DynamicStructure

Version 1.1.0 – September 2002

Osmose Éditeur
33, avenue Jean Monnet
F-13410 Lambesc
Tel.: +33 (0)4 42 92 83 55
Fax.: +33 (0)4 42 92 83 27
http://www.osmose.net

Version 1.1.0
Bug fixes and new features

Bugs Fixes

- **ds_GetObjectComment** did not always return a correct text if the comment is stylised.
- **ds_GetStyleSheet** returned sometimes some « garbage » when the font chosen was « System Font » or « Application Font ».
- **ds_GetObjectMethodIDs** did not return informations about variables or fields included in a group, and did not make the difference between process and interprocess variables.

New features

- **ds_GetObjectComments** and ds_SetObjectComments accept one more parameter: a BLOB for the style of the comment, if there is one. This style can be used with 4D Write, via the clipboard.
- **ds_GetObjectMethodIDs** accept one more optionnal parameter: an array (synchronized with previous ones) holding the number of the page where the variable/field is.
- **ds_SetStyleSheet**: it is possible to set the stylesheet at « System Font », « Application Font » or « Little Font System » using special chars under 6.7.
- **ds_GetStyleSheet** and **ds_SetStyleSheet** accept more parameters on Mac Os X and 4D 6.8 to use new steelsheets.

New routines

ds_GetPluginNames return the list of every installed plugins, even if they are on Mac4DX or Win4DX.
ds_GetPluginRoutines return the list of a plugin routine’s.
ds_GetFormObjectMethodIDs do the same thing as ds_GetObjectMethodIDs, limited to only one form.
ds_Preferences lets the devloper changes the behavior of dynamic structure when launching a lot of method names on slow computers.

Misc.

Constants for error codes have been added . They all start with « kdsERR_ »

DynamicStructure

DynamicStructure is a plugin thats lets the 4D developer get and set dynamically the properties and contents of objects in the structure file.It is a powerfull tool,very usefull to developpers: from automatic documentation to dynamic manipulation of objects, DynamicStructures covers a lot of features that can’t be accessed with the 4D language.

Comptability

DynamicStructure is compatible with the following versions of 4D:

- **4D 6.7** and the supported OS.
  Use at least 4D 6.7.4
- **4D 6.8** and the supported OS (including Mac OS X and Wndows XP).
- **Version 6.5**: The plugin is not supposed to be used under 4D 65, and Osmose Éditeur can not offer tech support when the plugin is used with this version of 4D. We can say anyway that the beta-test has shown that most of the routines are OK with 4D 6.5, and we hope to be able to support version 6.5 in the future. If you use the plugin under 4D 65, please, let us know which routines work, and which don't: we'll fix any problem - if possible - in order to support 4D 65. Use at least 4D 6.5.9.r2.

Installation

Drop the plugin in the appropriate Mac4DX or Win4DX folder.

4D 6.7 on Mac OS as on Windows: Use DynamicStructure.4DX (and DynamicStructure.RSR on Windows)

4D 6.8

- On Windows, use DynamicStructure.4DX and DynamicStructure.RSR
- On Mac (OS 9.2 and OS X), use the carbonised version: DynamicStructure.4CX

Register the plugin - Demo mode

Once installed, the plugin runs in demo mode until you enter a valid license number, using the ds_Register routine:

ds_Register(selector;licence) -> errorCode

selector	Integer	Kind of license
Licence	String	License number

Possible values for selector

1: Developer license for Mac OS
2: Developer license for Windows
3: Compiled Runtime Deployment
4: Unlimited For One Application Compiled Deployment

If the license is valid, the routine returns 0 (no error). Else, it returns error code -30000 and still runs in demo mode.

There are 3 kinds of license:

- The *developer license*, which allows the development (for one platform) in interpreted mode, and let the developer do some testing in compiled mode. One license per machine is needed. You can use the same license for every structure file you develop on this machine.
- The *compiled runtime license*, which runs only in compiled mode, and does nothing in interpreted mode. This license is the one you must use if you use DynamicStructure at runtime, compiled. This license goes for one machine too.
- At last, the *unlimited runtime license* for one app, which goes for only one application and one platform. You need this one if you want to distribute several copies of one compiled application.

Under client/server, you must have one licence per machine that uses the plug-in.
Mac OS = one platform, whatever the OS is: OS 9 or OS X.

Examples:

- A developer uses DynamicStructure only for optimising his/her developments, doing better technical documentation, … DynamicStructure is not used in his/her developments once installed at his/her customers's offices.
  -> He/She must buy one *developer license*.
- This developer uses the plugin in some compiled product:

-> He/She must buy one *developer license* and as much as *runtime compiled* licenses than machines that will use the plugin.

- A developer uses the plug-in one his Mac and on his PC: He/She must buy 2 *developer licenses* (one per plateform).
- DynamicStructure is used with 4D standalone Web Server, in a compiled structure: only one *runtime compiled license is needed*.

For any question, please contact infos@osmose.net, check our web site http://www.osmose.net, or ask your distributor.

Limitations of the demo mode are the following:

- Only 20 "Set" and 50 "Get" routines can be used. After that, the plugin does nothing but returning the "not registered" error code.
- If no error occured when calling a plugin's routine, error -30 000 ("not registered") is returned instead of 0.

Example of registration method:

```
C_INTEGER($err)
C_STRING(255;$license)
 `
$macLicense:="" ` <- Put your developer license number for Mac OS
$winLicense:="" ` <- Put your developer license number for Window
$runLicense:="" ` <- Put your runtime license number
If (Not(Compiled Application)) `This is a developer license
 If (wßB_IsMac )
 $err:=ds_Register (1$macLicense) ` selector 1 = Mac developer license
 Else
 $err:=ds_Register (2$winLicense) ` selector 2 = Windows developer license
 End if
Else "Compiled Runtime", license
 ` If the license starts with "UNL", it is an Unlimited license for one application". Else, it is
 ` a usual Compiled Runtime Deployment license.
 If ($license="UNL@")
 $err:=ds_Register (4 ;$runLicense)
 Else
 $err:=ds_Register (3; $runLicense)
 End if
 `
 ` NOTICE: the developer license lets the developer test in compiled mode:
 ` If (wßB_IsMac )
 ` $err:=ds_Register (1;"DEV....")
 ` Else
 ` $err:=ds_Register (2;"DEV....")
 ` End if
 `
End if
If ($err=-30000) ` -30 000 = invalid license
 . . .
End if
```

How to use DynamicStructure - important things

Routines that only return informations (the « get » routines) *never* modify the structure file. So, they can be used freely, except for special routines: for instance, it is not possible to read the database properties from a 4DClient or to get the text of a method in a compiled database.

Very Important

If you want to read only one thing, then read *this*

- Every routine that *sets* an information writes *immediatly* the structure file. Modifications are not restricted to the current process, they are « definitive » until the next change, just like if the object has been modified in Design mode.

- Because of that, it is *very important* that the 4D developper takes care himself of the consequences of the modifications he make, and takes care himself of multiple access to an object. It is very important in Client/Server environment and/or if important objets are modified, such as Trigger or a Database method… Because changes are immediatly applied to all the database. Using a semaphore is, in those situations, mandatory.

