

SysObject

Paul Manias

COLLABORATORS

	<i>TITLE :</i> SysObject		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Paul Manias	July 26, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	SysObject	1
1.1	Object: SysObject	1
1.2	Object: SysObject	1
1.3	Object: SysObject	2
1.4	Object: SysObject	3
1.5	Object: SysObject	3
1.6	Object: SysObject	4
1.7	Object: SysObject	4
1.8	Object: SysObject	5
1.9	Object: SysObject	6

Chapter 1

SysObject

1.1 Object: SysObject

OBJECT DOCUMENTATION

Name: SYSOBJECT AUTODOC
Version: 0.8 Beta.
Date: September 1997
Author: Paul Manias
Copyright: DreamWorld Productions, 1996-1997. All rights reserved.

CHANGES VERSION 0.9B

Added: SysObject->Flush
SysObject->Reset

Edited: SysObject->Free
SysObject->Get
SysObject->Init

1.2 Object: SysObject

OBJECT

Name: SysObject
Module: Kernel
Version: 1
Type: Complex
Notes: This structure concerns module programmers only.

DESCRIPTION

The SysObject forms the core of all object orientation in the DPKernel. Basically, a SysObject contains all of the details for a particular type of class. The system maintains a list of all current SysObjects, which you can find by calling FindSysObject(). You can add a new SysObject to the system through AddSysObject(), and you can remove them using Free() or RemSysObject().

ACTIONS

The SysObject supports the following actions:

```

Free() Free a SysObject.
Get() Get a new SysObject structure.
Init() Initialise a SysObject.

```

FUNCTIONS

Here are some functions supporting the SysObject:

```

AddSysObject() Add a new SysObject to the system.
FindSysObject() Search for a SysObject.
RemSysObject() Remove a SysObject from the system.

```

STRUCTURE

The SysObject structure consists of the following public fields:

```

ClassID ID for the class.
Name Pointer to a string containing the SysObject's name.
ObjectID ID for the object (eg ID_PICTURE, ID_HIDDEN).

```

The following function references, which are located inside the SysObject structure, are public only to system modules. Even then, you may only reference them during the InitTags() process. If you read or write to these fields, or attempt to make direct function calls with these addresses, your program/module will simply fail in future releases.

```

Activate Perform the native action for this object.
CheckFile Check for file recognition.
Clear Clear an object from its container.
Close Close the data filing section of this object.
CopyFromUnv Copy from universe to object.
CopyToUnv Copy from object to universe.
Deactivate End the native action for this object.
Display Display the object inside its container.
Draw Draw an object inside its container.
Flush Flush buffered data from an object.
Free Free object.
Get Get a new object structure (master only).
Hide Hide/Remove the object from the display.
Init Initialise object.
Load Load a file that belongs to this object.
Open Open this object as a virtual file.
Query Query the information held on this object.
Read Read some data from the object.
Reset Reset an object to a ready state.
Save Save this entire object as a file.
Write Write some data to the object.

```

1.3 Object: SysObject

FIELD

```

Name: ClassID
Type: WORD
Inheritance: ObjectID
On Change: Cannot change after initialisation.
Status: Read/IWrite

```

DESCRIPTION

This field specifies the Class or "group" that the SysObject belongs to. This is especially important for child classes, as they will inherit function and data support from the parent class when first initialised.

If you are initialising a hidden object, you will specify ID_HIDDEN in this field.

NOTE

It is illegal to specify an ID which is not provided in the system/register file.

SEE ALSO

Field: ObjectID

Includes: system/register.i

1.4 Object: SysObject

FIELD

Name: Name

Type: BYTE *

On Change: Cannot change after initialisation.

Status: Read/IWrite

DESCRIPTION

This field specifies the name of the system object. Names are fairly straight-forward, eg the Picture object has a name of "Picture". If you are writing a child class, then you must supply the name of the parent class first, followed by a second name describing your supporting class. For example: "Picture~Jpeg". The tilde symbol ~ must be used as the separator.

1.5 Object: SysObject

FIELD

Name: ObjectID

Type: WORD

On Change: Cannot change after initialisation.

