) {
    int len = get_payload_uint32(pvar, 0);

    if (grab_payload(pvar, len)) {
      notify_verbose_message(pvar, "Received TIS challenge", LOG_LEVEL_VERBOSE);

      AUTH_set_TIS_mode(pvar, pvar->ssh_state.payload + 4, len);
      AUTH_advance_to_next_cred(pvar);
      pvar->ssh_state.status_flags &= ~STATUS_DONT_SEND_CREDENTIALS;
      try_send_credentials(pvar);
    }
  }
  return FALSE;
}

static BOOL handle_auth_required(PTInstVar pvar) {
  notify_verbose_message(pvar, "Server requires authentication", LOG_LEVEL_VERBOSE);

  pvar->ssh_state.status_flags &= ~STATUS_DONT_SEND_CREDENTIALS;
  try_send_credentials(pvar);
  /* the first AUTH_advance_to_next_cred is issued early by ttxssh.c */

  return FALSE;
}

static BOOL handle_ignore(PTInstVar pvar) {
  if (grab_payload(pvar, 4) && grab_payload(pvar, get_payload_uint32(pvar, 0))) {
    /* ignore it! but it must be decompressed */
  }
  return TRUE;
}

static BOOL handle_debug(PTInstVar pvar) {
  BOOL always_display;
  char FAR * description;
  int description_len;
  char buf[2048];

  if (SSHv1(pvar)) {
    if (grab_payload(pvar, 4)
      && grab_payload(pvar, description_len = get_payload_uint32(pvar, 0))) {
      always_display = FALSE;
```

```
      description = pvar->ssh_state.payload + 4;
      description[description_len] = 0;
    } else {
      return TRUE;
    }
  } else {
    if (grab_payload(pvar, 5)
      && grab_payload(pvar, (description_len = get_payload_uint32(pvar, 1)) + 4)
      && grab_payload(pvar, get_payload_uint32(pvar, 5 + description_len))) {
      always_display = pvar->ssh_state.payload[0] != 0;
      description = pvar->ssh_state.payload + 5;
      description[description_len] = 0;
    } else {
      return TRUE;
    }
  }

  chop_newlines(description);
  _snprintf(buf, sizeof(buf), "DEBUG message from server: %s", description);
  buf[sizeof(buf) - 1] = 0;
  if (always_display) {
    notify_nonfatal_error(pvar, buf);
  } else {
    notify_verbose_message(pvar, buf, LOG_LEVEL_VERBOSE);
  }
  return TRUE;
}

static BOOL handle_disconnect(PTInstVar pvar) {
  int reason_code;
  char FAR * description;
  int description_len;
  char buf[2048];
  char FAR * explanation = "";

  if (SSHv1(pvar)) {
    if (grab_payload(pvar, 4)
      && grab_payload(pvar, description_len = get_payload_uint32(pvar, 0))) {
      reason_code = -1;
      description = pvar->ssh_state.payload + 4;
      description[description_len] = 0;
    } else {
      return TRUE;
    }
  } else {
    if (grab_payload(pvar, 8)
      && grab_payload(pvar, (description_len = get_payload_uint32(pvar, 4)) + 4)
      && grab_payload(pvar, get_payload_uint32(pvar, 8 + description_len))) {
      reason_code = get_payload_uint32(pvar, 0);
      description = pvar->ssh_state.payload + 8;
      description[description_len] = 0;
    } else {
      return TRUE;
    }
  }

  chop_newlines(description);
  if (description[0] == 0) {
    description = NULL;
  }

  if (get_handler(pvar, SSH_SMSG_FAILURE) == handle_forwarding_failure) {
    explanation = "\nあなたが要求したサーバからのポート転送を実行できなかったため、切
断された可能性があります．\n"
      "既に他人がサーバの同じポートからの転送を行っていることが考えられます．";
  }
```



```
  if (description != NULL) {
    _snprintf(buf, sizeof(buf), "サーバから接続を切断されました．メッセージ: '%s'%s",
      description, explanation);
  } else {
    _snprintf(buf, sizeof(buf), "サーバから接続を切断されました(原因は不明です)．%s",
      explanation);
  }
  buf[sizeof(buf) - 1] = 0;
  notify_fatal_error(pvar, buf);

  return TRUE;
}

static BOOL handle_unimplemented(PTInstVar pvar) {
  /* Should never receive this since we only send base 2.0 protocol messages */
  grab_payload(pvar, 4);
  return TRUE;
}

static BOOL handle_crypt_success(PTInstVar pvar) {
  notify_verbose_message(pvar, "Secure mode successfully achieved", LOG_LEVEL_VERBOSE
);
  return FALSE;
}

static BOOL handle_noauth_success(PTInstVar pvar) {
  notify_verbose_message(pvar, "Server does not require authentication", LOG_LEVEL_VE
RBOSE);
  prep_compression(pvar);
  return FALSE;
}

static BOOL handle_auth_success(PTInstVar pvar) {
  notify_verbose_message(pvar, "Authentication accepted", LOG_LEVEL_VERBOSE);
  prep_compression(pvar);
  return FALSE;
}

static BOOL handle_server_public_key(PTInstVar pvar) {
  int server_key_public_exponent_len;
  int server_key_public_modulus_pos;
  int server_key_public_modulus_len;
  int host_key_bits_pos;
  int host_key_public_exponent_len;
  int host_key_public_modulus_pos;
  int host_key_public_modulus_len;
  int protocol_flags_pos;
  int supported_ciphers;
  char FAR * inmsg;

  if (!grab_payload(pvar, 14)) return FALSE;
  server_key_public_exponent_len = get_mpint_len(pvar, 12);

  if (!grab_payload(pvar, server_key_public_exponent_len + 2)) return FALSE;
  server_key_public_modulus_pos = 14 + server_key_public_exponent_len;
  server_key_public_modulus_len = get_mpint_len(pvar, server_key_public_modulus_pos);

  if (!grab_payload(pvar, server_key_public_modulus_len + 6)) return FALSE;
  host_key_bits_pos = server_key_public_modulus_pos + 2 + server_key_public_modulus_l
en;
  host_key_public_exponent_len = get_mpint_len(pvar, host_key_bits_pos + 4);

  if (!grab_payload(pvar, host_key_public_exponent_len + 2)) return FALSE;
  host_key_public_modulus_pos = host_key_bits_pos + 6 + host_key_public_exponent_len;
  host_key_public_modulus_len = get_mpint_len(pvar, host_key_public_modulus_pos);
```

```
  if (!grab_payload(pvar, host_key_public_modulus_len + 12)) return FALSE;
  protocol_flags_pos = host_key_public_modulus_pos + 2 + host_key_public_modulus_len;

  inmsg = pvar->ssh_state.payload;

  CRYPT_set_server_cookie(pvar, inmsg);
  if (!CRYPT_set_server_RSA_key(pvar, get_uint32(inmsg + 8), pvar->ssh_state.payload
+ 12,
    inmsg + server_key_public_modulus_pos)) return FALSE;
  if (!CRYPT_set_host_RSA_key(pvar, get_uint32(inmsg + host_key_bits_pos),
    inmsg + host_key_bits_pos + 4,
    inmsg + host_key_public_modulus_pos)) return FALSE;
  pvar->ssh_state.server_protocol_flags = get_uint32(inmsg + protocol_flags_pos);

  supported_ciphers = get_uint32(inmsg + protocol_flags_pos + 4);
  if (!CRYPT_set_supported_ciphers(pvar, supported_ciphers, supported_ciphers)) retur
n FALSE;
  if (!AUTH_set_supported_auth_types(pvar, get_uint32(inmsg + protocol_flags_pos + 8)
)) return FALSE;

  /* this must be the LAST THING in this function, since it can cause
     host_is_OK to be called. */
  HOSTS_check_host_key(pvar, pvar->ssh_state.hostname,
    get_uint32(inmsg + host_key_bits_pos),
    inmsg + host_key_bits_pos + 4,
    inmsg + host_key_public_modulus_pos);

  return FALSE;
}

/*
The ID must have already been found to start with "SSH-". It must
be null-terminated.
*/
static BOOL parse_protocol_ID(PTInstVar pvar, char FAR * ID) {
  char FAR * str;

  for (str = ID + 4; *str >= '0' && *str <= '9'; str++) {
  }

  if (*str != '.') {
    return FALSE;
  }

  pvar->protocol_major = atoi(ID + 4);
  pvar->protocol_minor = atoi(str + 1);

  for (str = str + 1; *str >= '0' && *str <= '9'; str++) {
  }

  return *str == '-';
}

/*
On entry, the pvar->protocol_xxx fields hold the server's advertised
protocol number. We replace the fields with the protocol number we will
actually use, or return FALSE if there is no usable protocol version.
*/
static BOOL negotiate_protocol(PTInstVar pvar) {
  switch (pvar->protocol_major) {
  case 1:
    if (pvar->protocol_minor > 5) {
      pvar->protocol_minor = 5;
    }
```



```
    return TRUE;

  case 2:
  default:
    return FALSE;
  }
}

static void init_protocol(PTInstVar pvar) {
  CRYPT_initialize_random_numbers(pvar);
  HOSTS_prefetch_host_key(pvar, pvar->ssh_state.hostname);
  /* while we wait for a response from the server... */

  if (SSHv1(pvar)) {
    enque_handler(pvar, SSH_MSG_DISCONNECT, handle_disconnect);
    enque_handler(pvar, SSH_MSG_IGNORE, handle_ignore);
    enque_handler(pvar, SSH_MSG_DEBUG, handle_debug);
    enque_handler(pvar, SSH_SMSG_PUBLIC_KEY, handle_server_public_key);
  } else {
    enque_handler(pvar, SSH2_MSG_DISCONNECT, handle_disconnect);
    enque_handler(pvar, SSH2_MSG_IGNORE, handle_ignore);
    enque_handler(pvar, SSH2_MSG_DEBUG, handle_debug);
    enque_handler(pvar, SSH2_MSG_UNIMPLEMENTED, handle_unimplemented);
  }
}

BOOL SSH_handle_server_ID(PTInstVar pvar, char FAR * ID, int ID_len) {
  static const char prefix[] = "Received server prologue string: ";

  if (ID_len <= 0) {
    return FALSE;
  } else {
    char FAR * buf = (char FAR *)malloc(ID_len + NUM_ELEM(prefix));

    strcpy(buf, prefix);
    strncpy(buf + NUM_ELEM(prefix) - 1, ID, ID_len);
    buf[NUM_ELEM(prefix) + ID_len - 1] = 0;
    chop_newlines(buf);

    notify_verbose_message(pvar, buf, LOG_LEVEL_VERBOSE);

    free(buf);

    if (ID[ID_len - 1] != '\n') {
      pvar->ssh_state.status_flags |= STATUS_IN_PARTIAL_ID_STRING;
      return FALSE;
    } else if ((pvar->ssh_state.status_flags & STATUS_IN_PARTIAL_ID_STRING) != 0) {
      pvar->ssh_state.status_flags &= ~STATUS_IN_PARTIAL_ID_STRING;
      return FALSE;
    } else if (strncmp(ID, "SSH-", 4) != 0) {
      return FALSE;
    } else {
      ID[ID_len - 1] = 0;

      if (ID_len > 1 && ID[ID_len - 2] == '\r') {
        ID[ID_len - 2] = 0;
      }

      pvar->ssh_state.server_ID = _strdup(ID);

      if (!parse_protocol_ID(pvar, ID) || !negotiate_protocol(pvar)) {
        notify_fatal_error(pvar, "サーバで使われているSSHプロトコルのバージョンには対
応していません．");
      } else {
        char TTSSH_ID[1024];
        int TTSSH_ID_len;
```

```
        _snprintf(TTSSH_ID, sizeof(TTSSH_ID), "SSH-%d.%d-FTP Wrapper/0.01 Win32\n",
          pvar->protocol_major, pvar->protocol_minor);
        TTSSH_ID_len = strlen(TTSSH_ID);

        if (send(pvar->socket, TTSSH_ID, TTSSH_ID_len, 0) != TTSSH_ID_len) {
          notify_fatal_error(pvar, "SSH ID文字列を送信中にエラーが発生しました．\n"
            "接続を閉じます．");
        } else {
          init_protocol(pvar);
        }
      }

      return TRUE;
    }
  }
}

static BOOL handle_exit(PTInstVar pvar) {
  if (grab_payload(pvar, 4)) {
    begin_send_packet(pvar, SSH_CMSG_EXIT_CONFIRMATION, 0);
    finish_send_packet(pvar);
    notify_closed_connection(pvar);
  }
  return TRUE;
}

static BOOL handle_data(PTInstVar pvar) {
  if (grab_payload_limited(pvar, 4)) {
    pvar->ssh_state.payload_datalen = get_payload_uint32(pvar, 0);
    pvar->ssh_state.payload_datastart = 4;
  }
  return TRUE;
}

static BOOL handle_channel_open(PTInstVar pvar) {
  int host_len;
  int originator_len;

  if ((pvar->ssh_state.server_protocol_flags & SSH_PROTOFLAG_HOST_IN_FWD_OPEN) != 0)
{
    if (grab_payload(pvar, 8)
      && grab_payload(pvar, 8 + (host_len = get_payload_uint32(pvar, 4)))
      && grab_payload(pvar, originator_len = get_payload_uint32(pvar, host_len + 12))
) {
      int local_port = get_payload_uint32(pvar, 8 + host_len);

      pvar->ssh_state.payload[8 + host_len] = 0;
      FWD_open(pvar, get_payload_uint32(pvar, 0), pvar->ssh_state.payload + 8,
        local_port, pvar->ssh_state.payload + 16 + host_len, originator_len);
    }
  } else {
    if (grab_payload(pvar, 8)
      && grab_payload(pvar, 4 + (host_len = get_payload_uint32(pvar, 4)))) {
      int local_port = get_payload_uint32(pvar, 8 + host_len);

      pvar->ssh_state.payload[8 + host_len] = 0;
      FWD_open(pvar, get_payload_uint32(pvar, 0), pvar->ssh_state.payload + 8,
        local_port, NULL, 0);
    }
  }

  return TRUE;
}

static BOOL handle_X11_channel_open(PTInstVar pvar) {
```



```
  int originator_len;

  if ((pvar->ssh_state.server_protocol_flags & SSH_PROTOFLAG_HOST_IN_FWD_OPEN) != 0)
{
    if (grab_payload(pvar, 8)
      && grab_payload(pvar, originator_len = get_payload_uint32(pvar, 4))) {
      FWD_X11_open(pvar, get_payload_uint32(pvar, 0), pvar->ssh_state.payload + 8,
        originator_len);
    }
  } else {
    if (grab_payload(pvar, 4)) {
      FWD_X11_open(pvar, get_payload_uint32(pvar, 0), NULL, 0);
    }
  }

  return TRUE;
}

static BOOL handle_channel_open_confirmation(PTInstVar pvar) {
  if (grab_payload(pvar, 8)) {
    FWD_confirmed_open(pvar, get_payload_uint32(pvar, 0), get_payload_uint32(pvar, 4)
);
  }
  return FALSE;
}

static BOOL handle_channel_open_failure(PTInstVar pvar) {
  if (grab_payload(pvar, 4)) {
    FWD_failed_open(pvar, get_payload_uint32(pvar, 0));
  }
  return FALSE;
}

static BOOL handle_channel_data(PTInstVar pvar) {
  int len;

  if (grab_payload(pvar, 8) && grab_payload(pvar, len = get_payload_uint32(pvar, 4)))
 {
    FWD_received_data(pvar, get_payload_uint32(pvar, 0),
      pvar->ssh_state.payload + 8, len);
  }
  return TRUE;
}

static BOOL handle_channel_input_eof(PTInstVar pvar) {
  if (grab_payload(pvar, 4)) {
    FWD_channel_input_eof(pvar, get_payload_uint32(pvar, 0));
  }
  return TRUE;
}

static BOOL handle_channel_output_eof(PTInstVar pvar) {
  if (grab_payload(pvar, 4)) {
    FWD_channel_output_eof(pvar, get_payload_uint32(pvar, 0));
  }
  return TRUE;
}

void SSH_handle_packet(PTInstVar pvar, char FAR * data, int len, int padding) {
  unsigned char message = prep_packet(pvar, data, len, padding);

  if (message != SSH_MSG_NONE) {
    SSHPacketHandler handler = get_handler(pvar, message);

    if (handler == NULL) {
      if (SSHv1(pvar)) {
```

```
        char buf[1024];