- At last, while developping, you must work on copies if you develop anything that - using the plugin - may change the current structure file (ds_TextToMethod, ds_DuplicateForm, …)

Under Client/Server, modified objects are locked during the modifications. If they were used by another Client, the plugin returns the error: 30200 (Locked Objet). There is no object locking under 4D standalone.

Known plugin problems: 4D interpretor

In interpreted mode, some variables are retyped after a call to a plugin. In their routines definitions, plugins tell 4D what kind of parameters they expect, and 4D changes the type of parameters according to that. This concerns every plugin, not only DynamicStructure.

For instance: the routine RoutineOfThePlugin accepts a Long Integer as parameter #1:
C_REAL($aReal)
RoutineOfThePlugin($aReal)
=> After the call, $aReal has became a Long Integer.
You may loose information, since a long integer is 4 bytes long, and a real is 8 bytes long. In fact, if after the call you write in $aReal a number greater than MAXLONG, 4D will « troncate » the value to fit in a Long Integer (4 bytes).

It could be really problematic in case of array: imagine a routine that expects an integer as parameter. You want to use the current selected element of an array as the parameter, so you write something like:
ARRAY TEXT(theArray ;10)
. . . later in the code . . .
RoutineOfThePlugin(theArray)
=> after the call, theArray became a Long Integer !
The array theArray has became a Long Integer variable: we have lost the acces to its previous content.
In such situation, it could be very usefull to use a temporary buffer:
ARRAY TEXT(theArray ;10)
. . . later in the code . . .
$buffer:= theArray
RoutineOfThePlugin($buffer)

Please, note that when a plugin needs an array, and not a fixed-size variable, there is no typecasting: an Integer array doesn’t become a Text array or a something else.

Also, note that in compiled mode, there is no problems.

Known limitations of particular routines

(see the documentation of the routines)

4D must be reloaded after using these routines:

- ds_NewMethod
- ds_SetMethodName
- ds_DuplicateForm

Not allowed in compiled environment:

- ds_NewMethod
- ds_SetMethodName
- ds_MethodToText
- ds_TextToMethod
- ds_DuplicateForm
- ds_GetObjectComments
- ds_SetObjectComments

Not allowed on 4D Client:

- ds_GetDatabaseProperties
- ds_SetDatabaseProperties
- ds_NewMethod
- ds_SetMethodName
- ds_DuplicateForm

Not allowed on 4D Server:

- ds_NewMethod
- ds_SetMethodName
- ds_DuplicateForm

About every routine of the plug-in

All routines of the plugin are functions that return an error code: « errorCode », of type Long Integer. A 0 value means that no problem occured. Any other value is a system error, a 4D error, or a plugin error (see: Error Codes)

When the expected parameter is a Numeric Array you can use as you want an Integer Array, a Long Integer Array, a Real Array.

When the expected parameter is a Text Array you can use as you want a Text Array, or a string Array. If you choose a too small string lenght for the string Array, strings will be troncated to fill the space.

When the expected parameter is a Numeric you can use as you want an Integer, a Long Integer, or a Real. See the section « The 4D interpretor ».

The plugin cannot get or set a component’s object information, it returns an error instead.

DynamicStructure: Methods

With DynamicStructure, you can manipulate any kind of methods: project, object, form, database, triggers. You usually get the ID of the method to manipulate using one routine of the plugin (ds_GetMethodNames, ds_GetFormMethod, …), then you manipulate it: get/set the code, get/set the comment, or open the method in design mode.

ds_GetMethodNames

ds_GetMethodNames (names;filter{;visible{;IDs}}) ← errorCode

Names	Text Array	←	Method names array
Filter	String	→	Name search filter
Visible	Numeric Array or Boolean	←	Visibility
Ids	Numeric Array	←	Methods IDs
errorCode	Long Integer	←	Error Code (0 = no error)

Load synchronised Arrays:

- All project methods names
- Their visibility
- Their IDs

The filter parameter allows to load only methods which starts by the filter value.

Arrays Visible and Ids, and Filter are optionals, they may be omit when calling the routine. The visibility of a method is set in Design Mode, when the method editor is opened: use the "Get method properties…" of the "Method" menu. A method that is invisible is not listed in 4D Editors ("Execute method", Quick Report, …).

Beware: compilation changes the Ids.

Version 1.1: Names can be loaded less or more quickly, given less or more time to the 4D scheduler, using the ***ds_Preferences*** routine. Because loading a lot of methods on slow computers can take several seconds, and may not give the hand to the 4D scheduler. In this situation, the user may think his/her machine is freezed while it is not.

ds_MethodToText

ds_MethodToText (methodID;code) ← errorCode

methodID	Integer	→	Method ID

code	Text	←	The Code
errorCode	Long Integer	←	Error Code (0 = no error)

Returns the Text of the method which ID is passed.

ds_TextToMethod

ds_TextToMethod (newCode;methodID) ← errorCode

newCode	Text	→	Method code
methodID	Integer	→	Method ID
errorCode	Long Integer	←	Error Code (0 = no error)

Modify the code of a method which ID is passed.

NOTE: if the method is currently opened in Design Mode, its content is not redrawn in its window. If you close the window, it is the actual text in the window that becomes the code. To update the window, choose « Revert to saved » in the File menu.

ds_OpenMethod

ds_OpenMethod (methodName{ ;lineNum) ← errorCode

methodName	String	→	Name of the project method to open
lineNum	Integer	→	N° of the line to underline
errorCode	Long Integer	←	Error Code (0 = no error)

Open the method in Design Mode. If the line number is valid, the line number lineNum is selected. If the current user doesn’t have access to the Design mode or if the method is protected (component) an error is returned. You can only open project methods with this routine. To open an other kind of method, use ds_OpenMethodByID.

ds_OpenMethodByID

ds_OpenMethodByID (methodID{ ;lineNum) ← errorCode

methodID	Integer	→	ID of the method
lineNum	Entier	→	N° of the line to underline
errorCode	Long Integer	←	Error Code (0 = no error)

Open the method in Design Mode. If the line number is valid, the line number lineNum is selected. If the current user doesn’t have access to the structure mode or if the method is protected (component) an error is returned.
This routines enables you to open method that have "no name", such as Object methods, form methods, … Those Ids ca be get using other routines of the plugin such as ds_GetFormMethod.

ds_NewMethod

ds_NewMethod (newMethodName{ ;ID}) ← errorCode

newMethodName	String	→	Name of the new method
ID	Integer	→ ←	ID of the method
errorCode	Long Integer	←	Error Code (0 = no error)

Creates a new method named newMethodName.

ID is the ID of the newly created method. Pass 0 or –1 to force 4D to calculate a unique ID for you. If you pass an other value, 4D will use this value as ID for the method. If the ID is already used by an other method, an error is returned.

