Using Seehau as an OLE Automation Server

Object Linking and Embedding (OLE) Automation allows one application to drive another application. The driving application is known as an automation client or automation controller, and the application being driven is known as an automation server or automation component.

Seehau is an OLE Automation server. This means that Seehau based emulators can be manipulated from an application developed in any environment that supports OLE Automation. You can control Seehau from C++, Java, Delphi or Visual Basic.

To use Seehau in your application as an OLE Automation server, it must first be registered. To register, launch Seehau once, and then exit. The OLE commands are then sent to the Seehau version currently registered in the Windows Registry. After installing a new Seehau release, you must re-compile the application with a new Seehau type library.

Using Seehau Commands from Visual C++ v5.0 or Later

Make sure to select the Automation option for “What other support you would like to include?” while creating your project. Information about Seehau commands is contained in the type library “Seehau.tlb”. The type library can be found on Seehau distribution disks.

Once in the project, add a class from the type library Seehau.tlb (import type library Seehau.tlb). This creates two files Seehau.cpp and Seehau.h that contain an IDispatch wrapper class called IOleCmd for Seehau commands.

Each new Seehau release might contain a new version of Seehau.tlb. To use your application with a new type library, delete the files Seehau.cpp and Seehau.h, and then recreate them as previously described. Recompile the application.

To use commands in your code, include Seehau.h, declare an object variable of class IOleCmd and create IDispatch object for "Seehau.OleCmd" as in the following example.

```
#include "seehau.h"

//Declare object variable.
IOleCmd olecmd;

//Create IDispatch object.  It should launch Seehau.
BOOL bx = olecmd.CreateDispatch( "Seehau.OleCmd" );
if ((bx = = 0) | | (olecmd.m_lpDispatch = = NULL))      //check for an error
{
AfxMessageBox("Failed to create Dispatch object.  Is correct Seehau.exe
registered?");
}

//Now you are ready to call Seehau commands.
long i;
i = olecmd.File_GetVersion();   //return Seehau version.  (File_GetVersion is
a Seehau command)

//Exit Seehau
olecmd.Wnd_Exit();      //Wnd_Exit is a Seehau command which exits Seehau
```

Using Seehau Commands from Delphi

Add the following to the appropriate places in your program.

```
var
        v:variant;

procedure TForm1.Button1Click (Sender: TObject)
```

Using Seehau as an OLE Automation Server

```
begin
        //Create IDispatch object.  It should launch Seehau.
        v:=CreateOleObject('Seehau.OleCmd');
        //Now you are ready to call Seehau commands.
        v.File_LdCode('c:\myprogram');   // File_LdCode is a Seehau comand
                                         // which loads code
        //Exit Seehau
        v.Wnd_Exit();        //WndExit is a Seehau command which exits Seehau

end;
```

Using Seehau Commands from Visual Basic

Information about Seehau commands is contained in the type library Seehau.tlb. The type library is located in the Seehau directory and is automatically registered by Seehau Setup. You access the type library by setting references to it. To do this in VB v6.0, select Project, References. The References (project name) dialog box opens. Select "OleCmd1.0 Type Library" in the list of Available References.

The following example shows how to invoke Seehau commands from Visual Basic:

```
Create an object of type "Seehau.OleCmd"

Dim seehauApp As Seehau.OleCmd  'declare the object
Sett seehauApp = CreateObject("Seehau.OleCmd")  'create the object and start
Seehau

'Note the two lines above can be replaced by
Dim seehauApp As New Seehau.OleCmd      'create the object and start Seehau

MsgBox "Please wait while emulator is fully initialized, then ckick OK.",
VbOKOnly

'Call Seehau commands
'Start the emulator
seehauApp.Run_Go
MsgBox "Emulator must be running.", VbOKOnly
'Stop the emulator
seehauApp.Run_Break

MsgBox "Emulator stopped.", VbOKOnly
```

For more information refer to Visual Basic Help for CreateObject or GetObject.