        _snprintf(buf, sizeof(buf), "不明なタイプのパケットを受信しました: %d", messa
ge);
        buf[sizeof(buf) - 1] = 0;
        notify_fatal_error(pvar, buf);
      } else {
        unsigned char FAR * outmsg = begin_send_packet(pvar, SSH2_MSG_UNIMPLEMENTED,
4);

        set_uint32(outmsg, pvar->ssh_state.receiver_sequence_number - 1);
        finish_send_packet(pvar);
        /* XXX need to decompress incoming packet, but how? */
      }
    } else {
      if (!handler(pvar)) {
        deque_handlers(pvar, message);
      }
    }
  }
}

static BOOL handle_pty_success(PTInstVar pvar) {
  FWD_enter_interactive_mode(pvar);
  enque_handler(pvar, SSH_SMSG_EXITSTATUS, handle_exit);
  enque_handler(pvar, SSH_SMSG_STDOUT_DATA, handle_data);
  enque_handler(pvar, SSH_SMSG_STDERR_DATA, handle_data);
  enque_handler(pvar, SSH_MSG_CHANNEL_DATA, handle_channel_data);
  enque_handler(pvar, SSH_MSG_CHANNEL_INPUT_EOF, handle_channel_input_eof);
  enque_handler(pvar, SSH_MSG_CHANNEL_OUTPUT_CLOSED, handle_channel_output_eof);
  enque_handler(pvar, SSH_MSG_PORT_OPEN, handle_channel_open);
  enque_handler(pvar, SSH_SMSG_X11_OPEN, handle_X11_channel_open);

        PostMessage(pvar->cv.HWin, WM_USER_NOTIFY_CONNRESULT, (WPARAM)1, (LPARAM)pvar
);

  return FALSE;
}

static BOOL handle_pty_failure(PTInstVar pvar) {
  notify_nonfatal_error(pvar, "サーバは仮想端末を割り当てることができませんでした．"
    "端末に関する問題が発生するかもしれません．");
  return handle_pty_success(pvar);
}

static void prep_pty(PTInstVar pvar) {
  int len = strlen("vt100");
  unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_CMSG_REQUEST_PTY,
    4 + len + 16 + sizeof(ssh_ttymodes));
  static const int msgs[] = { SSH_SMSG_SUCCESS, SSH_SMSG_FAILURE };
  static const SSHPacketHandler handlers[]
    = { handle_pty_success, handle_pty_failure };

  set_uint32(outmsg, len);
  memcpy(outmsg + 4, "vt100", len);
  set_uint32(outmsg + 4 + len, pvar->ssh_state.win_rows);
  set_uint32(outmsg + 4 + len + 4, pvar->ssh_state.win_cols);
  set_uint32(outmsg + 4 + len + 8, 0);
  set_uint32(outmsg + 4 + len + 12, 0);
  memcpy(outmsg + 4 + len + 16, ssh_ttymodes, sizeof(ssh_ttymodes));
  finish_send_packet(pvar);

  enque_handlers(pvar, 2, msgs, handlers);

  begin_send_packet(pvar, SSH_CMSG_EXEC_SHELL, 0);
  finish_send_packet(pvar);
```



```
}

static void prep_forwarding(PTInstVar pvar) {
  FWD_prep_forwarding(pvar);
  prep_pty(pvar);
}

static void enable_compression(PTInstVar pvar) {
  pvar->ssh_state.compress_stream.zalloc = NULL;
  pvar->ssh_state.compress_stream.zfree = NULL;
  pvar->ssh_state.compress_stream.opaque = NULL;
  if (deflateInit(&pvar->ssh_state.compress_stream, pvar->ssh_state.compression_level
) != Z_OK) {
    notify_fatal_error(pvar, "圧縮処理の初期化中にエラーが発生しました．\n"
      "接続を閉じます．");
    return;
  } else {
    pvar->ssh_state.compressing = TRUE;
  }

  pvar->ssh_state.decompress_stream.zalloc = NULL;
  pvar->ssh_state.decompress_stream.zfree = NULL;
  pvar->ssh_state.decompress_stream.opaque = NULL;
  if (inflateInit(&pvar->ssh_state.decompress_stream) != Z_OK) {
    deflateEnd(&pvar->ssh_state.compress_stream);
    notify_fatal_error(pvar, "圧縮処理の初期化中にエラーが発生しました．\n"
      "接続を閉じます．");
    return;
  } else {
    pvar->ssh_state.decompressing = TRUE;
    buf_ensure_size(&pvar->ssh_state.postdecompress_inbuf,
      &pvar->ssh_state.postdecompress_inbuflen, 1000);
  }
}

static BOOL handle_enable_compression(PTInstVar pvar) {
  enable_compression(pvar);
  prep_forwarding(pvar);
  return FALSE;
}

static BOOL handle_disable_compression(PTInstVar pvar) {
  prep_forwarding(pvar);
  return FALSE;
}

static void prep_compression(PTInstVar pvar) {
  if (pvar->session_settings.CompressionLevel > 0) {
    static const int msgs[] = { SSH_SMSG_SUCCESS, SSH_SMSG_FAILURE };
    static const SSHPacketHandler handlers[]
      = { handle_enable_compression, handle_disable_compression };

    unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_CMSG_REQUEST_COMPRESSION
, 4);

    set_uint32(outmsg, pvar->session_settings.CompressionLevel);
    finish_send_packet(pvar);

    pvar->ssh_state.compression_level = pvar->session_settings.CompressionLevel;

    enque_handlers(pvar, 2, msgs, handlers);
  } else {
    prep_forwarding(pvar);
  }
}
```

```
static void enque_simple_auth_handlers(PTInstVar pvar) {
  static const int msgs[] = { SSH_SMSG_SUCCESS, SSH_SMSG_FAILURE };
  static const SSHPacketHandler handlers[]
    = { handle_auth_success, handle_auth_failure };

  enque_handlers(pvar, 2, msgs, handlers);
}

static BOOL handle_rsa_challenge(PTInstVar pvar) {
  int challenge_bytes;

  if (!grab_payload(pvar, 2)) {
    return FALSE;
  }

  challenge_bytes = get_mpint_len(pvar, 0);

  if (grab_payload(pvar, challenge_bytes)) {
    unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_CMSG_AUTH_RSA_RESPONSE,
16);

    if (CRYPT_generate_RSA_challenge_response(pvar, pvar->ssh_state.payload + 2,
      challenge_bytes, outmsg)) {
      AUTH_destroy_cur_cred(pvar);
      finish_send_packet(pvar);

      enque_simple_auth_handlers(pvar);
    } else {
      notify_fatal_error(pvar, "RSAチャレンジの復号化中にエラーが発生しました．\n"
        "おそらく鍵ファイルが破損しています．");
    }
  }

  return FALSE;
}

#define OBFUSCATING_ROUND_TO 32

static int obfuscating_round_up(PTInstVar pvar, int size) {
  return (size + OBFUSCATING_ROUND_TO - 1) & ~(OBFUSCATING_ROUND_TO - 1);
}

static void try_send_credentials(PTInstVar pvar) {
  if ((pvar->ssh_state.status_flags & STATUS_DONT_SEND_CREDENTIALS) == 0) {
    AUTHCred FAR * cred = AUTH_get_cur_cred(pvar);
    static const int RSA_msgs[] = { SSH_SMSG_AUTH_RSA_CHALLENGE, SSH_SMSG_FAILURE };
    static const SSHPacketHandler RSA_handlers[]
      = { handle_rsa_challenge, handle_rsa_auth_refused };
    static const int TIS_msgs[] = { SSH_SMSG_AUTH_TIS_CHALLENGE, SSH_SMSG_FAILURE };
    static const SSHPacketHandler TIS_handlers[]
      = { handle_TIS_challenge, handle_auth_failure };

    switch (cred->method) {
    case SSH_AUTH_NONE: return;
    case SSH_AUTH_PASSWORD: {
      int len = strlen(cred->password);
      // Round up password length to discourage traffic analysis
      int obfuscated_len = obfuscating_round_up(pvar, len);
      unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_CMSG_AUTH_PASSWORD, 4
+ obfuscated_len);

      notify_verbose_message(pvar, "Trying PASSWORD authentication...", LOG_LEVEL_VER
BOSE);

      set_uint32(outmsg, obfuscated_len);
      memcpy(outmsg + 4, cred->password, len);
```



```
      memset(outmsg + 4 + len, 0, obfuscated_len - len);
      AUTH_destroy_cur_cred(pvar);
      enque_simple_auth_handlers(pvar);
      break;
    }
    case SSH_AUTH_RHOSTS: {
      int len = strlen(cred->rhosts_client_user);
      unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_CMSG_AUTH_RHOSTS, 4 +
len);

      notify_verbose_message(pvar, "Trying RHOSTS authentication...", LOG_LEVEL_VERBO
SE);

      set_uint32(outmsg, len);
      memcpy(outmsg + 4, cred->rhosts_client_user, len);
      AUTH_destroy_cur_cred(pvar);
      enque_simple_auth_handlers(pvar);
      break;
    }
    case SSH_AUTH_RSA: {
      int len = BN_num_bytes(cred->key_pair->RSA_key->n);
      unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_CMSG_AUTH_RSA, 2 + len
);

      notify_verbose_message(pvar, "Trying RSA authentication...", LOG_LEVEL_VERBOSE)
;

      set_ushort16_MSBfirst(outmsg, len*8);
      BN_bn2bin(cred->key_pair->RSA_key->n, outmsg + 2);
      /* don't destroy the current credentials yet */
      enque_handlers(pvar, 2, RSA_msgs, RSA_handlers);
      break;
    }
    case SSH_AUTH_RHOSTS_RSA: {
      int mod_len = BN_num_bytes(cred->key_pair->RSA_key->n);
      int name_len = strlen(cred->rhosts_client_user);
      int exp_len = BN_num_bytes(cred->key_pair->RSA_key->e);
      int index;
      unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_CMSG_AUTH_RHOSTS_RSA,
12 + mod_len
        + name_len + exp_len);

      notify_verbose_message(pvar, "Trying RHOSTS+RSA authentication...", LOG_LEVEL_V
ERBOSE);

      set_uint32(outmsg, name_len);
      memcpy(outmsg + 4, cred->rhosts_client_user, name_len);
      index = 4 + name_len;

      set_uint32(outmsg + index, 8*mod_len);
      set_ushort16_MSBfirst(outmsg + index + 4, 8*exp_len);
      BN_bn2bin(cred->key_pair->RSA_key->e, outmsg + index + 6);
      index += 6 + exp_len;

      set_ushort16_MSBfirst(outmsg + index, 8*mod_len);
      BN_bn2bin(cred->key_pair->RSA_key->n, outmsg + index + 2);
      /* don't destroy the current credentials yet */
      enque_handlers(pvar, 2, RSA_msgs, RSA_handlers);
      break;
    }
    case SSH_AUTH_TIS: {
      if (cred->password == NULL) {
        unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_CMSG_AUTH_TIS, 0);

        notify_verbose_message(pvar, "Trying TIS authentication...", LOG_LEVEL_VERBOS
E);
```

```
        enque_handlers(pvar, 2, TIS_msgs, TIS_handlers);
      } else {
        int len = strlen(cred->password);
        int obfuscated_len = obfuscating_round_up(pvar, len);
        unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_CMSG_AUTH_TIS_RESPON
SE, 4 + obfuscated_len);

        notify_verbose_message(pvar, "Sending TIS response", LOG_LEVEL_VERBOSE);

        set_uint32(outmsg, obfuscated_len);
        memcpy(outmsg + 4, cred->password, len);
        memset(outmsg + 4 + len, 0, obfuscated_len - len);
        enque_simple_auth_handlers(pvar);
      }
      AUTH_destroy_cur_cred(pvar);
      break;
    }
    default:
      notify_fatal_error(pvar, "内部エラー: サポートされていない認証方式です．");
      return;
    }

    finish_send_packet(pvar);
    destroy_packet_buf(pvar);

    pvar->ssh_state.status_flags |= STATUS_DONT_SEND_CREDENTIALS;
  }
}

static void try_send_user_name(PTInstVar pvar) {
  if ((pvar->ssh_state.status_flags & STATUS_DONT_SEND_USER_NAME) == 0) {
    char FAR * username = AUTH_get_user_name(pvar);

    if (username != NULL) {
      int len = strlen(username);
      int obfuscated_len = obfuscating_round_up(pvar, len);
      unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_CMSG_USER, 4 + obfusca
ted_len);
      char buf[1024] = "Sending user name: ";
      static const int msgs[] = { SSH_SMSG_SUCCESS, SSH_SMSG_FAILURE };
      static const SSHPacketHandler handlers[]
        = { handle_noauth_success, handle_auth_required };

      set_uint32(outmsg, obfuscated_len);
      memcpy(outmsg + 4, username, len);
      memset(outmsg + 4 + len, 0, obfuscated_len - len);
      finish_send_packet(pvar);

      pvar->ssh_state.status_flags |= STATUS_DONT_SEND_USER_NAME;

      strncpy(buf + strlen(buf), username, sizeof(buf) - strlen(buf) - 2);
      buf[sizeof(buf) - 1] = 0;
      notify_verbose_message(pvar, buf, LOG_LEVEL_VERBOSE);

      enque_handlers(pvar, 2, msgs, handlers);
    }
  }
}

static void send_session_key(PTInstVar pvar) {
  int encrypted_session_key_len = CRYPT_get_encrypted_session_key_len(pvar);
  unsigned char FAR * outmsg;

  if (!CRYPT_choose_ciphers(pvar)) return;
  outmsg = begin_send_packet(pvar, SSH_CMSG_SESSION_KEY, 15 + encrypted_session_key_l
en);
```