IMPORTANT NOTICE: Actual versions of 4D (67/68) don't enable a plugin to inform 4D that a new method has been created: It is important to reload 4D after having created a method this way. What can be done is to create several methods, then quit. Don't change the code of a method if the new code includes names of newly created method, the tokenization will not recognize the new methods. The following code does this: it creates 50 methods named « aMethod_01 » to « aMethod_50 », each method will contain a comment saying that it was created using DynamicStructure:

```
` Create_50_Methods
C_LONGINT($i;$err;$id)
C_TEXT($prefix;$code;$name)
$i:=0
$prefix:="aMethod_"
$code:="` Created using ds_NewMethod, "+String(Current date)
$code:=$code+" - "+String(Current time;HH MM )+"`"+("-"*50)
For ($i;1;50)
 $name:=$prefix+String($i;"00")
 $id:=-1
 $err:=ds_NewMethod ($name;$id)
 If ($err#0)
 ALERT("New method #"+String($i)+" => error #"+String($err))
 $i:=50
 Else
 $err:=ds_TextToMethod ("` "+$name+$code;$id)
 If ($err#0)
```

```
 ALERT("ds_TextToMethod#"+String($i)+" => error #"+String($err))
 $i:=50
 End if
 End if
End for
`
QUIT 4D
`
`
```

ds_SetMethodName

ds_SetMethodName (oldName ;newName) ← errorCode

oldName	String	→	Method that we want to change the name
newName	String	→	New method name
errorCode	Long Integer	←	Error Code (0 = no error)

Modify the name of the method oldName. If oldName is not a valid projet method, or if the newName is an existant method or if newName is an invalid name (empty or more than 31 caracters), the routine does nothing an returns an error.

IMPORTANT NOTICE: Actual versions of 4D (67/68) don't enable a plugin to inform 4D that a method name has been modified: It is important to reload 4D after having changed a method name this way.

ds_SetMethodVisible

ds_SetMethodVisible (methodName;isVisible) ← errorCode

methodName	String	→	Name of the method to modify
isVisible	Numérique	→	New visibility (1 = visible, 0 = invisible)
errorCode	Long Integer	←	Error Code (0 = no error)

Modify the visibility of the method methodName. An invisible method doesn't appear in 4D dialogs: « Execute a method », in User mode, formula editor, …

ds_GetObjectMethodIDs

ds_GetObjectMethodIDs((IDs ;varNames ;formIDs{ ;pageNums}) ← errorCode

Ids	Numeric Array	←	IDs of object methods
varNames	String/Text Array	←	Names of variables

formIDs	Numeric Array	←	Ids of forms
pageNums	Numeric Array	←	Num. of the page containing the object
errorCode	Long Integer	←	Error Code (0 = no error)

This routine loads in synchronised arrays informations about *all* objects methods of the structure file:

- Their IDs (usable with ds_MethodToText fo instance)
- The name of the variable/Field that holds the method
  For fields, the name
- IDs of forms which contain the variables/fields
- The number of the page that contains the object

NOTICE: fields names are formated using this naming convention: number of the table on 3 chars + number of the field on 3 chars. When the table is the table of the form, its number is 0 (it is formated "000")
Examples:

Field #12 of the table of the form: "000012"
Field #8 of table #36 in a form of an other table: "036008"

It is up to the developer to extract the numbers and build the full name of the field.

To load the objects of one single form, use the ***ds_getFormObjectMethodIDs*** routine.

Version 1.1: Names can be loaded less or more quickly, given less or more time to the 4D scheduler, using the ***ds_Preferences*** routine. Because loading a lot of methods on slow computers can take several seconds, and may not give the hand to the 4D scheduler. In this situation, the user may think its machine is freezed while it is not.

ds_TokenizeStatement

ds_TokenizeStatement (toTokenize;tokens) ← errorCode

toTokenize	Text	→	Text
Tokens	Text	←	Tokens
errorCode	Long Integer	←	Error Code (0 = no error)

Tokenize the Text toTokenize. Only one line can be tokenised.

ds_ByteSwapTokens

ds_ByteSwapTokens (tokens;swapped) ← errorCode

Tokens	Text	→	OrignalsTokens
Swapped	Text	←	Tokens Byte swapped
errorCode	Long Integer	←	Error Code (0 = no error)

Transform a Mac token to a PC token, or a PC token to a Mac token. It makes it possible for you to transport tokens from one plateform to the other.

ds_DetokenizeStatement

ds_DetokenizeStatement (tokens;text) ← errorCode

Tokens	Text	→	Tokens
Text	Text	←	Text
errorCode	Long Integer	←	Error Code (0 = no error)

Transform a token to a readable Text.

ds_ExecuteToken

ds_ExecuteToken (tokens) ← errorCode

Tokens	Text	→	Tokens
errorCode	Long Integer	←	Error Code (0 = no error)

Execut the tokenised code. To execute a token is faster than calling «EXECUTE ». if you have to execute several times the same instruction, it will be faster to tokenise the expression and then execute the tokens.

DynamicStructure: Forms

ds_GetFormNames

ds_GetFormNames (names;tableNum{;IDs{;formKinds{;inputName{;outputName}}}}) ← errorCode

Names	Text Array	←	Forms name
tableNum	Integer	→	Table number
IDs	Numeric Array	←	Ids of the forms
formKinds	Numeric Array	←	Kinds of the forms
inputName	String/Text	←	Name of the Input form
oututName	String/Text	←	Name of the Output form
errorCode	Long Integer	←	Error Code (0 = no error)

Load forms names of the table tableNum.
IDs is optionnal and may not be passed to the routine.
if formKinds is passed, it is filled with the kind of each form. Possible values are:

0: no particular kind
1: Detail form
2: List form
3: Detail form for printing
4: List form for printing

ds_GetFormObjectMethodIDs

ds_GetFormObjectMethodIDs (formID;IDs ;varNames {;pageNums}) ← errorCode

formID	Numeric	→	ID of the form
Ids	Numeric Array	←	IDs of object methods
varNames	String/Text Array	←	Names of variables
pageNums	Numeric Array	←	Num. of the page containing the object
errorCode	Long Integer	←	Error Code (0 = no error)

This routine loads in synchronised arrays informations about objects methods of the form whose ID is formID (you get form Ids with ds_GetFormNames):

- Their IDs (usable with ds_MethodToText fo instance)
- The name of the variable/Field that holds the method
  For fields, the name
- The number of the page that contains the object

NOTICE: fields names are formated using this naming convention: number of the table on 3 chars + number of the field on 3 chars. When the table is the table of the form, its number is 0 (it is formated "000")
Examples:

Field #12 of the table of the form: "000012"

Field #8 of table #36 in a form of an other table: "036008"

It is up to the developer to extract the numbers and build the full name of the field.

To load the objects of one single form, use the ***ds_getFormObjectMethodIDs*** routine.

ds_DuplicateForm

ds_DuplicateForm (tableNum;nameOfOriginal;nameOfCopy) ← errorCode

tableNum	Integer	→	Table number
nameOfOriginal	String	→	Source form name
nameOfCopy	String	→	Created form name
errorCode	Long Integer	←	Error Code (0 = no error)

Duplicate the form nameOfOriginal. The copy is created in the same table. The form method and every object methods are also duplicated, likewise form properties (resizing options, event, inherited, …)

IMPORTANTE NOTE: after calling this routine, you must quit 4D, because the list of the forms of the table is not automatically updated.

ds_GetFormEvents(formID;events) ← errorCode

formID	Integer	→	Form ID
theEvents	Long Integer	←	List of events
errorCode	Long Integer	←	Error Code (0 = no error)

Returns in theEvents the list of checked events in the form. theEvents is return as a Long Integer, every bit tells if the event is checked or not. You must use the 4D constants of the theme « Form events» to get the state of a particular event. A value of –1 means « all events checked ». Think about declaring theEvents as a long integer.

Sample: how to know is the event « On Load » is checked ?

