# FC_DOSNotify

Olivier LAVIALE 2004

| COLLABORATORS | | | |
|---|---|---|---|
| | *TITLE* :<br><br>FC_DOSNotify | | |
| *ACTION* | *NAME* | *DATE* | *SIGNATURE* |
| WRITTEN BY | Olivier LAVIALE<br>2004 | January 13, 2023 | |

| REVISION HISTORY | | | |
|---|---|---|---|
| NUMBER | DATE | DESCRIPTION | NAME |
| | | | |

# Contents

# Chapter 1

# FC_DOSNotify

## 1.1 Feelin : FC_DOSNotify

FC_DOSNotify

IDs: Dynamic Super: NONE Include: <libraries/feelin.h>

This class is an anchor for DOS notifications. It uses its own server to handle notification events, so you don't need to worry about managing your own message ports.

Using this class, you will be notified when the file or directory changes. For files, you will be notified after the file is closed. Not all filesystem will support this, in particuliar, most network filesystems won't support it.

ATTRIBUTES

FA_DOSNotify_Name FA_DOSNotify_Hook

FA_DOSNotify_HookEntry FA_DOSNotify_Method

FA_DOSNotify_Object

## 1.2 FC_DOSNotify / FA_DOSNotify_Hook

NAME

FA_DOSNotify_Hook -- (01.00) -- [I..], struct Hook *

FUNCTION

Use this attribute if you want to be notified using a call-back hook. If a notification happens, your hook is called with a struct FS_DOSNotify_Notify as message.

The struct Hook must be valid until you dispose the object.

EXAMPLE

F_HOOKM(void,Hook_Notify,FS_DOSNotify_Notify) { struct FeelinClass *Class = Hook -> h_Data; struct LocalObjectData *LOD = F_LOD(Class,Obj);

F_Log(0,"NOTIFY - Class 0x%08lx - File '%s'",Class,Msg -> FileName); }

F_METHOD(void,mNew) { ...

LOD -> Notify.h_Entry = (HOOKFUNC) Hook_Notify; LOD -> Notify.h_Data = Class;

Obj = F_NewObj(FC_DOSNotify, "FA_DOSNotify_Name", "ENV:MyApp.prefs", "FA_DOSNotify_Hook", Hook_Notify, "FA_DOSN
Obj, TAG_DONE);

... }

NOTE

You don't need to define FA_DOSNotify_Object with this attribute. It's often the better solution but you can manage your hook as you want.

FA_DOSNotify_Hook, FA_DOSNotify_HookEntry and FA_DOSNotify_Method override each other e.i the last defined is used as call-back. You should define only one of them.

SEE ALSO

FA_DOSNotify_Name

## 1.3 FC_DOSNotify / FA_DOSNotify_HookEntry

NAME

FA_DOSNotify_HookEntry -- (01.00) -- [I..], HOOKFUNC

FUNCTION

Use this attribute if you want to be notified using a call-back hook entry. If a notification happens, your hook entry is called with a struct FS_DOSNotify_Notify as message.

Using hook entries is useful if you don't need more information than the FC_DOSNotify object and the filename beeing watched. Hook entries don't require a struct Hook to be setuped.

EXAMPLE

F_HOOKM(void,Update_ColorPrecision,FS_DOSNotify_Notify) { ULONG precision = MAKE_ID('e',0,0,0);

GetVar(Msg -> FileName,(STRPTR)(&precision),4,NULL);

CUD.Precision = precision >> 24;

F_Log(0,"Color Precision '%lc' (%ld)",precision,CUD.Precision); }

F_METHOD(void,mNew) { ...

Obj = DOSNotifyObject, "FA_DOSNotify_Name", "ENV:Feelin/VAR_MYCLASS_COLORPRECISION", "FA_DOSNotify_HookEnt
Update_ColorPrecision, End;

... }

NOTE

FA_DOSNotify_Hook , FA_DOSNotify_HookEntry and FA_DOSNotify_Method override each other e.i the last defined is used as call-back. You should define only one of them.

SEE ALSO

FA_DOSNotify_Name

## 1.4 FC_DOSNotify / FA_DOSNotify_Method

NAME

FA_DOSNotify_Method -- (01.00) -- [I..], ULONG

FUNCTION

Use this attribute if you want to be notified using a method. If a notification happens, the object defined by the FA_DOSNotify_Object attribute is invoked with the method and a struct FS_DOSNotify_Notify as message.

This attribute is usefull because you don't need to setup a struct Hook. The FA_DOSNotify_Object attribute must be defined though, otherwise the object is not created (it will be useless because without an object, you will never heard about notifications).

EXAMPLE

F_METHODM(void,mNotify,FS_DOSNotify_Notify) { struct LocalObjectData *LOD = F_LOD(Class,Obj);

F_DebugOut("NOTIFY - Class 0x%08lx - File '%s'\n",Class,Msg -> FileName); }

F_METHOD(void,mNew) { ...

Obj = DOSNotifyObject, "FA_DOSNotify_Name", "ENV:MyApp.prefs", "FA_DOSNotify_Method", F_IDM(FM_MyApp_Update), "FA_DOSNotify_Object", Obj, End; ... }

NOTE

FA_DOSNotify_Hook , FA_DOSNotify_HookEntry and FA_DOSNotify_Method override each other e.i the last defined is used as call-back. You should define only one of them.

SEE ALSO

FA_DOSNotify_Name

## 1.5 FC_DOSNotify / FA_DOSNotify_Name

NAME

FA_DOSNotify_Name -- (01.00) -- [I..], STRPTR

FUNCTION

To create a FC_DOSNotify object you must define the FA_DOSNotify_Name attribute. This attribute is the name of the file or directory that should be watched.

Then, depending on the notification mecanism you choose (call-back hook or method), you must either define FA_DOSNotify_Hook or FA_DOSNotify_Method and FA_DOSNotify_Object .

EXAMPLE

Obj = F_NewObj(FC_DOSNotify, "FA_DOSNotify_Name", "ENV:MyApp.prefs", "FA_DOSNotify_Hook", Hook_Notify, TAG_DONE

or

Obj = F_NewObj(FC_DOSNotify, "FA_DOSNotify_Name", "ENV:MyApp.prefs", "FA_DOSNotify_Method", F_IDM(FM_MyApp_U "FA_DOSNotify_Object", Obj, TAG_DONE);

## 1.6 FC_DOSNotify / FA_DOSNotify_Object

NAME

FA_DOSNotify_Object -- (01.00) -- [I..], FObject

FUNCTION

Use this attribute to define the object that should heard about notification.

If you want to be notified with a method (defined by FA_DOSNotify_Method ), you must supply this attribute. FC_DOSNotify object is not created otherwise. The method defined by FA_DOSNotify_Method is invoked on the object defined by FA_DOSNotify_Obje when a file or directory changes.

This attribute is not required if you use a call-back hook (defined by FA_DOSNotify_Hook ), but it may help.

SEE ALSO

FA_DOSNotify_Name