```
  outmsg[0] = (unsigned char)CRYPT_get_sender_cipher(pvar);
  memcpy(outmsg + 1, CRYPT_get_server_cookie(pvar), 8); /* antispoofing cookie */
  outmsg[9] = (unsigned char)(encrypted_session_key_len >> 5);
  outmsg[10] = (unsigned char)(encrypted_session_key_len << 3);
  if (!CRYPT_choose_session_key(pvar, outmsg + 11)) return;
  set_uint32(outmsg + 11 + encrypted_session_key_len,
    SSH_PROTOFLAG_SCREEN_NUMBER | SSH_PROTOFLAG_HOST_IN_FWD_OPEN);
  finish_send_packet(pvar);

  if (!CRYPT_start_encryption(pvar)) return;
  notify_established_secure_connection(pvar);

  enque_handler(pvar, SSH_SMSG_SUCCESS, handle_crypt_success);

  pvar->ssh_state.status_flags &= ~STATUS_DONT_SEND_USER_NAME;
  try_send_user_name(pvar);
}

/************************
   END of message handlers
   ************************/

void SSH_init(PTInstVar pvar) {
  int i;

  buf_create(&pvar->ssh_state.outbuf, &pvar->ssh_state.outbuflen);
  buf_create(&pvar->ssh_state.precompress_outbuf, &pvar->ssh_state.precompress_outbuf
len);
  buf_create(&pvar->ssh_state.postdecompress_inbuf, &pvar->ssh_state.postdecompress_i
nbuflen);
  pvar->ssh_state.payload = NULL;
  pvar->ssh_state.compressing = FALSE;
  pvar->ssh_state.decompressing = FALSE;
  pvar->ssh_state.status_flags = STATUS_DONT_SEND_USER_NAME | STATUS_DONT_SEND_CREDEN
TIALS;
  pvar->ssh_state.payload_datalen = 0;
  pvar->ssh_state.hostname = NULL;
  pvar->ssh_state.server_ID = NULL;
  pvar->ssh_state.receiver_sequence_number = 0;
  pvar->ssh_state.sender_sequence_number = 0;
  for (i = 0; i < NUM_ELEM(pvar->ssh_state.packet_handlers); i++) {
    pvar->ssh_state.packet_handlers[i] = NULL;
  }
}

void SSH_open(PTInstVar pvar) {
  pvar->ssh_state.hostname = _strdup(pvar->HostName);
/*
保留
*/
  pvar->ssh_state.win_cols = 80;
  pvar->ssh_state.win_rows = 25;

}

void SSH_notify_disconnecting(PTInstVar pvar, char FAR * reason) {
  int len = reason == NULL ? 0 : strlen(reason);
  unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_MSG_DISCONNECT, len + 4);

  set_uint32(outmsg, len);
  if (reason != NULL) {
    memcpy(outmsg + 4, reason, len);
  }
  finish_send_packet(pvar);
}
```

```
void SSH_notify_host_OK(PTInstVar pvar) {
  if ((pvar->ssh_state.status_flags & STATUS_HOST_OK) == 0) {
    pvar->ssh_state.status_flags |= STATUS_HOST_OK;
    send_session_key(pvar);
  }
}

void SSH_notify_win_size(PTInstVar pvar, int cols, int rows) {
  pvar->ssh_state.win_cols = cols;
  pvar->ssh_state.win_rows = rows;

  if (get_handler(pvar, SSH_SMSG_STDOUT_DATA) == handle_data) {
    unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_CMSG_WINDOW_SIZE, 16);

    set_uint32(outmsg, rows);
    set_uint32(outmsg + 4, cols);
    set_uint32(outmsg + 8, 0);
    set_uint32(outmsg + 12, 0);
    finish_send_packet(pvar);
  }
}

int SSH_get_min_packet_size(PTInstVar pvar) {
  if (SSHv1(pvar)) {
    return 12;
  } else {
    int block_size = CRYPT_get_decryption_block_size(pvar);

    return max(16, block_size);
  }
}

/* data is guaranteed to be at least SSH_get_min_packet_size bytes long
   at least 5 bytes must be decrypted */
void SSH_predecrpyt_packet(PTInstVar pvar, char FAR * data) {
  if (SSHv2(pvar)) {
    CRYPT_decrypt(pvar, data, get_predecryption_amount(pvar));
  }
}

int SSH_get_clear_MAC_size(PTInstVar pvar) {
  if (SSHv1(pvar)) {
    return 0;
  } else {
    return CRYPT_get_receiver_MAC_size(pvar);
  }
}

void SSH_notify_user_name(PTInstVar pvar) {
  try_send_user_name(pvar);
}

void SSH_notify_cred(PTInstVar pvar) {
  try_send_credentials(pvar);
}

void SSH_send(PTInstVar pvar, unsigned char const FAR * buf, int buflen) {
  if (get_handler(pvar, SSH_SMSG_STDOUT_DATA) != handle_data) {
    return;
  }

  while (buflen > 0) {
    int len = buflen > SSH_MAX_SEND_PACKET_SIZE ? SSH_MAX_SEND_PACKET_SIZE : buflen;
    unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_CMSG_STDIN_DATA, 4 + len
);
```



```
    set_uint32(outmsg, len);

    if (pvar->ssh_state.compressing) {
      buf_ensure_size(&pvar->ssh_state.outbuf, &pvar->ssh_state.outbuflen, len + (len
 >> 6) + 50);
      pvar->ssh_state.compress_stream.next_in = pvar->ssh_state.precompress_outbuf;
      pvar->ssh_state.compress_stream.avail_in = 5;
      pvar->ssh_state.compress_stream.next_out = pvar->ssh_state.outbuf + 12;
      pvar->ssh_state.compress_stream.avail_out = pvar->ssh_state.outbuflen - 12;

      if (deflate(&pvar->ssh_state.compress_stream, Z_NO_FLUSH) != Z_OK) {
        notify_fatal_error(pvar, "データの圧縮中にエラーが発生しました．");
        return;
      }

      pvar->ssh_state.compress_stream.next_in = (unsigned char FAR *)buf;
      pvar->ssh_state.compress_stream.avail_in = len;

      if (deflate(&pvar->ssh_state.compress_stream, Z_SYNC_FLUSH) != Z_OK) {
        notify_fatal_error(pvar, "データの圧縮中にエラーが発生しました．");
        return;
      }
    } else {
      memcpy(outmsg + 4, buf, len);
    }

    finish_send_packet_special(pvar, 1);

    buflen -= len;
    buf += len;
  }
}

int SSH_extract_payload(PTInstVar pvar, unsigned char FAR * dest, int len) {
  int num_bytes = pvar->ssh_state.payload_datalen;

  if (num_bytes > len) {
    num_bytes = len;
  }

  if (!pvar->ssh_state.decompressing) {
    memcpy(dest, pvar->ssh_state.payload + pvar->ssh_state.payload_datastart, num_byt
es);
    pvar->ssh_state.payload_datastart += num_bytes;
  } else if (num_bytes > 0) {
    pvar->ssh_state.decompress_stream.next_out = dest;
    pvar->ssh_state.decompress_stream.avail_out = num_bytes;

    if (inflate(&pvar->ssh_state.decompress_stream, Z_SYNC_FLUSH) != Z_OK) {
      notify_fatal_error(pvar, "受信したパケットの圧縮データが不正です．");
      return 0;
    }
  }

  pvar->ssh_state.payload_datalen -= num_bytes;

  return num_bytes;
}

void SSH_get_compression_info(PTInstVar pvar, char FAR * dest, int len) {
  char buf[1024];
  char buf2[1024];

  if (pvar->ssh_state.compressing) {
    unsigned long total_in = pvar->ssh_state.compress_stream.total_in;
    unsigned long total_out = pvar->ssh_state.compress_stream.total_out;
```

```
    if (total_out > 0) {
      _snprintf(buf, sizeof(buf), "レベル %d, 圧縮比 %.1f (%ld:%ld)",
        pvar->ssh_state.compression_level, ((double)total_in)/total_out,
        total_in, total_out);
    } else {
      _snprintf(buf, sizeof(buf), "レベル %d", pvar->ssh_state.compression_level);
    }
  } else {
    strcpy(buf, "なし");
  }
  buf[sizeof(buf) - 1] = 0;

  if (pvar->ssh_state.decompressing) {
    unsigned long total_in = pvar->ssh_state.decompress_stream.total_in;
    unsigned long total_out = pvar->ssh_state.decompress_stream.total_out;

    if (total_in > 0) {
      _snprintf(buf2, sizeof(buf2), "レベル %d, 圧縮比 %.1f (%ld:%ld)",
        pvar->ssh_state.compression_level, ((double)total_out)/total_in,
        total_out, total_in);
    } else {
      _snprintf(buf2, sizeof(buf2), "レベル %d", pvar->ssh_state.compression_level);
    }
  } else {
    strcpy(buf2, "なし");
  }
  buf2[sizeof(buf2) - 1] = 0;

  _snprintf(dest, len, "アップロード %s / ダウンロード %s", buf, buf2);
  dest[len - 1] = 0;
}

void SSH_get_server_ID_info(PTInstVar pvar, char FAR * dest, int len) {
  strncpy(dest, pvar->ssh_state.server_ID == NULL ? "不明"
    : pvar->ssh_state.server_ID, len);
  dest[len - 1] = 0;
}

void SSH_get_protocol_version_info(PTInstVar pvar, char FAR * dest, int len) {
  if (pvar->protocol_major == 0) {
    strncpy(dest, "不明", len);
  } else {
    _snprintf(dest, len, "%d.%d", pvar->protocol_major, pvar->protocol_minor);
  }
  dest[len - 1] = 0;
}

void SSH_end(PTInstVar pvar) {
  int i;

  for (i = 0; i < 256; i++) {
    SSHPacketHandlerItem FAR * first_item = pvar->ssh_state.packet_handlers[i];

    if (first_item != NULL) {
      SSHPacketHandlerItem FAR * item = first_item;

      do {
        SSHPacketHandlerItem FAR * cur_item = item;

        item = item->next_for_message;
        free(cur_item);
      } while (item != first_item);
    }
    pvar->ssh_state.packet_handlers[i] = NULL;
  }
```



```
  free(pvar->ssh_state.hostname);
  pvar->ssh_state.hostname = NULL;
  free(pvar->ssh_state.server_ID);
  pvar->ssh_state.server_ID = NULL;
  buf_destroy(&pvar->ssh_state.outbuf, &pvar->ssh_state.outbuflen);
  buf_destroy(&pvar->ssh_state.precompress_outbuf, &pvar->ssh_state.precompress_outbu
flen);
  buf_destroy(&pvar->ssh_state.postdecompress_inbuf, &pvar->ssh_state.postdecompress_
inbuflen);

  if (pvar->ssh_state.compressing) {
    deflateEnd(&pvar->ssh_state.compress_stream);
    pvar->ssh_state.compressing = FALSE;
  }
  if (pvar->ssh_state.decompressing) {
    inflateEnd(&pvar->ssh_state.decompress_stream);
    pvar->ssh_state.decompressing = FALSE;
  }
}

/* support for port forwarding */
void SSH_channel_send(PTInstVar pvar, uint32 remote_channel_num,
                      unsigned char FAR * buf, int len) {
  unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_MSG_CHANNEL_DATA, 8 + len)
;

  set_uint32(outmsg, remote_channel_num);
  set_uint32(outmsg + 4, len);

  if (pvar->ssh_state.compressing) {
    buf_ensure_size(&pvar->ssh_state.outbuf, &pvar->ssh_state.outbuflen, len + (len >
> 6) + 50);
    pvar->ssh_state.compress_stream.next_in = pvar->ssh_state.precompress_outbuf;
    pvar->ssh_state.compress_stream.avail_in = 9;
    pvar->ssh_state.compress_stream.next_out = pvar->ssh_state.outbuf + 12;
    pvar->ssh_state.compress_stream.avail_out = pvar->ssh_state.outbuflen - 12;

    if (deflate(&pvar->ssh_state.compress_stream, Z_NO_FLUSH) != Z_OK) {
      notify_fatal_error(pvar, "データの圧縮中にエラーが発生しました．");
      return;
    }

    pvar->ssh_state.compress_stream.next_in = (unsigned char FAR *)buf;
    pvar->ssh_state.compress_stream.avail_in = len;

    if (deflate(&pvar->ssh_state.compress_stream, Z_SYNC_FLUSH) != Z_OK) {
      notify_fatal_error(pvar, "データの圧縮中にエラーが発生しました．");
      return;
    }
  } else {
    memcpy(outmsg + 8, buf, len);
  }

  finish_send_packet_special(pvar, 1);
}

void SSH_fail_channel_open(PTInstVar pvar, uint32 remote_channel_num) {
  unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_MSG_CHANNEL_OPEN_FAILURE,
4);

  set_uint32(outmsg, remote_channel_num);
  finish_send_packet(pvar);
}

void SSH_confirm_channel_open(PTInstVar pvar, uint32 remote_channel_num, uint32 local
```