C_LONGINT($events)
$err:=***ds_GetFormEvents***(formID;$events)
if($events ?? On Load)

… On Load is checked …

ds_SetFormEvents

ds_SetFormEvents (formID;newEvents) ← errorCode

formID	Integer	→	Form ID
Events	Long Integer	→	New events values
errorCode	Long Integer	←	Error Code (0 = no error)

Modify the checked events of the form. See: ds_GetFormEvents.

ds_GetFormRulers

ds_GetFormRulers
(tableNum;formName;header;detail;break;footer;moreHeaders;moreBreaks) <- errorCode

tableNum	Integer	→	Table number
formName	String	→	Form name
Header	Integer	←	Main Header position
Detail	Integer	←	Detail position
Break	Integer	←	Main Break positiposition
Footer	Integer	←	Footer position
moreHeaders	Numeric Array	←	Other headers
moreBreaks	Numeric Array	←	Other breaks
errorCode	Long Integer	←	Error Code (0 = no error)

Return the values of form rulers. If there are no more than 1 header or 1 footer, the size of the arrays moreHeaders and moreBreaks will be 0.

ds_SetFormRulers

ds_SetFormRulers
(tableNum;formName;header;detail;break;footer;moreHeaders;moreBreaks) <- errorCode

tableNum	Integer	→	Table number
formName	String	→	Form name
Header	Integer	←	Main Header position
Detail	Integer	←	Detail position
Break	Integer	←	Main Break position
Footer	Integer	←	Footer position
moreHeaders	Numeric Array	←	Other headers

moreBreaks	Numeric Array	←	Other breaks
errorCode	Long Integer	←	Error Code (0 = no error)

Define form rulers. The routine returns an error if you passed invalid parameters (Header higher than Footer, …). You can't pass more than 9 more breaks and 10 more headers.

ds_GetFormMenubar

ds_GetFormMenubar (tableNum;formName;menuBar) ← errorCode

tableNum	Integer	→	Table number
formName	String	→	Form name
Menubar	Integer	←	Number of associated menu bar
errorCode	Long Integer	←	Error Code (0 = no error)

Get the menu bar associated to a form. A negative number means that « Menu Bar Active » is checked.

ds_SetFormMenubar

ds_SetFormMenubar (tableNum;formName;newMenuBar) ← errorCode

tableNum	Integer	→	Table number
formName	String	→	Form name
newMenuBar	Integer	→	Number of associated menu bar
errorCode	Long Integer	←	Error Code (0 = no error)

Define the menu bar associated to a form. Giving a negative value is as to check the box « Menu Bar Active ».

ds_GetFormMethod

ds_GetFormMethod (tableNum;formName;methodID) ← errorCode

tableNum	Integer	→	Table number
formName	String	→	Form name
methodID	Integer	←	Form method ID (0 = no form method)
errorCode	Long Integer	←	Error Code (0 = no error)

Retuns in methodID the ID of the form method associated to the form formName of the table tableNum. If this form doesn't have any form method, methodID is set to 0. After getting the form method ID, you can use ds_MethodToText and ds_TextToMethod.

ds_SetFormMethod

ds_SetFormMethod (tableNum;formName;newMethodID) ← errorCode

tableNum	Integer	→	Table number
formName	String	→	Form name
methodID	Integer	→	Method ID
errorCode	Long Integer	←	Error Code (0 = no error)

Change the form method associated to the form formName.

IMPORTANTE NOTICE: The routine doesn't check that newMethodID is an existing method ID. To stop associating a from method with a form you just have to pass 0 in methodID. Notice that the old form method, if any and if it was not a valid project method ID, stays in the structure, as an orphan.

ds_GetFormInfos

ds_GetFormInfos (formID;
resizable;fixedWidth;fixedHeight;
heightMini;heightMaxi;hauteurMini;hauteurMaxi;
hMarginOrWidth;vMarginOrHeight;
nomObjetDelimiteur;windowTitle;platForm
inheritedTableNum ;inheritedFormName) ← errorCode

formID	Integer	→	Form number
Resizable	Integer	←	Resizeable
FixedWidth	Integer	←	Fixed Width
FixedHeight	Integer	←	Fixed Height
minWidth	Integer	←	
maxWidth	Integer	←	
minHeight	Integer	←	
maxHeight	Integer	←	
HmarginOrWidth	Integer	←	horizontal Margin or Height
VmarginOrHeight	Integer	←	vertical Margin or Width
limitObjectName	String	←	
WindowTitle	String	←	
PlatForm	Integer	←	
inheritedTableNum	Integer	←	Inherited form table num.
inheritedFormName	String	←	Inherited form name

errorCode	Long Integer	←	Error Code (0 = no error)

This routine makes it possible for you to load most of form properties. Dues to some options are checked or not, some parameters may be negative.

DynamicStructure: Tables

ds_GetTriggerInfos

ds_GetTriggerInfos (tableNum;triggerID;onSaveNew;onSave;onDelete;onLoad)
← errorCode

tableNum	Integer	→	Table number
triggerID	Integer	←	Trigger method ID. 0 = no trigger
onSaveNew	Integer	←	
onSave	Integer	←	Trigger enabled (1)
onDelete	Integer	←	or disabled (0)
onLoad	Integer	←	
errorCode	Long Integer	←	Error Code (0 = no error)

Returns in triggerID the ID of the trigger method (0 if there isn't any for this table). This ID can be used with other routines of the plugin (ds_MethodToText, ds_GetObjectComments, …).
These values inform you about the activation state of the trigger for some database events. If the value is 1 the trigger is enabled, else if the value is 0 the trigger is disabled.
Notice that starting at 4D 6.7, those informations (except the trigger method ID) are already accessible using the 4D language.

ds_SetTriggerInfos

ds_SetTriggerInfos (tableNum;triggerID;onSaveNew;onSave;onDelete;onLoad)
← errorCode

tableNum	Integer	→	Table number
triggerID	Integer	→	new Trigger method ID, or 0 or -1.
onSaveNew	Integer	→	
onSave	Integer	→	enabled(1) or disabled(0) the trigger
onDelete	Integer	→	-1: do not modify
onLoad	Integer	→	
errorCode	Long Integer	←	Error Code (0 = no error)

Modify:

- The method to execute when the trigger is called. 0 means « no trigger method »
- The state enable/disable of this or that database event.

A value of -1 in any parameter means « do not modify this parameter ».

IMPORTANTE NOTICE: The routine doesn't verify that triggerID is an existing method ID. To stop associating a method to a trigger you just have to pass 0 in triggerID. Notice that

the old trigger method,if any and if it was not a valid project method ID, will still exists in the structure, as an orphan.

ds_TablesIDs(IDs)

ds_TablesIDs(IDs) ← errorCode

Ids	Numeric Array	←	List of table IDs
errorCode	Long Integer	←	Error Code (0 = no error)

Returns a list of tables IDs. Used only to get global comments on the table (see ds_GetObjectComments).

ds_SetTableName

ds_SetTableName (tableNum;newName) ← errorCode

tableNum	Entier	→	Table number
newName	Alpha	→	New table name
errorCode	Entier Long	←	Error Code (0 = no error)

Changes the name of the table number tableNum. Note that the new name is taken in account by 4D everywhere but in the Explorer.

ds_SetTableVisible

ds_SetTableVisible (tableNum;isVisible) ← errorCode

tableNum	Integer	→	Table number
isVisible	Integer	→	Set visible (1) or invisible(0)
errorCode	Long Integer	←	Error Code (0 = no error)

Modify the visibility of the table tableNum. To know if a table is visible, use 4D Pack.

