Frequently Asked Questions

How can I get the Windows screen resolution from within Prolog?

There is no built-in way of accessing the Windows screen resolution directly from Prolog, but the following program uses winapi/5 to access a Windows API function to return these values:

```
screen_size( Xpix, Ypix ) :-
  winapi( user, (word,getsystemmetrics), [(word,0)], Xscreen, _ ),
  winapi( user, (word,getsystemmetrics), [(word,1)], Yscreen, _ ),
  Xpix = Xscreen,
  Ypix = Yscreen .
```

Why do I get errors when compiling C++ code based on your DLL example program?

Because C++ allows the overloading of functions, it performs what is known as "name mangling". This can cause "Function Not Defined" errors when trying to access the functions according to their given name. To avoid this you should include the following lines in the "C" code:

```
#ifdef _cplusplus

extern "C" {        /* Assume C declarations for C++ */

#endif    /* _cplusplus */

          Insert in here the functions that you want to access from outside
          Including BOOL FAR PASCAL functions

#ifdef _cplusplus

}         /* End of extern C { */

#endif /* _cplusplus */
```

Note that you should ensure that the PrologProcedure function pointer is declared with explicit arguments, as in:

```
FARPROC PrologProcedure(WORD,DWORD,WORD)
```

Can I run the DLL example without creating a stand-alone Prolog application?

You can set the DLL example for interactive use by adding the following programs to the INT386W.PL file.

```
'?ABORT? :-
  int386w_abort.

'?MESSAGE?'(Window, Message, Data, Goal) :-
  int386w_message(Window, Message, Data, Goal).
```

Then you should change the LoadProlog function in the INT386W.C file to contain the following WinExec command and recompile the DLL:

```
/* load WIN-PROLOG from C, and set it waiting for commands */

BOOL      FAR PASCAL LoadProlog()
{
          MSG       msg ;

          if        (PrologProcedure)
          {
                    return(FALSE) ;
          } ;
```

```
        if      (WinExec("pro386w  /v1 [int386w]. int386w_main",SW_SHOW) < 32)
        {
                return(FALSE) ;
        } ;

        while   (!PrologProcedure)
        {
                if      (PeekMessage(&msg,NULL,0,0,PM_REMOVE))
                {
                        TranslateMessage(&msg) ;
                        DispatchMessage(&msg) ;
                } ;
        } ;
        return(TRUE) ;
}
```

How do I avoid errors on my network when trying to save settings in PRO386W.INI?

To allow LPA Prolog for Windows to work correctly in this situation you should set the /N command line switch to specify the correct location for the initialisation files given your network setup. The /N command line switch takes the following values:

Switch	Location
/N0	Use Prolog directory (default)
/N1	Use Windows directory (usual setting for networks)
/N2	Use Current directory
/N3	Do not save .INI file

For example, you can set Prolog to use the Windows directory as the location for the initialisation file by entering the following at the DOS prompt:

```
C> WIN PRO386W /N1
```

You can also configure the initialisation file by setting the /N command line switch on the command line property of Prolog's program item.

Why do I get "Error 21" when using the not/1 predicate?

You should use the \+/1 predicate (negation as failure) instead of not/1. This is because not/1 is an attempt to implement truly "logical" negation where any variables in the call being negated that become bound during the process of negation cause error 21.

How do I get an edit control in a dialog to accept carriage returns?

Normally, dialogs process the <*enter*> key directly, taking it to mean "push the default button". If you want this key to act as a "carriage return" key, you should create your edit control with the es_wantreturn style. Even if you don't use this style, Windows provides the <*ctrl-J*> keystroke as an alternative way to insert a new line.

How can I handle the menus that I have installed on the menu bar?

Menu messages are now sent to the window that is in focus when the menu item is selected. To handle menu messages you need to attach a handler to that window that includes a case for the msg_menu message. For example the following code creates and adds a menu to the menu bar, then attaches a handler to the console window that handles the msg_menu message for both menu IDs and passes all other messages on to the default *window_handler/4*. After running the *make_menu/0* goal, selecting items from the *mymenu* menu will invoke the *my_window_handler/4* handler whenever the console window is in focus.

```
make_menu :-
  wmcreate(mymenu),
  wmnuadd(mymenu,-1,`Item &1`, 1000),
  wmnuadd(mymenu,-1,`Item &2`, 1001),
  wmnuadd(0,2,`&Mymenu`,mymenu),
  window_handler(1,my_window_handler).

my_window_handler(W,msg_menu,1000,R):-
  write('I am handling menu id 1000'),
  nl.

my_window_handler(W,msg_menu,1001,R):-
  write('I am handling menu id 1001'),
  nl.

my_window_handler(W,M,D,R) :-
  window_handler(W,M,D,R).
```

What should I do if I get the following message when starting up Prolog?

```
Error 42, Syntax Error
Internal Error 00005EF2: Exit From Prolog [Y/N]?
```

Are you running a memory resident virus shield?

If so you should make sure that you exlude the PRO386W.OVL file from the virus shield as this violates LPA's own checksum validation.

If not please contact LPA and let us know the exact details of your problem.

How do I use my 3D - look dialogs in a stand alone application?

The 3D-look dialogs are on as default in the development environment. This is not the case in stand-alone applications. To turn on the use of 3D-look dialogs in stand-alone applications you should include the following Prolog goal in your startup program:

```
ctl3d(1)
```

(this should be called prior to the creation of any dialogs).