```
_channel_num) {
  unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_MSG_CHANNEL_OPEN_CONFIRMAT
ION, 8);

  set_uint32(outmsg, remote_channel_num);
  set_uint32(outmsg + 4, local_channel_num);
  finish_send_packet(pvar);
}

void SSH_channel_output_eof(PTInstVar pvar, uint32 remote_channel_num) {
  unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_MSG_CHANNEL_OUTPUT_CLOSED,
 4);

  set_uint32(outmsg, remote_channel_num);
  finish_send_packet(pvar);
}

void SSH_channel_input_eof(PTInstVar pvar, uint32 remote_channel_num) {
  unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_MSG_CHANNEL_INPUT_EOF, 4);

  set_uint32(outmsg, remote_channel_num);
  finish_send_packet(pvar);
}

void SSH_request_forwarding(PTInstVar pvar, int from_server_port,
                            char FAR * to_local_host, int to_local_port) {
  int host_len = strlen(to_local_host);
  unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_CMSG_PORT_FORWARD_REQUEST,
    12 + host_len);

  set_uint32(outmsg, from_server_port);
  set_uint32(outmsg + 4, host_len);
  memcpy(outmsg + 8, to_local_host, host_len);
  set_uint32(outmsg + 8 + host_len, to_local_port);
  finish_send_packet(pvar);

  enque_forwarding_request_handlers(pvar);
}

void SSH_request_X11_forwarding(PTInstVar pvar,
                                char FAR * auth_protocol, unsigned char FAR * auth_da
ta,
                                int auth_data_len, int screen_num) {
  int protocol_len = strlen(auth_protocol);
  int data_len = auth_data_len*2;
  unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_CMSG_X11_REQUEST_FORWARDIN
G,
    12 + protocol_len + data_len);
  int i;
  char FAR * auth_data_ptr;

  set_uint32(outmsg, protocol_len);
  memcpy(outmsg + 4, auth_protocol, protocol_len);
  set_uint32(outmsg + 4 + protocol_len, data_len);
  auth_data_ptr = outmsg + 8 + protocol_len;
  for (i = 0; i < auth_data_len; i++) {
    sprintf(auth_data_ptr + i*2, "%.2x", auth_data[i]);
  }
  set_uint32(outmsg + 8 + protocol_len + data_len, screen_num);

  finish_send_packet(pvar);

  enque_forwarding_request_handlers(pvar);
}

void SSH_open_channel(PTInstVar pvar, uint32 local_channel_num,
```



```
                      char FAR * to_remote_host, int to_remote_port, char FAR * origi
nator) {
  static const int msgs[]
    = { SSH_MSG_CHANNEL_OPEN_CONFIRMATION, SSH_MSG_CHANNEL_OPEN_FAILURE };
  static const SSHPacketHandler handlers[]
    = { handle_channel_open_confirmation, handle_channel_open_failure };

  int host_len = strlen(to_remote_host);

  if ((pvar->ssh_state.server_protocol_flags & SSH_PROTOFLAG_HOST_IN_FWD_OPEN) != 0)
{
    int originator_len = strlen(originator);
    unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_MSG_PORT_OPEN,
      16 + host_len + originator_len);

    set_uint32(outmsg, local_channel_num);
    set_uint32(outmsg + 4, host_len);
    memcpy(outmsg + 8, to_remote_host, host_len);
    set_uint32(outmsg + 8 + host_len, to_remote_port);
    set_uint32(outmsg + 12 + host_len, originator_len);
    memcpy(outmsg + 16 + host_len, originator, originator_len);
  } else {
    unsigned char FAR * outmsg = begin_send_packet(pvar, SSH_MSG_PORT_OPEN,
      12 + host_len);

    set_uint32(outmsg, local_channel_num);
    set_uint32(outmsg + 4, host_len);
    memcpy(outmsg + 8, to_remote_host, host_len);
    set_uint32(outmsg + 8 + host_len, to_remote_port);
  }

  finish_send_packet(pvar);

  enque_handlers(pvar, 2, msgs, handlers);
}
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
This code is copyright (C) 1998-1999 Robert O'Callahan.
See LICENSE.TXT for the license.
*/

#ifndef __SSH_H
#define __SSH_H

#include "zlib.h"

/* Some of this code has been adapted from Ian Goldberg's Pilot SSH */

typedef enum {
    SSH_MSG_NONE, SSH_MSG_DISCONNECT, SSH_SMSG_PUBLIC_KEY,
    SSH_CMSG_SESSION_KEY, SSH_CMSG_USER, SSH_CMSG_AUTH_RHOSTS,
    SSH_CMSG_AUTH_RSA, SSH_SMSG_AUTH_RSA_CHALLENGE,
    SSH_CMSG_AUTH_RSA_RESPONSE, SSH_CMSG_AUTH_PASSWORD,
    SSH_CMSG_REQUEST_PTY, SSH_CMSG_WINDOW_SIZE, SSH_CMSG_EXEC_SHELL,
    SSH_CMSG_EXEC_CMD, SSH_SMSG_SUCCESS, SSH_SMSG_FAILURE,
    SSH_CMSG_STDIN_DATA, SSH_SMSG_STDOUT_DATA, SSH_SMSG_STDERR_DATA,
    SSH_CMSG_EOF, SSH_SMSG_EXITSTATUS,
    SSH_MSG_CHANNEL_OPEN_CONFIRMATION, SSH_MSG_CHANNEL_OPEN_FAILURE,
    SSH_MSG_CHANNEL_DATA, SSH_MSG_CHANNEL_INPUT_EOF,
    SSH_MSG_CHANNEL_OUTPUT_CLOSED, SSH_MSG_OBSOLETED0,
    SSH_SMSG_X11_OPEN, SSH_CMSG_PORT_FORWARD_REQUEST, SSH_MSG_PORT_OPEN,
    SSH_CMSG_AGENT_REQUEST_FORWARDING, SSH_SMSG_AGENT_OPEN,
    SSH_MSG_IGNORE, SSH_CMSG_EXIT_CONFIRMATION,
    SSH_CMSG_X11_REQUEST_FORWARDING, SSH_CMSG_AUTH_RHOSTS_RSA,
    SSH_MSG_DEBUG, SSH_CMSG_REQUEST_COMPRESSION,
    SSH_CMSG_MAX_PACKET_SIZE, SSH_CMSG_AUTH_TIS,
    SSH_SMSG_AUTH_TIS_CHALLENGE, SSH_CMSG_AUTH_TIS_RESPONSE,
    SSH_CMSG_AUTH_KERBEROS, SSH_SMSG_AUTH_KERBEROS_RESPONSE
} SSHMessage;
```



```
typedef enum {
    SSH_CIPHER_NONE, SSH_CIPHER_IDEA, SSH_CIPHER_DES, SSH_CIPHER_3DES,
    SSH_CIPHER_TSS, SSH_CIPHER_RC4, SSH_CIPHER_BLOWFISH
} SSHCipher;

#define SSH_CIPHER_MAX SSH_CIPHER_BLOWFISH

typedef enum {
    SSH_AUTH_NONE, SSH_AUTH_RHOSTS, SSH_AUTH_RSA, SSH_AUTH_PASSWORD,
    SSH_AUTH_RHOSTS_RSA, SSH_AUTH_TIS, SSH_AUTH_KERBEROS
} SSHAuthMethod;

/* we don't support Kerberos at this time */
#define SSH_AUTH_MAX SSH_AUTH_TIS

typedef enum {
    SSH_GENERIC_AUTHENTICATION, SSH_TIS_AUTHENTICATION
} SSHAuthMode;

#define SSH_PROTOFLAG_SCREEN_NUMBER 1
#define SSH_PROTOFLAG_HOST_IN_FWD_OPEN 2

#define SSH_MAX_SEND_PACKET_SIZE   250000

/* SSH2 constants */

/* SSH2 messages */
#define SSH2_MSG_DISCONNECT             1
#define SSH2_MSG_IGNORE                 2
#define SSH2_MSG_UNIMPLEMENTED          3
#define SSH2_MSG_DEBUG                  4
#define SSH2_MSG_SERVICE_REQUEST        5
#define SSH2_MSG_SERVICE_ACCEPT         6

#define SSH2_MSG_KEXINIT                20
#define SSH2_MSG_NEWKEYS                21

#define SSH2_MSG_KEXDH_INIT             30
#define SSH2_MSG_KEXDH_REPLY            31

#define SSH2_MSG_USERAUTH_REQUEST            50
#define SSH2_MSG_USERAUTH_FAILURE            51
#define SSH2_MSG_USERAUTH_SUCCESS            52
#define SSH2_MSG_USERAUTH_BANNER             53

#define SSH2_MSG_USERAUTH_PK_OK              60
#define SSH2_MSG_USERAUTH_PASSWD_CHANGEREQ   60

#define SSH2_MSG_GLOBAL_REQUEST                  80
#define SSH2_MSG_REQUEST_SUCCESS                 81
#define SSH2_MSG_REQUEST_FAILURE                 82
#define SSH2_MSG_CHANNEL_OPEN                    90
#define SSH2_MSG_CHANNEL_OPEN_CONFIRMATION       91
#define SSH2_MSG_CHANNEL_OPEN_FAILURE            92
#define SSH2_MSG_CHANNEL_WINDOW_ADJUST           93
#define SSH2_MSG_CHANNEL_DATA                    94
#define SSH2_MSG_CHANNEL_EXTENDED_DATA           95
#define SSH2_MSG_CHANNEL_EOF                     96
#define SSH2_MSG_CHANNEL_CLOSE                   97
#define SSH2_MSG_CHANNEL_REQUEST                 98
#define SSH2_MSG_CHANNEL_SUCCESS                 99
#define SSH2_MSG_CHANNEL_FAILURE                 100

/* SSH2 miscellaneous constants */
#define SSH2_DISCONNECT_HOST_NOT_ALLOWED_TO_CONNECT      1
```

```
#define SSH2_DISCONNECT_PROTOCOL_ERROR                  2
#define SSH2_DISCONNECT_KEY_EXCHANGE_FAILED             3
#define SSH2_DISCONNECT_HOST_AUTHENTICATION_FAILED      4
#define SSH2_DISCONNECT_MAC_ERROR                       5
#define SSH2_DISCONNECT_COMPRESSION_ERROR               6
#define SSH2_DISCONNECT_SERVICE_NOT_AVAILABLE           7
#define SSH2_DISCONNECT_PROTOCOL_VERSION_NOT_SUPPORTED  8
#define SSH2_DISCONNECT_HOST_KEY_NOT_VERIFIABLE         9
#define SSH2_DISCONNECT_CONNECTION_LOST                 10
#define SSH2_DISCONNECT_BY_APPLICATION                  11

#define SSH2_OPEN_ADMINISTRATIVELY_PROHIBITED    1
#define SSH2_OPEN_CONNECT_FAILED                 2
#define SSH2_OPEN_UNKNOWN_CHANNEL_TYPE           3
#define SSH2_OPEN_RESOURCE_SHORTAGE              4

/* The packet handler returns TRUE to keep the handler in place,
   FALSE to remove the handler. */
typedef BOOL (* SSHPacketHandler)(PTInstVar pvar);

typedef struct _SSHPacketHandlerItem SSHPacketHandlerItem;
struct _SSHPacketHandlerItem {
  SSHPacketHandler handler;
  /* Circular list of handlers for given message */
  SSHPacketHandlerItem FAR * next_for_message;
  SSHPacketHandlerItem FAR * last_for_message;
  /* Circular list of handlers in set */
  SSHPacketHandlerItem FAR * next_in_set;
  int active_for_message;
};

typedef struct {
  char FAR * hostname;

  int server_protocol_flags;
  char FAR * server_ID;

  /* This buffer is used to hold the outgoing data, and encrypted in-place
     here if necessary. */
  unsigned char FAR * outbuf;
  long outbuflen;
  /* This buffer is used by the SSH protocol processing to store uncompressed
     packet data for compression. User data is never streamed through here;
     it is compressed directly from the user's buffer. */
  unsigned char FAR * precompress_outbuf;
  long precompress_outbuflen;
  /* this is the length of the packet data, including the type header */
  long outgoing_packet_len;

  /* This buffer is used by the SSH protocol processing to store decompressed
     packet data. User data is never streamed through here; it is decompressed
     directly to the user's buffer. */
  unsigned char FAR * postdecompress_inbuf;
  long postdecompress_inbuflen;

  unsigned char FAR * payload;
  long payload_grabbed;
  long payloadlen;
  long payload_datastart;
  long payload_datalen;

  uint32 receiver_sequence_number;
  uint32 sender_sequence_number;

  z_stream compress_stream;
  z_stream decompress_stream;
```



```
  BOOL compressing;
  BOOL decompressing;
  int compression_level;

  SSHPacketHandlerItem FAR * packet_handlers[256];
  int status_flags;

  int win_cols;
  int win_rows;
} SSHState;

#define STATUS_DONT_SEND_USER_NAME            0x01
#define STATUS_EXPECTING_COMPRESSION_RESPONSE 0x02
#define STATUS_DONT_SEND_CREDENTIALS          0x04
#define STATUS_HOST_OK                        0x08
#define STATUS_INTERACTIVE                    0x10
#define STATUS_IN_PARTIAL_ID_STRING           0x20

void SSH_init(PTInstVar pvar);
void SSH_open(PTInstVar pvar);
void SSH_notify_disconnecting(PTInstVar pvar, char FAR * reason);
/* SSH_handle_server_ID returns TRUE iff a valid ID string has been
   received. If it returns FALSE, we need to keep looking for another
   ID string. */
BOOL SSH_handle_server_ID(PTInstVar pvar, char FAR * ID, int ID_len);
/* SSH_handle_packet requires NO PAYLOAD on entry.
   'len' is the size of the packet: payload + padding (+ CRC for SSHv1)
   'padding' is the size of the padding.
   'data' points to the start of the packet data (the length field)
*/
void SSH_handle_packet(PTInstVar pvar, char FAR * data, int len, int padding);
void SSH_notify_win_size(PTInstVar pvar, int cols, int rows);
void SSH_notify_user_name(PTInstVar pvar);
void SSH_notify_cred(PTInstVar pvar);
void SSH_notify_host_OK(PTInstVar pvar);
void SSH_send(PTInstVar pvar, unsigned char const FAR * buf, int buflen);
/* SSH_extract_payload returns number of bytes extracted */
int SSH_extract_payload(PTInstVar pvar, unsigned char FAR * dest, int len);
void SSH_end(PTInstVar pvar);

void SSH_get_server_ID_info(PTInstVar pvar, char FAR * dest, int len);
void SSH_get_protocol_version_info(PTInstVar pvar, char FAR * dest, int len);
void SSH_get_compression_info(PTInstVar pvar, char FAR * dest, int len);

/* len must be <= SSH_MAX_SEND_PACKET_SIZE */
void SSH_fail_channel_open(PTInstVar pvar, uint32 remote_channel_num);
void SSH_confirm_channel_open(PTInstVar pvar, uint32 remote_channel_num, uint32 local
_channel_num);
void SSH_request_forwarding(PTInstVar pvar, int from_server_port,
  char FAR * to_local_host, int to_local_port);
void SSH_request_X11_forwarding(PTInstVar pvar,
  char FAR * auth_protocol, unsigned char FAR * auth_data, int auth_data_len, int scr
een_num);

/* auxiliary SSH2 interfaces for pkt.c */
int SSH_get_min_packet_size(PTInstVar pvar);
/* data is guaranteed to be at least SSH_get_min_packet_size bytes long
   at least 5 bytes must be decrypted */
void SSH_predecrpyt_packet(PTInstVar pvar, char FAR * data);
int SSH_get_clear_MAC_size(PTInstVar pvar);


#ifdef __cplusplus
extern "C" {
#endif
```

```
void SSH_open_channel(PTInstVar pvar, uint32 local_channel_num,
  char FAR * to_remote_host, int to_remote_port, char FAR * originator);
void SSH_channel_output_eof(PTInstVar pvar, uint32 remote_channel_num);
void SSH_channel_input_eof(PTInstVar pvar, uint32 remote_channel_num);
void SSH_channel_send(PTInstVar pvar, uint32 remote_channel_num,
                      unsigned char FAR * buf, int len);

#ifdef __cplusplus
}
#endif


#define SSH_is_any_payload(pvar) ((pvar)->ssh_state.payload_datalen > 0)
#define SSH_get_host_name(pvar) ((pvar)->ssh_state.hostname)
#define SSH_get_compression_level(pvar) ((pvar)->ssh_state.compressing ? (pvar)->ts_S
SH_CompressionLevel : 0)

#endif
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/* Teraterm extension mechanism
   Robert O'Callahan (roc+tt@cs.cmu.edu)