---

ds_SetTableDeletion

ds_SetTableDeletion (tableNum;physicallyDeleted) ← errorCode

tableNum	Integer	→	Table number
physicallyDeleted	Integer	→	physicaly deleted(1) or no(0)
errorCode	Long Integer	←	Error Code (0 = no error)

---

ds_GetTablePosition

ds_GetTablePosition (tableNum;top;left;bottom;right) ← errorCode

tableNum	Integer	→	Table number
Top	Integer	←	Higher point position
Left	Integer	←	Left point position
Bottom	Integer	←	Lower point position
Right	Integer	←	Right point position
errorCode	Long Integer	←	Error Code (0 = no error)

Return the position (in the structure window) of the table tableNum.

ds_SetTablePosition

ds_SetTablePosition (tableNum;top;left;bottom;right) ← errorCode

tableNum	Integer	→	Table number
Top	Integer	→	Higher point position
Left	Integer	→	Left point position
Bottom	Integer	→	Lower point position
Right	Integer	→	Right point position
errorCode	Long Integer	←	Error Code (0 = no error)

Modify the position (in the structure window) of the table tableNum.

ds_GetTableColor

ds_GetTableColor (tableNum;isAutomatic;colorIndex) ← errorCode

tableNum	Integer	→	Table number
isAutomatic	Integer	←	Default color(1) or no(0)
colorIndex	Integer	←	Color Index if isAutomatic = 0
errorCode	Long Integer	←	Error Code (0 = no error)

Returns informations about the table color (in the structure window). If isAutomatic is 1, the Color is « Default ». Else, colorIndex is the color in the 256 colors palette of 4D. Numbers start at 1 and end at 255.

ds_SetTableColor

ds_SetTableColor (tableNum;automatic;newcolorIndex) ← errorCode

tableNum	Integer	→	Table number
isAutomatic	Integer	→	Default color(1) ou no (0)

colorIndex	Integer	→	Color
errorCode	Long Integer	←	Error Code (0 = no error)

Define informations about the table color (in the structure window). If isAutomatic is 1, the Color is « Default ». Else, colorIndex is the color in the 256 colors palette of 4D. Numbers start at 1 and end at 255.

ds_GetFieldColor

ds_GetFieldColor (tableNum;fieldNum; isAutomatic;colorIndex) ← errorCode

tableNum	Integer	→	Table number
fieldNum	Integer	→	Field number
isAutomatic	Integer	←	Default color(1) ou no (0)
colorIndex	Integer	←	Color
errorCode	Long Integer	←	Error Code (0 = no error)

Returns informations about the field color (in the structure window). If isAutomatic is 1, the Color is « Default ». Else, colorIndex is the color in the 256 colors palette of 4D. Numbers start at 1 and end at 255.

ds_SetFieldColor

ds_SetFieldColor (tableNum;fieldNum;automatic;newcolorIndex) ← errorCode

tableNum	Integer	→	Table number
fieldNum	Integer	→	Field number
isAutomatic	Integer	→	Default color(1) ou no (0)
colorIndex	Integer	→	Color
errorCode	Long Integer	←	Error Code (0 = no error)

Changes informations about the field color (in the structure window). If isAutomatic is 1, the Color is « Default ». Else, colorIndex is the color in the 256 colors palette of 4D. Numbers start at 1 and end at 255.

ds_SetFieldAttributes

ds_SetFieldAttributes (tableNum;fieldNum;isMandatory;isEnterable;noModif;
indexUnique;invisible) ← errorCode

tableNum	Integer	→	Table number
fieldNum	Integer	→	Field number

isMandatory	Integer	→	1 = mandatory, 0 = not mandatory
isEnterable	Integer	→	1 = enterable, 0 = not enterable
noModif	Integer	→	1 = is modifable, 0 is not modifiable
IndexUnique	Integer	→	1 -> field is indexed-unique
Invisible	Integer	→	1 -> Set invisible
errorCode	Long Integer	←	Error Code (0 = no error)

Modifiy the field attributes. Current values can be get with 4D commands. It is not possible to change the attributes of a field of kind subTable.

ds_SetFieldRelation

ds_SetFieldRelation (tableNum;fieldNum;relatedTable;relatedField) ← errorCode

tableNum	Integer	→	Table number
fieldNum	Integer	→	Field number
relatedTable	Integer	→	Related table number
relatedField	Integer	→	Related field number
errorCode	Long Integer	←	Error Code (0 = no error)

Change relation of the field.
If the types are invalids, the routine does nothing (ie, it is impossible to link a Long Integer field with a field which is of an other type).

ds_SetFieldProperties

ds_SetFieldProperties (tableNum;fieldNum;newName;newType;newStringSize;newList)
<- errorCode

tableNum	Integer	→	Table number
fieldNum	Integer	→	Field number
newName	String	→	New field name. "" = don’t change
newType	Integer	→	New type
newStringSize	Integer	→	Length (2-80) for String fields. –1 = don’t change
NewList	String	→	Associated list name. "" = don’t change
errorCode	Long Integer	←	Error Code (0 = no error)

Change a field properties.

A subtable field cannot be changed, and it is not allowed to change a « regular » field to a subtable. If newName is not "" and is already used by an other field of the table, the routines returns an error code and does nothing.
Pass in newType a value of the theme « Constants: Field and Variable Types »:

0	Is Alpha Field
1	Is Real
2	Is Text
3	Is Picture
4	Is Date
6	Is Boolean
8	Is Integer
9	Is LongInt
11	Is Time
30	Is BLOB

DynamicStructure: Menus

ds_MenuBarsID

ds_MenuBarsID (IDs) ← errorCode

Ids	Numeric Array	←	Menu bars IDs
errorCode	Long Integer	←	Error Code (0 = no error)

Return the list of menu bars IDs. The array is synchronised with the bar number. This ID can used with other routines of the plugin (ds_GetObjectComment, ds_MenusIDs, …)

ds_MenusIDs

ds_MenusIDs (menuBarID;menuIDs) ← errorCode

menuBarID	Long Integer	→	menubar ID
menuIDs	Numeric Array	←	array of menu IDs
errorCode	Long Integer	←	Error Code (0 = no error)

Returns the list of the IDs of the menus of the menubar ID menuBarID.

ds_GetMenuBarPICT

ds_GetMenuBarPICT (menuBarNum;menuBarPict) ← errorCode

menuBarNum	Long Integer	→	menubar range
menuBarPict	Picture	←	Menu bar picture
errorCode	Long Integer	←	Error Code (0 = no error)

Get the picture of the menu bar menuBarNum. If no image is found for this menu bar, menuBarPict is returned empty. This means that 4D will draw it's own picture, which is stored in its own resources as PICT ID 5002.

ds_SetMenuBarPICT

ds_SetMenuBarPICT (menuBarNum;newPict) ← errorCode

menuBarNum	Long Integer	→	menubar range
newPict	Image	→	New picture for this bar.
errorCode	Long Integer	←	Error Code (0 = no error)

Set the picture of the menu bar menuBarNum. If you pass an empty picture, the previous picture (if any) is removed, so that 4D will draw its own picture.

ds_GetMenuInfos

ds_GetMenuInfos (menuBarNum;menuRange;menuTitle;labels;methods) ← errorCode

menuBarNum	Long Integer	→	menubar range
menuRange	Long Integer	→	Menu range
MenuTitle	Text	←	Menu title
labels	Text Array	←	Labels
methods	Text Array	←	Methods
errorCode	Long Integer	←	Error Code (0 = no error)

Returns informations about the menu number menuRange in the menu bar number menuBarNum. Labels and methods are synchronised arrays returning item labels and 4D associated methods.

ds_SetMenuInfos

ds_SetMenuInfos (menuBarNum;menuRange;menuTitle;labels;methods) ← errorCode

menuBarNum	Long Integer	→	menubar range
menuRange	Long Integer	→	Menu range
MenuTitle	Text	→	New menu title ("" = do not modify)
labels	Text Array	→	Labels
methods	Text Array	→	Methods
errorCode	Long Integer	←	Error Code (0 = no error)

Modify informations of the menu number menuRange is the menu bar number menuBarNum.

