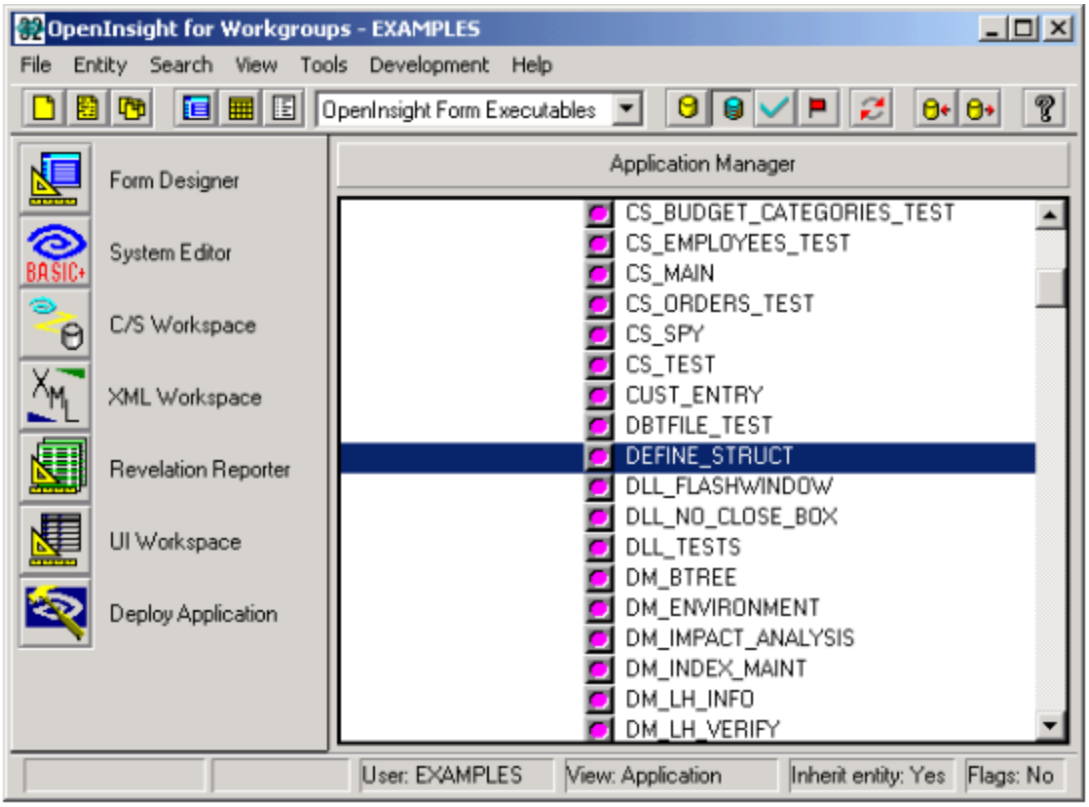
Displaying Memory Size

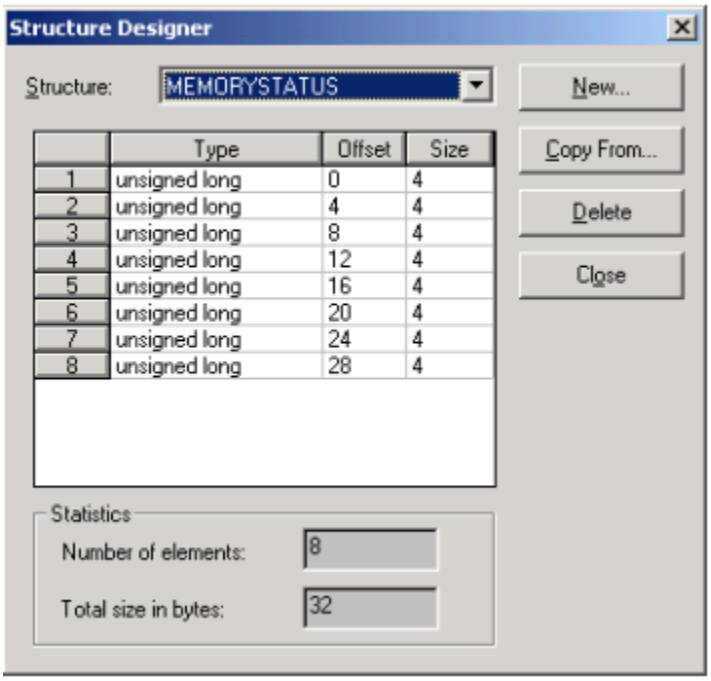
When deploying an application, it may be necessary to determine whether the computer has enough memory. The Windows API function GlobalMemoryStatus returns this information, in a structure called **MEMORYSTATUS**.

Defining the MEMORYSTATUS Structure

Before calling this function, it is necessary to create this structure. To accomplish this, OpenInsight contains a window, called **DEFINE_STRUCT**. To run this window, navigate to the OpenInsight Executables section of the Repository Outline and locate the **DEFINE_STRUCT** window, as shown below:



Then press **Shift-Double Click** to run the Structure Designer. Create a new structure called **MEMORYSTATUS**, and populate it with eight unsigned long variables (the structure GlobalMemoryStatus() populates), as shown below:



The name of the structure is not critical, but for documentation and consistency it should match the structure name expected by the function.

The Code

Below is the code for calling GlobalMemoryStatus. To test, paste this code in the CLICK event of a button.

```
declare function Get_Property, Blank_Struct
declare subroutine GlobalMemoryStatus
declare subroutine Parse_Struct

gMemoryStatus = Blank_Struct("MEMORYSTATUS")
GlobalMemoryStatus(gMemoryStatus)
Parse_Struct(gMemoryStatus, "MEMORYSTATUS", length, memoryload, totalphys, availphys, totalpagefile,
availpagefile, |
        total_virtual, avail_virtual)
call msg ( @window, ' Total physical memory ': totalphys / 1024 : 'KB, Available page file ': availpagefile /
1024: 'KB')
```

GlobalMemoryStatus can be declared as a subroutine because the return value is ignored.

Note how the variable **gMemoryStatus** is allocated memory (32 bytes) required by the **MEMORYSTATUS** structure, using the Blank_Struct() function. Then GlobalMemoryStatus is called, filling in the **gMemoryStatus** variable. Finally, the Parse_Struct subroutine populates the individual variables (**length, memoryload**, etc.) from the **gMemoryStatus** structure.

The Windows API Declaration

The code above will not run until the declaration for GlobalMemoryStatus has been added. To add the declaration, do the following:

1. Log out of the application.
2. Log into the **SYSPROG** application.
3. Add a row, (call it **DLL_APICALLS_KERNEL32**), with the first line as **KERNEL32** and containing the declarations as shown below. Note that the structure parameter is passed as **LPCHAR**, a pointer to a character string. Since the function does not return a value, its return parameter is **VOID** .

   ```
   KERNEL32
   VOID  STDCALL GlobalMemoryStatus(LPCHAR)
   //...other KERNEL32 functions, if necessary
   ```

4. Save the row.
5. Run Declare_FCNS at the System Editor Exec Line to create the declaration header, as shown below:

```
RUN DECLARE_FCNS 'DLL_APICALLS_KERNEL32'
```

6. Exit the editor.
7. Log out of **SYSPROG**.
8. Log into your application.
9. Run the window. A message, displaying the total physical memory and available page file size, displays.