   Teraterm by Takashi Teranishi (teranishi@rikaxp.riken.go.jp)
*/

#include "ttxssh.h"
#include "fwdui.h"
#include "util.h"

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <io.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <time.h>

#include "resource.h"
#include <commctrl.h>
#include <commdlg.h>
#include <winsock2.h>

#define MATCH_STR(s, o) _strnicmp((s), (o), NUM_ELEM(o) - 1)

/* This extension implements SSH, so we choose a load order in the
   "protocols" range. */
//#define ORDER 2500

  /* WIN32 allows multiple instances of a DLL */
//  static TInstVar InstVar;
#define GET_VAR()   ;
```



```
HANDLE hInst = NULL;

/*
This code makes lots of assumptions about the order in which Teraterm
does things, and how. A key assumption is that the Notification window
passed into WSAAsyncSelect is the main terminal window. We also assume
that the socket used in the first WSAconnect is the main session socket.
*/

void init_TTSSH(PTInstVar pvar) {
  pvar->socket = INVALID_SOCKET;
  pvar->NotificationWindow = NULL;
  pvar->hostdlg_activated = FALSE;
  pvar->socket = INVALID_SOCKET;
  pvar->NotificationWindow = NULL;
  pvar->protocol_major = 0;
  pvar->protocol_minor = 0;

  PKT_init(pvar);
  SSH_init(pvar);
  CRYPT_init(pvar);
  AUTH_init(pvar);
  HOSTS_init(pvar);
  FWD_init(pvar);
  FWDUI_init(pvar);
}

static void uninit_TTSSH(PTInstVar pvar) {
  SSH_end(pvar);
  PKT_end(pvar);
  AUTH_end(pvar);
  CRYPT_end(pvar);
  HOSTS_end(pvar);
  FWD_end(pvar);
  FWDUI_end(pvar);
/*
  if (pvar->OldLargeIcon != NULL) {
    PostMessage(pvar->NotificationWindow, WM_SETICON, ICON_BIG, (LPARAM)pvar->OldLarg
eIcon);
    pvar->OldLargeIcon = NULL;
  }
  if (pvar->OldSmallIcon != NULL) {
    PostMessage(pvar->NotificationWindow, WM_SETICON, ICON_SMALL, (LPARAM)pvar->OldSm
allIcon);
    pvar->OldSmallIcon = NULL;
  }
  */
}


static void normalize_cipher_order(char FAR * buf) {
  char ciphers_listed[SSH_CIPHER_MAX + 1];
  char ciphers_allowed[SSH_CIPHER_MAX + 1];
  int i, j;

  /* SSH_CIPHER_NONE means that all ciphers below that one are disabled.
     We *never* allow no encryption. */
  static char default_ciphers[] = {
    SSH_CIPHER_3DES,
    SSH_CIPHER_NONE,
    SSH_CIPHER_DES, SSH_CIPHER_BLOWFISH, SSH_CIPHER_IDEA, SSH_CIPHER_RC4
  };

  memset(ciphers_listed, 0, sizeof(ciphers_listed));

  memset(ciphers_allowed, 0, sizeof(ciphers_allowed));
```

```
  for (i = 0; i < NUM_ELEM(default_ciphers); i++) {
    ciphers_allowed[default_ciphers[i]] = 1;
  }

  for (i = 0; buf[i] != 0; i++) {
    int cipher_num = buf[i] - '0';

    if (cipher_num < 0 || cipher_num > SSH_CIPHER_MAX
      || !ciphers_allowed[cipher_num] || ciphers_listed[cipher_num]) {
      memmove(buf + i, buf + i + 1, strlen(buf + i + 1) + 1);
      i--;
    } else {
      ciphers_listed[cipher_num] = 1;
    }
  }

  for (j = 0; j < NUM_ELEM(default_ciphers); j++) {
    int cipher_num = default_ciphers[j];

    if (!ciphers_listed[cipher_num]) {
      buf[i] = cipher_num + '0';
      i++;
    }
  }

  buf[i] = 0;
}

/* Remove local settings from the shared memory block. */
static void clear_local_settings(PTInstVar pvar) {
//  pvar->ts_SSH->TryDefaultAuth = FALSE;
}

static BOOL read_BOOL_option(PCHAR fileName, char FAR * keyName,
  BOOL def) {
  char buf[1024];

  buf[0] = 0;
  GetPrivateProfileString("TTSSH", keyName, "", buf, sizeof(buf), fileName);
  if (buf[0] == 0) {
    return def;
  } else {
    return atoi(buf) != 0 ||
      stricmp(buf, "yes") == 0 || stricmp(buf, "y") == 0;
  }
}

static void read_string_option(PCHAR fileName, char FAR * keyName,
  char FAR * def, char FAR * buf, int bufSize) {

  buf[0] = 0;
  GetPrivateProfileString("TTSSH", keyName, def, buf, bufSize, fileName);
}

static void read_ssh_options(PTInstVar pvar, PCHAR fileName) {
/*
  char buf[1024];
  TS_SSH FAR * settings = pvar->ts_SSH;

#define READ_STD_STRING_OPTION(name) \
  read_string_option(fileName, #name, "", settings->name, sizeof(settings->name))

  settings->Enabled = read_BOOL_option(fileName, "Enabled", FALSE);

  buf[0] = 0;
  GetPrivateProfileString("TTSSH", "Compression", "", buf, sizeof(buf), fileName);
```



```
  settings->CompressionLevel = atoi(buf);
  if (settings->CompressionLevel < 0 ||
    settings->CompressionLevel > 9) {
    settings->CompressionLevel = 0;
  }
  READ_STD_STRING_OPTION(DefaultUserName);
  READ_STD_STRING_OPTION(DefaultForwarding);
  READ_STD_STRING_OPTION(DefaultRhostsLocalUserName);
  READ_STD_STRING_OPTION(DefaultRhostsHostPrivateKeyFile);
  READ_STD_STRING_OPTION(DefaultRSAPrivateKeyFile);
  settings->TryDefaultAuth = read_BOOL_option(fileName, "TryDefaultAuth", FALSE);

  READ_STD_STRING_OPTION(CipherOrder);
  normalize_cipher_order(settings->CipherOrder);

  read_string_option(fileName, "KnownHostsFiles", "ssh_known_hosts",
    settings->KnownHostsFiles, sizeof(settings->KnownHostsFiles));

  buf[0] = 0;
  GetPrivateProfileString("TTSSH", "DefaultAuthMethod", "", buf, sizeof(buf), fileNam
e);
  settings->DefaultAuthMethod = atoi(buf);
  if (settings->DefaultAuthMethod == SSH_AUTH_RHOSTS_RSA)
          settings->DefaultAuthMethod = SSH_AUTH_RHOSTS;
  else if (settings->DefaultAuthMethod != SSH_AUTH_PASSWORD
    && settings->DefaultAuthMethod != SSH_AUTH_RSA
    && settings->DefaultAuthMethod != SSH_AUTH_RHOSTS) {
    /* this default can never be SSH_AUTH_RHOSTS_RSA because that is not a
       selection in the dialog box; SSH_AUTH_RHOSTS_RSA is automatically chosen
       when the dialog box has rhosts selected and an host private key file
       is supplied. */
/*
    settings->DefaultAuthMethod = SSH_AUTH_PASSWORD;
  }

  buf[0] = 0;
  GetPrivateProfileString("TTSSH", "LogLevel", "", buf, sizeof(buf), fileName);
  settings->LogLevel = atoi(buf);

  buf[0] = 0;
  GetPrivateProfileString("TTSSH", "WriteBufferSize", "", buf, sizeof(buf), fileName)
;
  settings->WriteBufferSize = atoi(buf);
  if (settings->WriteBufferSize <= 0) {
    settings->WriteBufferSize = 2*1024*1024;
  }

  settings->LocalForwardingIdentityCheck =
    read_BOOL_option(fileName, "LocalForwardingIdentityCheck", TRUE);

  clear_local_settings(pvar);
*/
}

static void write_ssh_options(PTInstVar pvar, PCHAR fileName, TS_SSH FAR * settings)
{
  char buf[1024];

  WritePrivateProfileString("TTSSH", "Enabled",
    settings->Enabled ? "1" : "0", fileName);

  _itoa(settings->CompressionLevel, buf, 10);
  WritePrivateProfileString("TTSSH", "Compression", buf, fileName);

  WritePrivateProfileString("TTSSH", "DefaultUserName",
    settings->DefaultUserName, fileName);
```

```
  WritePrivateProfileString("TTSSH", "DefaultForwarding",
    settings->DefaultForwarding, fileName);

  WritePrivateProfileString("TTSSH", "CipherOrder",
    settings->CipherOrder, fileName);

  WritePrivateProfileString("TTSSH", "KnownHostsFiles",
    settings->KnownHostsFiles, fileName);

  WritePrivateProfileString("TTSSH", "DefaultRhostsLocalUserName",
    settings->DefaultRhostsLocalUserName, fileName);

  WritePrivateProfileString("TTSSH", "DefaultRhostsHostPrivateKeyFile",
    settings->DefaultRhostsHostPrivateKeyFile, fileName);

  WritePrivateProfileString("TTSSH", "DefaultRSAPrivateKeyFile",
    settings->DefaultRSAPrivateKeyFile, fileName);

  _itoa(settings->DefaultAuthMethod, buf, 10);
  WritePrivateProfileString("TTSSH", "DefaultAuthMethod", buf, fileName);

  WritePrivateProfileString("TTSSH", "TryDefaultAuth",
    settings->TryDefaultAuth ? "1" : "0", fileName);

  _itoa(settings->LogLevel, buf, 10);
  WritePrivateProfileString("TTSSH", "LogLevel", buf, fileName);

  _itoa(settings->WriteBufferSize, buf, 10);
  WritePrivateProfileString("TTSSH", "WriteBufferSize", buf, fileName);

  WritePrivateProfileString("TTSSH", "LocalForwardingIdentityCheck",
    settings->LocalForwardingIdentityCheck ? "1" : "0", fileName);
}

static unsigned short find_local_port(PTInstVar pvar) {
  int tries;

  if (pvar->session_settings.DefaultAuthMethod != SSH_AUTH_RHOSTS) {
    return 0;
  }

  /* The random numbers here are only used to try to get fresh
     ports across runs (dangling ports can cause bind errors
     if we're unlucky). They do not need to be (and are not)
     cryptographically strong.
  */
  srand((unsigned)GetTickCount());

  for (tries = 20; tries > 0; tries--) {
    SOCKET connecter = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in connecter_addr;

    connecter_addr.sin_family = AF_INET;
    connecter_addr.sin_port = (unsigned)rand() % 512 + 512;
    connecter_addr.sin_addr.s_addr = htonl(INADDR_ANY);

    if (connecter == INVALID_SOCKET) {
      return 0;
    }

    if (bind(connecter, (struct sockaddr FAR *)&connecter_addr, sizeof(connecter_addr
)) != SOCKET_ERROR) {
      closesocket(connecter);
      return connecter_addr.sin_port;
    } else if (WSAGetLastError() != WSAEADDRINUSE) {
```



```
      closesocket(connecter);
      return 0;
    }

    closesocket(connecter);
  }

  return 0;
}

int PASCAL FAR TTXconnect( PTInstVar pvar,
  SOCKET s, const struct sockaddr FAR *name, int namelen) {

  if (pvar->socket == INVALID_SOCKET) {
    struct sockaddr_in addr;

    pvar->socket = s;

    addr.sin_family = AF_INET;
    addr.sin_port = htons(find_local_port(pvar));
    addr.sin_addr.s_addr = INADDR_ANY;
    memset(addr.sin_zero, 0, sizeof(addr.sin_zero));

    bind(s, (struct sockaddr FAR *)&addr, sizeof(addr));
  }

  return connect(s, name, namelen);
}

int PASCAL FAR TTXWSAAsyncSelect( PTInstVar pvar,
  SOCKET s, HWND hWnd, u_int wMsg,long lEvent) {

  if (s == pvar->socket) {
    pvar->notification_events = lEvent;
    pvar->notification_msg = wMsg;

    if (pvar->NotificationWindow == NULL) {
      pvar->NotificationWindow = hWnd;
      AUTH_advance_to_next_cred(pvar);
    }
  }

  return WSAAsyncSelect(s, hWnd, wMsg, lEvent);
}

int PASCAL FAR TTXrecv(PTInstVar pvar, SOCKET s, char FAR * buf, int len, int flags)
{

  if (s == pvar->socket) { // SSH
    return PKT_recv(pvar, buf, len);
  } else {
    return recv(s, buf, len, flags);
  }
}

int PASCAL FAR TTXsend(PTInstVar pvar, SOCKET s, char const FAR * buf, int len, int f
lags) {

  if (s == pvar->socket) {
    SSH_send(pvar, buf, len);
    return len;
  } else {
    return send(s, buf, len, flags);
  }
}
```

```
void notify_established_secure_connection(PTInstVar pvar) {
      /*
  if (SecureIcon == NULL) {
    SecureIcon = LoadIcon(hInst, MAKEINTRESOURCE(IDI_SECURETT));
  }

  if (SecureIcon != NULL) {
    pvar->OldSmallIcon = (HICON)SendMessage(pvar->NotificationWindow, WM_GETICON, ICO
N_SMALL, 0);
    pvar->OldLargeIcon = (HICON)SendMessage(pvar->NotificationWindow, WM_GETICON, ICO
N_BIG, 0);
    PostMessage(pvar->NotificationWindow, WM_SETICON, ICON_BIG, (LPARAM)SecureIcon);
    PostMessage(pvar->NotificationWindow, WM_SETICON, ICON_SMALL, (LPARAM)SecureIcon)
;
  }
  */

  notify_verbose_message(pvar, "Entering secure mode", LOG_LEVEL_VERBOSE);
}

void notify_closed_connection(PTInstVar pvar) {
  PostMessage(pvar->NotificationWindow, WM_USER_COMMNOTIFY,
    pvar->socket, MAKELPARAM(FD_CLOSE, 0));

  SSH_notify_disconnecting(pvar, NULL);
  AUTH_notify_disconnecting(pvar);
  HOSTS_notify_disconnecting(pvar);
}

static void add_err_msg(PTInstVar pvar, char FAR * msg) {
  if (pvar->err_msg != NULL) {
    char FAR * buf = (char FAR *)malloc(strlen(pvar->err_msg) + 3 + strlen(msg));

    strcpy(buf, pvar->err_msg);
    strcat(buf, "\n\n");
    strcat(buf, msg);
    free(pvar->err_msg);
    pvar->err_msg = buf;
  } else {
    pvar->err_msg = _strdup(msg);
  }
}

void notify_nonfatal_error(PTInstVar pvar, char FAR * msg) {
  if (!pvar->showing_err) {
    PostMessage(pvar->NotificationWindow, WM_COMMAND, ID_SSHASYNCMESSAGEBOX, 0);
  }
  if (msg[0] != 0) {
    notify_verbose_message(pvar, msg, LOG_LEVEL_ERROR);
    add_err_msg(pvar, msg);
  }
}

void notify_fatal_error(PTInstVar pvar, char FAR * msg) {
  if (msg[0] != 0) {
    notify_verbose_message(pvar, msg, LOG_LEVEL_FATAL);
    add_err_msg(pvar, msg);
  }

  if (!pvar->fatal_error) {
    pvar->fatal_error = TRUE;