IMPORTANTE NOTICE: Labels and methods must be synchronised and be exactly of the same size as the originals. It is not possible,with this routine, to increase or to decrease the number of items of the menu.

DynamicStructure: Database

ds_GetDatabaseProperties

ds_GetDatabaseProperties (selector;value) ← errorCode

selector	Long Integer	→	selector
value	Long Integer	←	value for this selector
errorCode	Long Integer	←	Error Code (0 = no error)

Return the value of a database parameter. These informations are the one that the developer can modify in the « Database properties » dialog.
Possible values of the selector are definied in the plugin as constants:

Constant	value	Returned value
kds_ StartupEnvironment	1	1 = Start in Design mode 2 = Start in User mode mode 3 = Start in Custom menus mode
kds_ ProgressIndicator	9	0 = Number 1 = Thermometers
kds_DragDropHighLight	10	0 = No effect 1 = Frame only 2 = Pattern 3 = Both
kds_MandatoryLogFile	5	0 = Use Log File 1 = Dont use Log File
kds_DeletionControl	3	0 = Disable Deletion Control 1 = Enable Deletion Control
kds_AutomaticTransaction	4	0 = Disable automatic transactions 1 = Enable automatic transactions
kds_ArobaseIsChar	11	0 = Don’t use @ as a char 1 = Use @ as a char
kds_4DOpenAllowed	2	0 = Disable 4D Open connexions 1 = Enable 4D Open connexions
kds_UserList	6	0 = Dont show user list 1 = Show user list
kds_SortUserList	7	0 = Don’t sort user list 1 = Sort user list

kds_FlushData	8	Delay to flush buffers, in ticks
kds_CacheMin	12	Minimum cache
kds_CacheMax	13	Maximum cache
kds_UseNewMemOnMac	14	0 = Disable new memory scheme on Mac 1 = Enable it
kds_WinBlockCount	15	Count of memory blocks (Windows only)
kds_WinOneBlockSize	16	One memory block size (Windows only)

Calling this routine on Mac with selectors kds_WinBlockCount or kds_WinOneBlockSize generates an error.

ds_SetDatabaseProperties

ds_SetDatabaseProperties (selector;newValue) ← errorCode

selector	Long Integer	→	selector
NewValue	Long Integer	→	New selector value
errorCode	Long Integer	←	Error Code (0 = no error)

Modify the value of a database parameter. See ds_GetDatabaseProperties for the possible selector values.
« kds_ArobaseIsChar » value can’t be set.

Calling this routine on Mac with selectors kds_WinBlockCount or kds_WinOneBlockSize generates an error.

ds_GetDatabaseMethodIDs

ds_GetDatabaseMethodIDs (onStartUp;
onServerStart;
onExit;
onServerShutDown;
onServerOpenConnexion;
onWebConnexion;
onServerCloseConnexion;
onWebLog) ← errorCode

onStartUp	Integer	←	Database « On startup » method ID
onServerStart	Integer	←	Database « On Server Startup » method ID

OnExit	Integer	←	Database « On exit » method ID
onServerShutDown	Integer	←	Database « On Server shut down » method ID
onServerOpenConnexion	Integer	←	Database « On Server Open Connexion » method ID
onWebConnexion	Integer	←	Database « On Web Connexion » method ID
onServerCloseConnexion	Integer	←	Database « On Server Close Connexion » method ID
onWebLog	Integer	←	Database « On Web Log » method ID
errorCode	Long Integer	←	Error Code (0 = no error)

Get all database methods IDs. A value of –1 means « No Database method ».

ds_SetDatabaseMethodIDs

```
ds_SetDatabaseMethodIDs ( onStartUp;
                          onServerStart;
                          onExit;
                          onServerShutDown;
                          onServerOpenConnexion;
                          onWebConnexion;
                          onServerCloseConnexion;
                          onWebLog) ← errorCode
```

onStartUp	Integer	→	New database « On startup » method ID
onServerStart	Integer	→	New database « On Server Startup » method ID
OnExit	Integer	→	New database « On exit » method ID
onServerShutDown	Integer	→	New database « On Server shut down » method ID
onServerOpenConnexion	Integer	→	New database « On Server Open Connexion » method ID
onWebConnexion	Integer	→	New database « On Web Connexion » method ID
onServerCloseConnexion	Integer	→	New database « On Server Close

onWebLog	Integer	→	Connexion » method ID New database « On Web Log » method ID
errorCode	Long Integer	←	Error Code (0 = no error)

Set the database methods IDs. A value of –1 means « no Database method ». Be carefull: if you pass –1 as database method, the previous code will be not be deleted and will become, if any and if the ID is not a Project method, « orphan» in the structure.

ds_GetStyleSheets

Get all Style sheets and their specifications. Arrays are synchronised, one line = one style sheet. Style sheets has been modified startting with 4D 681 (to be used with Mac OS X and Windows XP). The routine accepts 2 differents syntaxes, changing only by the number of expected parapeters.

Under 4D 6.7:

ds_GetStyleSheets (names;macFontNames;macFontSizes;macFontFaces;
ntFontNames;ntFontSizes;ntFontFaces,
winFontNames;winFontSizes;winFontFaces) ← errorCode

Names	Text Array	←	Style sheets names
macFontNames	Text Array	←	Mac fonts names
macFontSizes	Numeric Array	←	Mac fonts sizes
macFontFaces	Numeric Array	←	Mac fonts faces
ntFontNames	Text Array	←	WinNT fonts names
ntFontSizes	Numeric Array	←	WinNT fonts sizes
ntFontFaces	Numeric Array	←	WinNT fonts faces
winFontNames	Text Array	←	Win98 fonts names
winFontSizes	Numeric Array	←	Win98 fonts sizes
winFontFaces	Numeric Array	←	Win98 fonts faces
errorCode	Long Integer	←	Error Code (0 = no error)

Under 4D 6.8:

ds_GetStyleSheets(names;macFontNames;macFontSizes;macFontFaces;
ntFontNames;ntFontSizes;ntFontFaces;
osxFontNames; osxFontSizes; osxFontFaces;
xpFontNames;xpFontSizes;xpFontFaces) ← errorCode

Names	Text Array	←	Style sheets names
macFontNames	Text Array	←	Mac fonts names
macFontSizes	Numeric Array	←	Mac fonts sizes
macFontFaces	Numeric Array	←	Mac fonts faces
ntFontNames	Text Array	←	WinNT fonts names
ntFontSizes	Numeric Array	←	WinNT fonts sizes
ntFontFaces	Numeric Array	←	WinNT fonts faces
osxFontNames	Text Array	←	Mac OS X fonts names
osxFontSizes	Numeric Array	←	Mac OS X fonts sizes
osxFontFaces	Numeric Array	←	Mac OS X fonts faces
xpFontNames	Text Array	←	Windows XP fonts names
xpFontSizes	Numeric Array	←	Windows XP fonts sizes
xpFontFaces	Numeric Array	←	Windows XP fonts faces
errorCode	Long Integer	←	Error Code (0 = no error)

ds_SetStyleSheet

This routines changes a style sheet. Because style sheets have new formats starting with 4D 6.8.1 (to handle Mac OS X and Windows XP), the routine has 2 syntaxes, changing only by the number of parameters.