Status: Read/IWrite

DESCRIPTION

This field specifies the same number as the ClassID if this is the master of the class, or ID_HIDDEN if it is a hidden object or child class.

NOTE

It is illegal to specify an ID which is not provided in the system/register file.

SEE ALSO

Field: ClassID

Includes: system/register.i

1.6 Object: SysObject

FIELD

Name: Free()
Type: Function
Synopsis: void Free(APTR Object [a0]);
Status: IWrite only.
Inheritance: None.

DESCRIPTION

This function is called whenever the programmer wants to Free() one of your objects from the system. If you are a master, you must undo any of your allocations and then free the structure itself. If you are a child class, you must only undo any allocations that you have made.

NOTE

If you are a child class, Free() will first call your free function, then it will call the free function of the master. This is important as it is the only way that the structure and any child structures can be effectively removed from the system.

To prevent allocations between the master and child routines getting confused, if you free any fields make sure that you drive them to NULL afterwards.

EXAMPLE

This example is derived from the Picture master class.

```
__asm __saves void PIC_Free(register __a0 struct Picture *pic)
{
    if (pic->prvAFlags & AF_DATA) {
        FreeMemBlock(pic->Bitmap->Data);
        pic->Bitmap->Data = NULL;
    }

    if (pic->prvAFlags & AF_PALETTE) {
        FreeMemBlock(pic->Palette);
        pic->Palette = NULL;
    }

    Free(pic->Bitmap);
    pic->Bitmap = NULL;

    FreeMemBlock(pic);
}
```

1.7 Object: SysObject

FIELD

Name: Get()
Type: Function
Synopsis: APTR Object = Get(void);
Status: IWrite only.
Inheritance: Parent class.

DESCRIPTION

This function is called whenever the programmer wants to Get() the latest version of your particular object. The routine that you place here should be fairly simple, allocating an UNTRACKED public memory block of the MEM_DATA type. You need to set up the ID and Version number, plus you may set any simple field settings.

EXAMPLE

This example is derived from the Picture object, which as you will notice also has a child Bitmap that is allocated here.

```
__asm __saves struct Picture * PIC_Get(void)
{
    struct Picture *Picture;

    if (Picture = AllocMemBlock(sizeof(struct Picture), MEM_DATA|MEM_UNTRACKED)) {
        Picture->Head.ID      = ID_PICTURE;
        Picture->Head.Version = PICVERSION;

        if (Picture->Bitmap = Get(ID_BITMAP|GET_NOTRACK)) {
            Picture->Bitmap->Parent = Picture;
            return(Picture);
        }
        else EMsg("Failed to get Bitmap.");
    }
    else EMsg("Failed to allocate picture memory.");

    return(NULL);
}
```

NOTE

This particular action applies to parent classes only, so if you are a child class, you will have to accept whatever the master decides to allocate.

If you fail to allocate your structure, return NULL to indicate failure.

1.8 Object: SysObject

FIELD

Name: Init()
 Type: Function
 Synopsis: LONG ErrCode = Init(APTR Object [a0], APTR Container [a1]);
 Status: IWrite only.
 Inheritance: Parent class.

DESCRIPTION

This function is called when the programmer calls the Init() action on an object. It's purpose is to prepare the object for the necessary handling for actions such as Draw() and Activate().

Rules for Initialisation

If the initialisation fails due to the fact that the object cannot handle the data (eg IFF trying to init JPEG) then the object's Init() function

must undo any changes it has made to the object and get it back to its original state. It must not Free() the actual object itself, and it must return an error code of ERR_NOSUPPORT. This will cause the Init() action to look for an object that can support the object better.

If the object was recognised but initialisation fails due to a system error or similar, then any error-code can be returned except for ERR_NOSUPPORT and ERR_OK. This will cause the Init() action to fail immediately and return to the program.

NOTE

You must never call Free() on your object while inside the Init() routine. If your call fails (eg you return ERR_DATA), then the Init() action will call your Free() function for you at a later time.

SEE ALSO

Kernel: Init()

1.9 Object: SysObject

FIELD

Name:

Type: Function

Synopsis:

Status: IWrite only.

Inheritance: Parent class.

DESCRIPTION