    SSH_notify_disconnecting(pvar, msg);
    AUTH_notify_disconnecting(pvar);
    HOSTS_notify_disconnecting(pvar);
```



```
  PostMessage(pvar->NotificationWindow, WM_USER_COMMNOTIFY,
      pvar->socket, MAKELPARAM(FD_CLOSE, WSAGetLastError()));

  }
}

void notify_verbose_message(PTInstVar pvar, char FAR * msg, int level) {
  if (level <= pvar->session_settings.LogLevel) {
    char buf[1024];
    int file;

    get_teraterm_dir_relative_name(buf, NUM_ELEM(buf), "TTSSH.LOG");
    file = _open(buf, _O_RDWR | _O_APPEND | _O_CREAT | _O_TEXT,
      _S_IREAD | _S_IWRITE);

    if (file >= 0) {
      _write(file, msg, strlen(msg));
      _write(file, "\n", 1);
      _close(file);
    }
  }
}

void PASCAL FAR TTXOpenTCP(PTInstVar pvar) {

  if (pvar->settings.Enabled) {
    char buf[1024] = "\nInitiating SSH session at ";
    struct tm FAR * newtime;
    time_t long_time;

    pvar->session_settings = pvar->settings;

    time(&long_time);
    newtime = localtime(&long_time);
    strcat(buf, asctime(newtime));
    buf[strlen(buf) - 1] = 0;
    notify_verbose_message(pvar, buf, LOG_LEVEL_VERBOSE);

    FWDUI_load_settings(pvar);

/*
    pvar->Precv = *hooks->Precv;
    pvar->Psend = *hooks->Psend;
    pvar->PWSAAsyncSelect = *hooks->PWSAAsyncSelect;
    pvar->Pconnect = *hooks->Pconnect;
    pvar->PWSAGetLastError = *hooks->PWSAGetLastError;

    *hooks->Precv = TTXrecv;
    *hooks->Psend = TTXsend;
    *hooks->PWSAAsyncSelect = TTXWSAAsyncSelect;
    *hooks->Pconnect = TTXconnect;
*/
    SSH_open(pvar);
    HOSTS_open(pvar);
    FWDUI_open(pvar);
  }
}

void PASCAL FAR TTXCloseTCP(PTInstVar pvar) {

  if (pvar->session_settings.Enabled) {
    pvar->socket = INVALID_SOCKET;

    notify_verbose_message(pvar, "Terminating SSH session...", LOG_LEVEL_VERBOSE);
/*
    *hooks->Precv = pvar->Precv;
```

```
    *hooks->Psend = pvar->Psend;
    *hooks->PWSAAsyncSelect = pvar->PWSAAsyncSelect;
    *hooks->Pconnect = pvar->Pconnect;
*/
  }

  uninit_TTSSH(pvar);
  init_TTSSH(pvar);
}

static void enable_dlg_items(HWND dlg, int from, int to, BOOL enabled) {
  for (; from <= to; from++) {
    EnableWindow(GetDlgItem(dlg, from), enabled);
  }
}
/*
static BOOL CALLBACK TTXHostDlg(HWND dlg, UINT msg, WPARAM wParam, LPARAM lParam) {
  PGetHNRec GetHNRec;
  char EntName[7];
  char TempHost[HostNameMaxLength+1];
  WORD i, j, w;
  BOOL Ok;

  GET_VAR();

  switch (msg) {
  case WM_INITDIALOG:
    GetHNRec = (PGetHNRec)lParam;
    SetWindowLong(dlg, DWL_USER, lParam);

    if ( GetHNRec->PortType==IdFile )
      GetHNRec->PortType = IdTCPIP;
    CheckRadioButton(dlg,IDC_HOSTTCPIP,IDC_HOSTSERIAL,
      IDC_HOSTTCPIP + GetHNRec->PortType - 1);

    strcpy(EntName,"Host");

    i = 1;
    do {
      sprintf(&EntName[4],"%d",i);
      GetPrivateProfileString("Hosts",EntName,"",
        TempHost,sizeof(TempHost),GetHNRec->SetupFN);
      if ( strlen(TempHost) > 0 )
        SendDlgItemMessage(dlg, IDC_HOSTNAME, CB_ADDSTRING,
                           0, (LPARAM)TempHost);
      i++;
    } while ((i <= 99) && (strlen(TempHost)>0));

    SendDlgItemMessage(dlg, IDC_HOSTNAME, EM_LIMITTEXT,
      HostNameMaxLength-1, 0);

    SendDlgItemMessage(dlg, IDC_HOSTNAME, CB_SETCURSEL,0,0);

    CheckRadioButton(dlg, IDC_HOSTTELNET, IDC_HOSTOTHER,
      pvar->settings.Enabled ? IDC_HOSTSSH : GetHNRec->Telnet ? IDC_HOSTTELNET : IDC_
HOSTOTHER);
    SendDlgItemMessage(dlg, IDC_HOSTTCPPORT, EM_LIMITTEXT,5,0);
    SetDlgItemInt(dlg,IDC_HOSTTCPPORT,GetHNRec->TCPPort,FALSE);

    j = 0;
    w = 1;
    strcpy(EntName,"COM");
    for (i=1; i<=GetHNRec->MaxComPort ;i++)
    {
      sprintf(&EntName[3],"%d",i);
      SendDlgItemMessage(dlg, IDC_HOSTCOM, CB_ADDSTRING,
```



```
        0, (LPARAM)EntName);
      j++;
      if (GetHNRec->ComPort==i) w = j;
    }
    if (j>0)
      SendDlgItemMessage(dlg, IDC_HOSTCOM, CB_SETCURSEL,w-1,0);
    else
      GetHNRec->PortType = IdTCPIP;

    if ( GetHNRec->PortType==IdTCPIP )
      enable_dlg_items(dlg,IDC_HOSTCOMLABEL,IDC_HOSTCOM, FALSE);
    else
      enable_dlg_items(dlg,IDC_HOSTNAMELABEL,IDC_HOSTTCPPORT, FALSE);

    return TRUE;

  case WM_COMMAND:
    switch (LOWORD(wParam)) {
    case IDOK:
      GetHNRec = (PGetHNRec)GetWindowLong(dlg,DWL_USER);
      if ( GetHNRec!=NULL )
      {
        if (IsDlgButtonChecked(dlg,IDC_HOSTTCPIP)) {
          i = GetDlgItemInt(dlg,IDC_HOSTTCPPORT,&Ok,FALSE);
          if (Ok) {
            GetHNRec->TCPPort = i;
          } else {
            MessageBox(dlg, "Teraterm", "TCPポートは数値で指定して下さい．",
              MB_OK | MB_ICONEXCLAMATION);
            return TRUE;
          }
          GetHNRec->PortType = IdTCPIP;
          GetDlgItemText(dlg, IDC_HOSTNAME, GetHNRec->HostName, HostNameMaxLength);
          GetHNRec->Telnet = FALSE;
          pvar->hostdlg_activated = TRUE;
          pvar->hostdlg_Enabled = FALSE;
          if (IsDlgButtonChecked(dlg,IDC_HOSTTELNET)) {
            GetHNRec->Telnet = TRUE;
          } else if (IsDlgButtonChecked(dlg,IDC_HOSTSSH)) {
            pvar->hostdlg_Enabled = TRUE;
          }
        } else {
          GetHNRec->PortType = IdSerial;
          GetHNRec->HostName[0] = 0;
          memset(EntName,0,sizeof(EntName));
          GetDlgItemText(dlg, IDC_HOSTCOM, EntName, sizeof(EntName)-1);
          GetHNRec->ComPort = (BYTE)(EntName[3])-0x30;
          if (strlen(EntName)>4)
            GetHNRec->ComPort = GetHNRec->ComPort*10 + (BYTE)(EntName[4])-0x30;
        }
      }
      EndDialog(dlg, 1);
      return TRUE;

    case IDCANCEL:
      EndDialog(dlg, 0);
      return TRUE;

    case IDC_HOSTTCPIP:
      enable_dlg_items(dlg,IDC_HOSTNAMELABEL,IDC_HOSTTCPPORT, TRUE);
      enable_dlg_items(dlg,IDC_HOSTCOMLABEL,IDC_HOSTCOM, FALSE);

      return TRUE;

    case IDC_HOSTSERIAL:
      enable_dlg_items(dlg,IDC_HOSTCOMLABEL,IDC_HOSTCOM, TRUE);
```

```
      enable_dlg_items(dlg,IDC_HOSTNAMELABEL,IDC_HOSTTCPPORT, FALSE);
      return TRUE;

    case IDC_HOSTTELNET:
    case IDC_HOSTSSH:
    case IDC_HOSTOTHER:
      GetHNRec = (PGetHNRec)GetWindowLong(dlg,DWL_USER);

      if (IsDlgButtonChecked(dlg,IDC_HOSTTELNET)) {
        if ( GetHNRec!=NULL )
          SetDlgItemInt(dlg,IDC_HOSTTCPPORT,GetHNRec->TelPort,FALSE);
      } else if (IsDlgButtonChecked(dlg,IDC_HOSTSSH)) {
        SetDlgItemInt(dlg, IDC_HOSTTCPPORT, 22, FALSE);
      }
      return TRUE;

    case IDC_HOSTHELP:
      PostMessage(GetParent(dlg),WM_USER_DLGHELP2,0,0);
    }
  }
  return FALSE;
}
*/
/*
static BOOL FAR PASCAL TTXGetHostName(HWND parent, PGetHNRec rec) {
  return (BOOL)DialogBoxParam(hInst, MAKEINTRESOURCE(IDD_HOSTDLG),
    parent, TTXHostDlg, (LONG)rec);
}
*/
void FAR PASCAL TTXWriteINIFile(PCHAR fileName) {
/*
  (pvar->WriteIniFile)(fileName);
  clear_local_settings(pvar);
  notify_verbose_message(pvar, "Writing INI file", LOG_LEVEL_VERBOSE);
  write_ssh_options(pvar, fileName, pvar->ts_SSH);
*/
}

static void read_ssh_options_from_user_file(PTInstVar pvar,
  char FAR * user_file_name) {
  if (user_file_name[0] == '.') {
    read_ssh_options(pvar, user_file_name);
  } else {
    char buf[1024];

    get_teraterm_dir_relative_name(buf, sizeof(buf), user_file_name);
    read_ssh_options(pvar, buf);
  }

//  pvar->settings = *pvar->ts_SSH;
  FWDUI_load_settings(pvar);
}


void PASCAL FAR TTXSetWinSize(PTInstVar pvar, int rows, int cols) {

  SSH_notify_win_size(pvar, cols, rows);
}

static void append_about_text(HWND dlg, char FAR * prefix, char FAR * msg) {
  SendDlgItemMessage(dlg, IDC_ABOUTTEXT, EM_REPLACESEL, 0, (LPARAM)prefix);
  SendDlgItemMessage(dlg, IDC_ABOUTTEXT, EM_REPLACESEL, 0, (LPARAM)msg);
  SendDlgItemMessage(dlg, IDC_ABOUTTEXT, EM_REPLACESEL, 0, (LPARAM)(char FAR *)"\r\n"
);
}
```



```
static void init_about_dlg(PTInstVar pvar, HWND dlg) {
  char buf[1024];
  HANDLE hEditBoxWnd;

  if (pvar->socket != INVALID_SOCKET) {
    SSH_get_server_ID_info(pvar, buf, sizeof(buf));
    append_about_text(dlg, "サーバID: ", buf);
    SSH_get_protocol_version_info(pvar, buf, sizeof(buf));
    append_about_text(dlg, "プロトコルのバージョン: ", buf);
    CRYPT_get_cipher_info(pvar, buf, sizeof(buf));
    append_about_text(dlg, "暗号化方式: ", buf);
    CRYPT_get_server_key_info(pvar, buf, sizeof(buf));
    append_about_text(dlg, "サーバ鍵: ", buf);
    AUTH_get_auth_info(pvar, buf, sizeof(buf));
    append_about_text(dlg, "認証方法: ", buf);
    SSH_get_compression_info(pvar, buf, sizeof(buf));
    append_about_text(dlg, "圧縮状況: ", buf);
        hEditBoxWnd = GetDlgItem( dlg, IDC_ABOUTTEXT );
/*
        PostMessage( hEditBoxWnd, EM_FMTLINES, TRUE, 0 );
        PostMessage( hEditBoxWnd, EM_SETSEL, 0, -1 );
        PostMessage( hEditBoxWnd, WM_CUT, 0, 0 );
        PostMessage( hEditBoxWnd, WM_PASTE, 0, 0 );
*/
        // 選択解除
    PostMessage( hEditBoxWnd, EM_SETSEL, -1, -1 );
  }
}

BOOL CALLBACK TTXAboutDlg(HWND dlg, UINT msg, WPARAM wParam, LPARAM lParam) {
  switch (msg) {
  case WM_INITDIALOG:
    init_about_dlg((PTInstVar)lParam, dlg);
    return TRUE;
  case WM_COMMAND:
    switch (LOWORD(wParam)) {
    case IDOK:
      EndDialog(dlg, 1);
      return TRUE;
    case IDCANCEL: /* there isn't a cancel button, but other Windows
                      UI things can send this message */
      EndDialog(dlg, 0);
      return TRUE;
    }
    break;
  }

  return FALSE;
}

static char FAR * get_cipher_name(int cipher) {
  switch (cipher) {
  case SSH_CIPHER_NONE:
    return "<以下の暗号アルゴリズムは使用されません>";
  case SSH_CIPHER_RC4:
    return "RC4";
  case SSH_CIPHER_3DES:
    return "3DES";
  case SSH_CIPHER_DES:
    return "DES";
  case SSH_CIPHER_IDEA:
    return "IDEA";
  case SSH_CIPHER_TSS:
    return "TSS";
  case SSH_CIPHER_BLOWFISH:
```

```
    return "Blowfish";
  default:
    return NULL;
  }
}

static void set_move_button_status(HWND dlg) {
  HWND cipherControl = GetDlgItem(dlg, IDC_SSHCIPHERPREFS);
  int curPos = (int)SendMessage(cipherControl, LB_GETCURSEL, 0, 0);
  int maxPos = (int)SendMessage(cipherControl, LB_GETCOUNT, 0, 0) - 1;