Under 4D6.7:

ds_SetStyleSheet(name;newName;
macFontName;macFontSize;macFontFace;
ntFontName;ntFontSize;ntFontFace;
winFontName;winFontSize;winFontFace) <- errorCode

Name	String	→	Style sheet to modify
newName	String	→	New name for the style sheet
macFontName	String	→	Mac font name
macFontSize	Integer	→	Mac font size
macFontFace	Integer	→	Mac font face
ntFontName	String	→	WinNT font name
ntFontSize	Integer	→	WinNT font size
ntFontFace	Integer	→	WinNT font face
winFontName	String	→	Win98 font name
winFontSize	Integer	→	Win98 font size
winFontFace	Integer	→	Win98 font face
errorCode	Long Integer	←	Error Code (0 = no error)

Under 4D6.8:

ds_SetStyleSheet(name;newName;
macFontName;macFontSize;macFontFace;
ntFontName;ntFontSize;ntFontFace;
osxFontName; osxFontSize; osxFontFace ;
xpFontName;xpFontSize;xpFontFace) <- errorCode

Name	String	→	Style sheet to modify
newName	String	→	New name for the style sheet
macFontName	String	→	Mac font name
macFontSize	Integer	→	Mac font size
macFontFace	Integer	→	Mac font face
ntFontName	String	→	WinNT font name
ntFontSize	Integer	→	WinNT font size
ntFontFace	Integer	→	WinNT font face
osxFontName	String	→	Mac OS X font name
osxFontSize	Integer	→	Mac OS X font size
osxFontFace	Integer	→	Mac OS X font face
xpFontName	String	→	Windows XP font name
xpFontSize	Integer	→	Windows XP font size
xpFontFace	Integer	→	Windows XP font face
errorCode	Long Integer	←	Error Code (0 = no error)

« Special fonts »

Under 4D 6.7 only (it seems that 4D 682 has a bug on this and it is the current version at the time DynamicStructure is released), you can set the font to « System font », « Application font », . . . by using a special char ias the font name :

For font…	…use:
System font	**Char**(1)
Application font	**Char** (2)
System small font	**Char** (2)

ds_GetFilters

ds_GetFilters (names;values) ← errorCode

Names	Text Array	←	Names of ormats/filters
Values	Numeric Array	←	Format/filters values

errorCode	Long Integer	←	Error Code (0 = no error)

Get the list of all formats/filters and their values in synchronised arrays.

ds_SetFilter

ds_SetFilter (filterName;newName;newValue) ← errorCode

filterName	String/Text	→	Filter/Format to modify
Newname	String/Text	→	New format/filter name
value	String/Text	→	Filter value
errorCode	Long Integer	←	errorCode

Modify the format/filter filterName.

IMPORTANT NOTE: filters are stored *by name* in forms. Changing a filter's name will not be dispatched in every form that uses it.

DynamicStructure: Miscellaneous

ds_GetListNames

ds_GetListNames (names{;IDs}) ← errorCode

Names	Text Array	←	List names Array
Ids	Numeric Array	←	Lists IDs Array
errorCode	Long Integer	←	Error Code (0 = no error)

Get names and IDs of every list of the database. IDs is optionnal and may not be passed to the routine.

ds_GetListUserModifiable

ds_GetListUserModifiable (listID;isModifiable) ← errorCode

ListID	Integer	→	List ID
IsModifiable	Integer	←	1 = List is modifiable
errorCode	Long Integer	←	Error Code (0 = no error)

Return 1 in isModifiable if the list ID listID is modifiable, 0 in other case.

ds_SetListUserModifiable

ds_SetListUserModifiable (listID;isModifiable) ← errorCode

ListID	Integer	→	List ID
IsModifiable	Integer	→	1 = List is modifiable, 0 = not modifiable
errorCode	Long Integer	←	Error Code (0 = no error)

Modify the status of a list (modifiable or not).

ds_GetComponentNames

ds_GetComponentNames (tabNoms;tabIDs) ← errorCode

TabNoms	Text Array	←	Components names array
TabIDs	Numeric Array	←	Components IDs array
errorCode	Long Integer	←	Error Code (0 = no error)

Get all installed component names and IDs. The Ids can then be used to get the global comment.

---

ds_GetObjectComments

ds_GetObjectComments (selector;id;fieldID;comment{ ;commentStyle}) <- errorCode

selector	Integer	→	selector
Id	Integer	→	Object ID
FieldID	Integer	→	Field number
Comment	Text	←	Comment
commentStyle	BLOB	←	Style of the comment
errorCode	Long Integer	←	Error Code (0 = no error)

Return the object comment. It is a global comment, writen in insider or in 4D.
selector values are:

kds_DataBaseComment	0
kds_GroupComment	1
kds_ObjectComment	2
kds_TableComment	3
kds_FieldComment	4
kds_FormComment	5
kds_MenuBarComment	6
kds_MenuComment	7

The ID expected is the one returned by other routines of the plugin such as ds_GetMethodNames, ds_GetFormNames, ds_TableIDs, … To get a field comment, you must pass the table ID (not the table number) and the field number.

New in version 1.1 : blobStyl

If blobStyl is passed, it will be filled with the style description of the comment. This is available only if the comment has been written in the Explorer of 4D, not if it has been writtent with 4D Insider. The blob contains a description of the style in the « styl » Macintosh format. Note that this format can be used with 4D Write under Windows. So, usually, the developer will load the comment, put texte and style in the clipboard and paste it in a 4D Wriite area:

```
C_BLOB($styl)
SET BLOB SIZE($styl;0)
$comment:=" "
$L_err:=ds_GetObjectComment(kds_ObjectComment ;methodID;0;$comment;$styl)
CLEAR CLIPBOARD
SET TEXT TO CLIPBOARD($comment)
APPEND TO CLIPBOARD(" styl ";$styl)
 `the clipboard can now be pasted in a 4D Write area, under Mac and Windows.
```

ds_SetObjectComments

ds_SetObjectComments (selector;ID;fieldID;newComment{ ;newStyle}) <- errorCode

selector	Integer	→	selector
Id	Integer	→	Object ID
FieldID	Integer	→	Field number
newComment	Text	→	Comment
newCode	Long Integer	←	Error Code (0 = no error)

Add a comment to an object. To remove the comment, pass an empty string.
See ds_GetObjectComments to have the different posible values for the selector.