  EnableWindow(GetDlgItem(dlg, IDC_SSHMOVECIPHERUP), curPos > 0 && curPos <= maxPos);
  EnableWindow(GetDlgItem(dlg, IDC_SSHMOVECIPHERDOWN), curPos >= 0 && curPos < maxPos
);
}

static void init_setup_dlg(PTInstVar pvar, HWND dlg) {
  HWND compressionControl = GetDlgItem(dlg, IDC_SSHCOMPRESSIONLEVEL);
  HWND cipherControl = GetDlgItem(dlg, IDC_SSHCIPHERPREFS);
  int i;
  int ch;

  SendMessage(compressionControl, TBM_SETRANGE, TRUE, MAKELONG(0, 9));
  SendMessage(compressionControl, TBM_SETPOS, TRUE, pvar->settings.CompressionLevel);

  normalize_cipher_order(pvar->settings.CipherOrder);
  for (i = 0; pvar->settings.CipherOrder[i] != 0; i++) {
    int cipher = pvar->settings.CipherOrder[i] - '0';
    char FAR * name = get_cipher_name(cipher);

    if (name != NULL) {
      SendMessage(cipherControl, LB_ADDSTRING, 0, (LPARAM)name);
    }
  }

  SendMessage(cipherControl, LB_SETCURSEL, 0, 0);
  set_move_button_status(dlg);

  for (i = 0; (ch = pvar->settings.KnownHostsFiles[i]) != 0 && ch != ';'; i++) {
  }
  if (ch != 0) {
    pvar->settings.KnownHostsFiles[i] = 0;
    SetDlgItemText(dlg, IDC_READWRITEFILENAME, pvar->settings.KnownHostsFiles);
    pvar->settings.KnownHostsFiles[i] = ch;
    SetDlgItemText(dlg, IDC_READONLYFILENAME, pvar->settings.KnownHostsFiles + i + 1)
;
  } else {
    SetDlgItemText(dlg, IDC_READWRITEFILENAME, pvar->settings.KnownHostsFiles);
  }
}

void get_teraterm_dir_relative_name(char FAR * buf, int bufsize, char FAR * basename)
 {
  int filename_start = 0;
  int i;
  int ch;

  if (basename[0] == '\\' || basename[0] == '/'
    || (basename[0] != 0 && basename[1] == ':')) {
    strncpy(buf, basename, bufsize);
    buf[bufsize - 1] = 0;
    return;
  }

  GetModuleFileName(NULL, buf, bufsize);
  for (i = 0; (ch = buf[i]) != 0; i++) {
```



```
    if (ch == '\\' || ch == '/' || ch == ':') {
      filename_start = i + 1;
    }
  }

  if (bufsize > filename_start) {
    strncpy(buf + filename_start, basename, bufsize - filename_start);
  }
  buf[bufsize - 1] = 0;
}

int copy_teraterm_dir_relative_path(char FAR * dest, int destsize, char FAR * basenam
e) {
  char buf[1024];
  int filename_start = 0;
  int i;
  int ch, ch2;

  if (basename[0] != '\\' && basename[0] != '/' && (basename[0] == 0 || basename[1] !
= ':')) {
    strncpy(dest, basename, destsize);
    dest[destsize - 1] = 0;
    return strlen(dest);
  }

  GetModuleFileName(NULL, buf, sizeof(buf));
  for (i = 0; (ch = buf[i]) != 0; i++) {
    if (ch == '\\' || ch == '/' || ch == ':') {
      filename_start = i + 1;
    }
  }

  for (i = 0; i < filename_start; i++) {
    ch = toupper(buf[i]);
    ch2 = toupper(basename[i]);

    if (ch == ch2 || ((ch == '\\' || ch == '/') && (ch2 == '\\' || ch2 == '/'))) {
    } else {
      break;
    }
  }

  if (i == filename_start) {
    strncpy(dest, basename + i, destsize);
  } else {
    strncpy(dest, basename, destsize);
  }
  dest[destsize - 1] = 0;
  return strlen(dest);
}

static void complete_setup_dlg(PTInstVar pvar, HWND dlg) {
  char buf[4096];
  char buf2[1024];
  HWND compressionControl = GetDlgItem(dlg, IDC_SSHCOMPRESSIONLEVEL);
  HWND cipherControl = GetDlgItem(dlg, IDC_SSHCIPHERPREFS);
  int i, j, buf2index, bufindex;
  int count = (int)SendMessage(cipherControl, LB_GETCOUNT, 0, 0);

  pvar->settings.CompressionLevel = (int)SendMessage(compressionControl, TBM_GETPOS,
0, 0);

  buf2index = 0;
  for (i = 0; i < count; i++) {
    int len = SendMessage(cipherControl, LB_GETTEXTLEN, i, 0);
```

```
    if (len > 0 && len < sizeof(buf)) { /* should always be true */
      buf[0] = 0;
      SendMessage(cipherControl, LB_GETTEXT, i, (LPARAM)buf);
      for (j = 0; j <= SSH_CIPHER_MAX && strcmp(buf, get_cipher_name(j)) != 0; j++) {
      }
      if (j <= SSH_CIPHER_MAX) {
        buf2[buf2index] = '0' + j;
        buf2index++;
      }
    }
  }
  buf2[buf2index] = 0;
  normalize_cipher_order(buf2);
  strcpy(pvar->settings.CipherOrder, buf2);

  buf[0] = 0;
  GetDlgItemText(dlg, IDC_READWRITEFILENAME, buf, sizeof(buf));
  j = copy_teraterm_dir_relative_path(pvar->settings.KnownHostsFiles,
    sizeof(pvar->settings.KnownHostsFiles), buf);
  buf[0] = 0;
  bufindex = 0;
  GetDlgItemText(dlg, IDC_READONLYFILENAME, buf, sizeof(buf));
  for (i = 0; buf[i] != 0; i++) {
    if (buf[i] == ';') {
      buf[i] = 0;
      if (j < sizeof(pvar->settings.KnownHostsFiles) - 1) {
        pvar->settings.KnownHostsFiles[j] = ';';
        j++;
        j += copy_teraterm_dir_relative_path(pvar->settings.KnownHostsFiles + j,
          sizeof(pvar->settings.KnownHostsFiles) - j, buf + bufindex);
      }
      bufindex = i + 1;
    }
  }
  if (bufindex < i && j < sizeof(pvar->settings.KnownHostsFiles) - 1) {
    pvar->settings.KnownHostsFiles[j] = ';';
    j++;
    copy_teraterm_dir_relative_path(pvar->settings.KnownHostsFiles + j,
      sizeof(pvar->settings.KnownHostsFiles) - j, buf + bufindex);
  }
}

static void move_cur_sel_delta(HWND listbox, int delta) {
  int curPos = (int)SendMessage(listbox, LB_GETCURSEL, 0, 0);
  int maxPos = (int)SendMessage(listbox, LB_GETCOUNT, 0, 0) - 1;
  int newPos = curPos + delta;
  char buf[1024];

  if (curPos >= 0 && newPos >= 0 && newPos <= maxPos) {
    int len = SendMessage(listbox, LB_GETTEXTLEN, curPos, 0);

    if (len > 0 && len < sizeof(buf)) { /* should always be true */
      buf[0] = 0;
      SendMessage(listbox, LB_GETTEXT, curPos, (LPARAM)buf);
      SendMessage(listbox, LB_DELETESTRING, curPos, 0);
      SendMessage(listbox, LB_INSERTSTRING, newPos, (LPARAM)(char FAR *)buf);
      SendMessage(listbox, LB_SETCURSEL, newPos, 0);
    }
  }
}

static int get_keys_file_name(HWND parent, char FAR * buf, int bufsize, int readonly)
 {
#ifdef TERATERM32
  OPENFILENAME params;
  char fullname_buf[2048] = "ssh_known_hosts";
```



```
  params.lStructSize = sizeof(OPENFILENAME);
  params.hwndOwner = parent;
  params.lpstrFilter = NULL;
  params.lpstrCustomFilter = NULL;
  params.nFilterIndex = 0;
  buf[0] = 0;
  params.lpstrFile = fullname_buf;
  params.nMaxFile = sizeof(fullname_buf);
  params.lpstrFileTitle = NULL;
  params.lpstrInitialDir = NULL;
  params.lpstrTitle = readonly ? "読み取り専用known-hostsファイルの選択"
    : "読み書き可能known-hostsファイルの選択";
  params.Flags = (readonly ? OFN_FILEMUSTEXIST | OFN_PATHMUSTEXIST : 0)
    | OFN_HIDEREADONLY | (!readonly ? OFN_NOREADONLYRETURN : 0);
  params.lpstrDefExt = NULL;

  if (GetOpenFileName(&params) != 0) {
    copy_teraterm_dir_relative_path(buf, bufsize, fullname_buf);
    return 1;
  } else {
    int err = CommDlgExtendedError();

    if (err != 0) {
      char buf[1024];

      _snprintf(buf, sizeof(buf), "ファイルダイアログボックスを表示できませんでした:
エラー %d", err);
      buf[sizeof(buf) - 1] = 0;
      MessageBox(parent, buf, "SSHエラー", MB_OK | MB_ICONEXCLAMATION);
    }

    return 0;
  }
#else
  return 0;
#endif
}

static void choose_read_write_file(HWND dlg) {
  char buf[1024];

  if (get_keys_file_name(dlg, buf, sizeof(buf), 0)) {
    SetDlgItemText(dlg, IDC_READWRITEFILENAME, buf);
  }
}

static void choose_read_only_file(HWND dlg) {
  char buf[1024];
  char buf2[4096];

  if (get_keys_file_name(dlg, buf, sizeof(buf), 1)) {
    buf2[0] = 0;
    GetDlgItemText(dlg, IDC_READONLYFILENAME, buf2, sizeof(buf2));
    if (buf2[0] != 0 && buf2[strlen(buf2) - 1] != ';') {
      strncat(buf2, ";", sizeof(buf2));
    }
    strncat(buf2, buf, sizeof(buf2));
    SetDlgItemText(dlg, IDC_READONLYFILENAME, buf2);
  }
}

BOOL CALLBACK TTXSetupDlg(HWND dlg, UINT msg, WPARAM wParam, LPARAM lParam) {
  switch (msg) {
  case WM_INITDIALOG:
    SetWindowLong(dlg, DWL_USER, lParam);
```

```
    init_setup_dlg((PTInstVar)lParam, dlg);
    return TRUE;
  case WM_COMMAND:
    switch (LOWORD(wParam)) {
    case IDOK:
      complete_setup_dlg((PTInstVar)GetWindowLong(dlg, DWL_USER), dlg);
      EndDialog(dlg, 1);
      return TRUE;
    case IDCANCEL: /* there isn't a cancel button, but other Windows
                      UI things can send this message */
      EndDialog(dlg, 0);
      return TRUE;
    case IDC_SSHMOVECIPHERUP:
      move_cur_sel_delta(GetDlgItem(dlg, IDC_SSHCIPHERPREFS), -1);
      set_move_button_status(dlg);
      SetFocus(GetDlgItem(dlg, IDC_SSHCIPHERPREFS));
      return TRUE;
    case IDC_SSHMOVECIPHERDOWN:
      move_cur_sel_delta(GetDlgItem(dlg, IDC_SSHCIPHERPREFS), 1);
      set_move_button_status(dlg);
      SetFocus(GetDlgItem(dlg, IDC_SSHCIPHERPREFS));
      return TRUE;
    case IDC_SSHCIPHERPREFS:
      set_move_button_status(dlg);
      return TRUE;
    case IDC_CHOOSEREADWRITEFILE:
      choose_read_write_file(dlg);
      return TRUE;
    case IDC_CHOOSEREADONLYFILE:
      choose_read_only_file(dlg);
      return TRUE;
    }
    break;
  }

  return FALSE;
}

int PASCAL FAR TTXProcessCommand(PTInstVar pvar, HWND hWin, WORD cmd) {
  GET_VAR();

  if (pvar->fatal_error) {
    return 0;
  }

  switch (cmd) {
  case ID_ABOUTMENU:
    if (DialogBoxParam(hInst, MAKEINTRESOURCE(IDD_ABOUTDIALOG), hWin, TTXAboutDlg, (L
PARAM)pvar)
      == -1) {
      MessageBox(hWin, "SSHについてのウィンドウを表示できませんでした．", "SSHエラー"
, MB_OK | MB_ICONEXCLAMATION);
    }
    return 1;
  case ID_SSHAUTH:
    AUTH_do_cred_dialog(pvar);
    return 1;
  case ID_SSHSETUPMENU:
    if (DialogBoxParam(hInst, MAKEINTRESOURCE(IDD_SSHSETUP), hWin, TTXSetupDlg, (LPAR
AM)pvar)
      == -1) {
      MessageBox(hWin, "SSH設定ウィンドウを表示できませんでした．", "SSHエラー", MB_O
K | MB_ICONEXCLAMATION);
    }
    return 1;
  case ID_SSHAUTHSETUPMENU:
```



```
    AUTH_do_default_cred_dialog(pvar);
    return 1;
  case ID_SSHFWDSETUPMENU:
    FWDUI_do_forwarding_dialog(pvar);
    return 1;
  case ID_SSHUNKNOWNHOST:
    HOSTS_do_unknown_host_dialog(hWin, pvar);
    return 1;
  case ID_SSHDIFFERENTHOST:
    HOSTS_do_different_host_dialog(hWin, pvar);
    return 1;
  case ID_SSHASYNCMESSAGEBOX:
    if (pvar->err_msg != NULL) {
      char FAR * msg = pvar->err_msg;

      /* Could there be a buffer overrun bug anywhere in Win32
         MessageBox? Who knows? I'm paranoid. */
      if (strlen(msg) > 2048) {
        msg[2048] = 0;
      }

      pvar->showing_err = TRUE;
      pvar->err_msg = NULL;
      MessageBox(NULL, msg, "SSH", MB_TASKMODAL | MB_ICONEXCLAMATION);
      free(msg);
      pvar->showing_err = FALSE;
      if (pvar->err_msg != NULL) {
        PostMessage(hWin, WM_COMMAND, ID_SSHASYNCMESSAGEBOX, 0);
      } else {
        AUTH_notify_end_error(pvar);
      }
    }
    return 1;
  default:
    return 0;
  }
}


/* This function is called when Teraterm is quitting. You can use it to clean
   up.
   This function is called for each extension, in reverse load order (see
   below).
*/
void PASCAL FAR TTXEnd(PTInstVar pvar) {

  uninit_TTSSH(pvar);

  if (pvar->err_msg != NULL) {
  /* Could there be a buffer overrun bug anywhere in Win32
    MessageBox? Who knows? I'm paranoid. */
    if (strlen(pvar->err_msg) > 2048) {
      pvar->err_msg[2048] = 0;
    }

    MessageBox(NULL, pvar->err_msg, "SSH", MB_TASKMODAL | MB_ICONEXCLAMATION);

    free(pvar->err_msg);
    pvar->err_msg = NULL;
  }
}
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
This code is copyright (C) 1998-1999 Robert O'Callahan.
See LICENSE.TXT for the license.
*/

#ifndef __TTXSSH_H
#define __TTXSSH_H

#pragma warning(3 : 4035)

#include <stdlib.h>
#include <crtdbg.h>

#include <winsock2.h>
#define TERATERM32
#define MAXPATHLEN 256
#define TTVERSION (WORD)23
/*
#include "tttypes.h"
#include "ttplugin.h"
*/
typedef struct _TInstVar FAR * PTInstVar;

#include "util.h"
#include "pkt.h"
#include "ssh.h"
#include "auth.h"
#include "crypt.h"
#include "hosts.h"
#include "fwd.h"

extern HANDLE hInst; /* Instance handle of TTXSSH.DLL */
```



```
#define ID_ABOUTMENU 62501
#define ID_SSHAUTH   62502
#define ID_SSHSETUPMENU 62503
#define ID_SSHUNKNOWNHOST 62504
#define ID_SSHDIFFERENTHOST 62505
#define ID_SSHAUTHSETUPMENU 62506
#define ID_SSHASYNCMESSAGEBOX 62507
#define ID_SSHFWDSETUPMENU 62508



#define InBuffSize 1024
#define OutBuffSize 1024

typedef struct {

    BYTE InBuff[InBuffSize];
    int InBuffCount, InPtr;
    BYTE OutBuff[OutBuffSize];
    int OutBuffCount, OutPtr;

    HWND HWin;
    BOOL Ready;
    BOOL Open;
    unsigned int s; /* SOCKET */
    WORD RetryCount;
    BOOL CanSend, RRQ;

//    int SendCode;
//    int EchoCode;
       int ComPort;

    /* message flag */
    WORD NoMsg;
  }TComVar;
typedef TComVar far *PComVar;


/*
These are the fields that WOULD go in Teraterm's 'ts' structure, if
we could put them there.
*/
typedef struct _TS_SSH {
  BOOL Enabled;
  int CompressionLevel; /* 0 = NONE, else 1-9 */
  char DefaultUserName[256];

  /* this next option is a string of digits. Each digit represents a
     cipher. The first digit is the most preferred cipher, and so on.
     The digit SSH_CIPHER_NONE signifies that any ciphers after it are
     disabled. */
  char CipherOrder[16];

  char KnownHostsFiles[2048];
  int DefaultAuthMethod;
  char DefaultRhostsLocalUserName[256];
  char DefaultRhostsHostPrivateKeyFile[1024];
  char DefaultRSAPrivateKeyFile[1024];

  char DefaultForwarding[2048];
  BOOL TryDefaultAuth;

  int LogLevel;      /* 0 = NONE, 100 = Verbose */
  int WriteBufferSize;
  int LocalForwardingIdentityCheck;
```

```
} TS_SSH;

typedef struct _TInstVar {

  SOCKET                socket;
  SOCKADDR_IN           SockAddr;
  TComVar cv;
  struct                _TInstVar *lpNext;
  TCHAR HostName[256];
  unsigned short TCPPort;
  HANDLE HAsync;

  int fatal_error;
  int showing_err;
  char FAR * err_msg;

  HWND NotificationWindow;
  unsigned int notification_msg;
  long notification_events;

  BOOL hostdlg_activated;
  BOOL hostdlg_Enabled;

  int protocol_major;
  int protocol_minor;

  PKTState pkt_state;
  SSHState ssh_state;
  AUTHState auth_state;
  CRYPTState crypt_state;
  HOSTSState hosts_state;
  FWDState fwd_state;

/* The settings applied to the current session. The user may change
   the settings but usually we don't want that to affect the session
   in progress (race conditions). So user setup changes usually
   modify the 'settings' field below. */
  TS_SSH session_settings;

/* our copy of the global settings. This is synced up with the shared
   memory only when we do a ReadIniFile or WriteIniFile
   (i.e. the user loads or saves setup) */
  TS_SSH settings;
} TInstVar;

#define LOG_LEVEL_FATAL      5
#define LOG_LEVEL_ERROR      10
#define LOG_LEVEL_URGENT     20
#define LOG_LEVEL_WARNING    30
#define LOG_LEVEL_VERBOSE    100

#define SSHv1(pvar) ((pvar)->protocol_major == 1)
#define SSHv2(pvar) ((pvar)->protocol_major == 2)

void notify_established_secure_connection(PTInstVar pvar);
void notify_closed_connection(PTInstVar pvar);
void notify_nonfatal_error(PTInstVar pvar, char FAR * msg);
void notify_fatal_error(PTInstVar pvar, char FAR * msg);
void notify_verbose_message(PTInstVar pvar, char FAR * msg, int level);

void get_teraterm_dir_relative_name(char FAR * buf, int bufsize, char FAR * basename)
;
int copy_teraterm_dir_relative_path(char FAR * dest, int destsize, char FAR * basenam
e);

#ifdef __cplusplus
```



```
extern "C" {
#endif

void init_TTSSH(PTInstVar pvar);
int PASCAL FAR TTXconnect( PTInstVar pvar, SOCKET s, const struct sockaddr FAR *name,
 int namelen);
int PASCAL FAR TTXWSAAsyncSelect( PTInstVar pvar, SOCKET s, HWND hWnd, u_int wMsg,lon
g lEvent);
int PASCAL FAR TTXrecv(PTInstVar pvar, SOCKET s, char FAR * buf, int len, int flags);
int PASCAL FAR TTXsend(PTInstVar pvar, SOCKET s, char const FAR * buf, int len, int f
lags);
void PASCAL FAR TTXOpenTCP(PTInstVar pvar);
void PASCAL FAR TTXCloseTCP(PTInstVar pvar);
int PASCAL FAR TTXProcessCommand(PTInstVar pvar, HWND hWin, WORD cmd);

void OnSSHRead(PTInstVar pvar, int localch, unsigned char *data, unsigned int length)
;

#ifdef __cplusplus
}
#endif

void CommOpen(HWND HW, PTInstVar pvar);
void CommStart(PTInstVar pvar, LONG lParam);
BOOL CommCanClose(PTInstVar pvar);
void CommClose(PTInstVar pvar);
void CommProcRRQ(PTInstVar pvar);
void CommReceive(PTInstVar pvar);
void CommSend(PTInstVar pvar);

#define WM_USER_COMMNOTIFY WM_USER+4
#define WM_USER_COMMOPEN WM_USER+5
#define WM_USER_COMMSTART WM_USER+6
#define WM_USER_DLGHELP2 WM_USER+7
#define WM_USER_GETHOST WM_USER+8
#define WM_USER_NOTIFY_CONNRESULT WM_USER+100
#define WM_USER_NOTIFY_CHOPENRESULT WM_USER+101

#endif
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
This code is copyright (C) 1998-1999 Robert O'Callahan.
See LICENSE.TXT for the license.
*/

#include "ttxssh.h"

void UTIL_init_sock_write_buf(UTILSockWriteBuf FAR * buf) {
  buf_create(&buf->bufdata, &buf->buflen);
  buf->datastart = 0;
  buf->datalen = 0;
}

static int send_until_block(PTInstVar pvar, SOCKET s, const char FAR * data, int len)
 {
  int total_sent = 0;

  while (len > 0) {
    int sent_amount = send(s, data, len, 0);

    if (sent_amount < 0) {
      if (WSAGetLastError() == WSAEWOULDBLOCK) {
        return total_sent;
      } else {
        return sent_amount;
      }
    } else {
      total_sent += sent_amount;
      data += sent_amount;
      len -= sent_amount;
    }
  }
```



```
  return total_sent;
}

BOOL UTIL_sock_buffered_write(PTInstVar pvar, UTILSockWriteBuf FAR * buf,
  UTILBlockingWriteCallback blocking_write, SOCKET socket, const char FAR * data, int
 len) {
  int curlen;
  int desiredlen;
  int space_required;
  int amount_to_write_from_buffer;
  BOOL did_block = FALSE;
  int first_copy_start;
  int first_copy_amount;

  /* Fast path case: buffer is empty, try nonblocking write */
  if (buf->datalen == 0) {
    int sent_amount = send_until_block(pvar, socket, data, len);

    if (sent_amount < 0) {
      return FALSE;
    }
    data += sent_amount;
    len -= sent_amount;
  }

  if (len == 0) {
    return TRUE;
  }

  /* We blocked or the buffer has data in it. We need to put this data
     into the buffer.
     First, expand buffer as much as possible and necessary. */
  curlen = buf->buflen;
  desiredlen = min(pvar->session_settings.WriteBufferSize, buf->datalen + len);

  if (curlen < desiredlen) {
    int wrap_amount = buf->datastart + buf->datalen - curlen;
    int newlen = min(pvar->session_settings.WriteBufferSize, 2*desiredlen);

    buf->bufdata = realloc(buf->bufdata, newlen);
    buf->buflen = newlen;

    if (wrap_amount > 0) {
      int wrap_to_copy = min(wrap_amount, newlen - curlen);

      memmove(buf->bufdata + curlen, buf->bufdata, wrap_to_copy);
      memmove(buf->bufdata, buf->bufdata + wrap_to_copy, wrap_amount - wrap_to_copy);
    }
  }

  /* 1) Write data from buffer
     2) Write data from user
     3) Copy remaining user data into buffer
  */
  space_required = max(0, buf->datalen + len - buf->buflen);
  amount_to_write_from_buffer = min(buf->datalen, space_required);

  if (amount_to_write_from_buffer > 0) {
    int first_part = min(amount_to_write_from_buffer, buf->buflen - buf->datastart);

    did_block = TRUE;
    if (!blocking_write(pvar, socket, buf->bufdata + buf->datastart, first_part)) {
      return FALSE;
    }
    if (first_part < amount_to_write_from_buffer) {
      if (!blocking_write(pvar, socket, buf->bufdata, amount_to_write_from_buffer - f
```

```
irst_part)) {
        return FALSE;
      }
    }

    buf->datalen -= amount_to_write_from_buffer;
    if (buf->datalen == 0) {
      buf->datastart = 0;
    } else {
      buf->datastart = (buf->datastart + amount_to_write_from_buffer)%buf->buflen;
    }
    space_required -= amount_to_write_from_buffer;
  }

  if (space_required > 0) {
    did_block = TRUE;
    if (!blocking_write(pvar, socket, data, space_required)) {
      return FALSE;
    }
    data += space_required;
    len -= space_required;
  }

  first_copy_start = (buf->datastart + buf->datalen)%buf->buflen;
  first_copy_amount = min(len, buf->buflen - first_copy_start);
  memcpy(buf->bufdata + first_copy_start, data, first_copy_amount);
  if (first_copy_amount < len) {
    memcpy(buf->bufdata, data + first_copy_amount, len - first_copy_amount);
  }
  buf->datalen += len;

  if (did_block) {
    return UTIL_sock_write_more(pvar, buf, socket);
  } else {
    return TRUE;
  }
}

BOOL UTIL_sock_write_more(PTInstVar pvar, UTILSockWriteBuf FAR * buf, SOCKET socket)
{
  int first_amount = min(buf->buflen - buf->datastart, buf->datalen);
  int sent = send_until_block(pvar, socket, buf->bufdata + buf->datastart, first_amou
nt);

  if (sent < 0) {
    return FALSE;
  } else {
    if (sent == first_amount && first_amount < buf->datalen) {
      int sentmore = send_until_block(pvar, socket, buf->bufdata, buf->datalen - firs
t_amount);

      if (sentmore < 0) {
        return FALSE;
      }
      sent += sentmore;
    }

    buf->datalen -= sent;
    if (buf->datalen == 0) {
      buf->datastart = 0;
    } else {
      buf->datastart = (buf->datastart + sent)%buf->buflen;
    }
  }

  return TRUE;
```



```
}

void UTIL_destroy_sock_write_buf(UTILSockWriteBuf FAR * buf) {
  memset(buf->bufdata, 0, buf->buflen);
  buf_destroy(&buf->bufdata, &buf->buflen);
}

BOOL UTIL_is_sock_deeply_buffered(UTILSockWriteBuf FAR * buf) {
  return buf->buflen/2 < buf->datalen;
}
```

```
/*
Copyright (c) 1998-2001, Robert O'Callahan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list
of conditions and the following disclaimer in the documentation and/or other material
s
provided with the distribution.

The name of Robert O'Callahan may not be used to endorse or promote products derived
from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ''AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT
IES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT S
HALL
THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
This code is copyright (C) 1998-1999 Robert O'Callahan.
See LICENSE.TXT for the license.
*/

#ifndef __UTIL_H
#define __UTIL_H

typedef unsigned long uint32;
typedef unsigned short uint16;

#define NUM_ELEM(a) (sizeof(a) / sizeof((a)[0]))

#define buf_create(buf, len) (*(buf) = 0, *(len) = 0)

#define buf_ensure_size(buf, len, desired) \
  (*(len) < (desired) ? (*(buf) = realloc(*(buf), (desired)), *(len) = (desired)) : *
(len))

#define buf_ensure_size_growing(buf, len, desired) \
  (*(len) < (desired) ? (*(buf) = realloc(*(buf), (desired)*2), *(len) = (desired)*2)
 : *(len))

#define buf_destroy(buf, len) (free(*(buf)), *(buf) = 0, *(len) = 0)

#define get_uint32_MSBfirst(buf) \
  (((uint32)(unsigned char)(buf)[0] << 24) + ((uint32)(unsigned char)(buf)[1] << 16)
+ \
  ((uint32)(unsigned char)(buf)[2] << 8) + (uint32)(unsigned char)(buf)[3])

#define get_ushort16_MSBfirst(buf) \
  (((uint32)(unsigned char)(buf)[0] << 8) + (uint32)(unsigned char)(buf)[1])
```



```
#define set_ushort16_MSBfirst(buf, v) \
  ((buf)[0] = (unsigned char)((uint16)(v) >> 8),     \
   (buf)[1] = (unsigned char)(uint16)(v))

#define set_uint32_MSBfirst(buf, v) \
  ((buf)[0] = (unsigned char)((uint32)(v) >> 24),     \
   (buf)[1] = (unsigned char)((uint32)(v) >> 16),     \
   (buf)[2] = (unsigned char)((uint32)(v) >> 8),      \
   (buf)[3] = (unsigned char)(uint32)(v))

typedef struct _UTILSockWriteBuf {
  char FAR * bufdata;
  int buflen;
  int datastart;
  int datalen;
} UTILSockWriteBuf;

typedef BOOL (* UTILBlockingWriteCallback)(PTInstVar pvar,
  SOCKET socket, const char FAR * data, int len);

void UTIL_init_sock_write_buf(UTILSockWriteBuf FAR * buf);
BOOL UTIL_sock_buffered_write(PTInstVar pvar, UTILSockWriteBuf FAR * buf,
  UTILBlockingWriteCallback blocking_write, SOCKET socket, const char FAR * data, int
 len);
BOOL UTIL_sock_write_more(PTInstVar pvar, UTILSockWriteBuf FAR * buf, SOCKET socket);
void UTIL_destroy_sock_write_buf(UTILSockWriteBuf FAR * buf);
BOOL UTIL_is_sock_deeply_buffered(UTILSockWriteBuf FAR * buf);

#endif
```