New in version 1.1 : blobStyl

If passed, blobStyl must be a valid « styl » Macintosh stylized text descrption. Note that this will probably done by copying stylized text in a 4D Write area. This is the only way to get a valid « styl » under Window. So, usually, the developer will use 4D Write as a « comment editor », and once the comment is writtnt, he/she can select the comment, copy it, and read the clipboard before changing the comment:

```
C_BLOB($styl)
SET BLOB SIZE($newStyle;0)
C_TEXT($newStyle)
$newComment:=Get text from clipboard
GET CLIPBOARD("styl";$newStyle)
$L_err:=ds_SetObjectComment(kds_ObjectComment ;methodID;0;$newStyle;$newStyle)
```

ds_GetTipList

ds_GetTipList (tipNames;tipIDs) ← errorCode

TipNames	Text Array	←	Tips Array
TipIDs	Numeric Array	←	Tips Ids Array

errorCode	Long Integer	←	Error Code (0 = no error)

Get Tips list and IDs.

ds_SetTip

ds_SetTip (tipID;newName;newValue) ← errorCode

TipID	Integer	←	Tip ID
newName	String	→	New name
NewValue	String	→	New text
errorCode	Long Integer	←	Error Code (0 = no error)

Modify the tip.

ds_GetObjectModifDate

ds_GetObjectModifDate (selector;ID;modifDate;modifTime) ← errorCode

selector	Integer	→	Kind of the object
ID	Numeric	→	ID of the object
modifDate	Date	←	Modification date
modifTime	Time	←	Modification time
errorCode	Long Integer	←	Error Code (0 = no error)

Returns the date/time of last modification for the object whose ID is passed. The possible values for selector are:

1 : Method
2 : Table
3 : Form
4 : Menu bar
5 : Menu
6 : Tip

Exemple : recherche des méthodes projet modifiées cette semaine. Après l'exécution de la méthode, les tableaux names/visible/Ids ne contiennent que les méthodes modifiées les 7 derniers jours.

```
ARRAY TEXT(names;0)
ARRAY BOOLEAN(visible;0)
ARRAY INTEGER(IDs;0)
C_DATE($modifDate;$date)
```

```
C_TIME($modifTime)
 ` Get all project methods
$err:=ds_GetMethodNames    (names;"";visible;lDs)
$date:=Current date-7
 ` For each method...
For ($i;Size of array(lDs);1;-1)
 ` ...get the modification date...
 $err:=ds_GetObjectModifDate    (1;lDs{i};$modifDate;$modifTime)
 ` ...if it is < date checked, remove the method from the list.
 If ($modifDate<$date)
 DELETE ELEMENT(names;$i)
 DELETE ELEMENT(visible;$i)
 DELETE ELEMENT(lDs;$i)
 End if
End for
```

ds_GetPluginNames

ds_GetPluginNames (plugNames) <- errorCode

plugNames	String/Text array	←	Names of the running plugins
errorCode	Long Integer	←	Error Code (0 = no error)

Returns the names of all the active plugins, wherever they are localized.
Note that this routine loads the « stub plugins », whi somtimes have no name.

ds_GetPluginRoutines

ds_GetPluginRoutines (plugName ;plugRoutines{;selector}) <- errorCode

plugName	Text	→	Name of the plugin
plugRoutines	String/Text array	←	Names of the routines
selector	Numeric	→	Format of the names.
errorCode	Long Integer	←	Error Code (0 = no error)

Returns the list of the routines of the plugin named plugName. Depending on the value of selector, the names will be less or more long:

0 : full name of the routines (name with parameters as the plugin declare them for 4D)
    Example : "ds_GetMethodNames(&Y;&S;&Y;&Y):L"

1 : Name of the theme + « / » + full name of the routine.
    Example : "Methods/ds_GetMethodNames(&Y;&S;&Y;&Y):L"

2 : Name without the parameters
    Example : "ds_GetMethodNames"

4 : Name of the theme + « / » + Name without the parameters
    Example : "Methods /ds_GetMethodNames"

If the plugin is a « stub plugin » and it does not have any « STR# » resource describing its routines, ds_GetPluginRoutines returns error –192 (« Resource not found »).

ds_Preferences

ds_Preferences (selector ;value) <- errorCode

prefSelector	Entier	→	Value to Get/Set
prefValue	Entier long	→	New value
		←	Current value
errorCode	Long Integer	←	Error Code (0 = no error)

This routine lets the developer get/set preferenecs of the plugin. In version 1.1, it has only 2 possible values for prefSelector

1	Change the frequency's calls to 4D 's Scheduler while loading arrays
-1	Get the frequency's calls to 4D 's Scheduler while loading arrays.

This selector changes the behavior of 2 routines : ds_GetMethodNames and ds_GetObjectMethodIDs. Because on slow computers and when loading a hunge number of methods, loading arrays could take several seconds. During the loading loop, in version 1.0 of DynamicStructure, the 4D's scheduler was never called. The user could think that his/her computer was freezed. Under 4D Server, other processes did not get a chance to have some CPU time. In version 1.1, by default, the plugin uses internally a call similar to IDLE every 30 times in a loop (for those 2 routines only). This can be changed. Setting this value to 0 => « load arrays with no call to IDLE ». In this case, the arrays will be loaded faster, but the plugin will not give hand to the 4D's scheduler. Beyond 0, the more the value is, the less 4D has the hand.

DynamicStructure: Errors

Value	Meaning	Constant
0	No error	kdsERR_NoError
-5	Internal resource (Method, comment, …) not found	kdsERR_ResNotFound
-15000	Invalid type of array	kdsERR_BadArrayType
-30000	Not Registered	kdsERR_NotRegistered
-30001	Time Has Bombed	kdsERR_TimeBombed
-30002	Bad Version Of 4D	kdsERR_Bad4DVersion
-30003	Need At Least 67	kdsERR_NeedAtLeast67
-30004	Could Not Load Struct	kdsERR_CouldNotLoadStruct
-30005	Not On Windows	kdsERR_NotOnWindows
-30006	Not On Mac	kdsERR_NotOnMac
-30007	Not On 4D Client	kdsERR_NotOn4DClient
-30008	Not On Compiled	kdsERR_NotOnCompiled
-30009	Bad selector	kdsERR_BadSelector
-30010	Blob is invalid	kdsERR_InvalidBlob
-30011	Runs only with 4D Standalone	kdsERR_Only4DMono
-30100	Out Of Range Parameter	kdsERR_OutOfRangeParameter
-30101	Parameter Error (internal error)	kdsERR_ParamErr
-30102	32000 chars reached, can’t put more in text	kdsERR_32000CharReached
-30200	Object Is Locked	kdsERR_4DResLocked
-30201	Invalid Name	kdsERR_InvalidName
-30300	Bad Style SheetI D	kdsERR_BadStyleSheetID
-30301	Method Name Exists	kdsERR_MethodNameExists
-30302	Method Name Not Found	kdsERR_MethodNameNotFound
-30303	No Flow Chart	kdsERR_NoFlowChart
-30304	Invalid String Field Size	kdsERR_InvalidStringFieldSize
-30305	Invalid Field Kind	kdsERR_InvalidFieldKind
-30306	Get Var Only Compiled	kdsERR_GetVarOnlyCompiled
-30307	Form Not Found	kdsERR_FormNotFound
-30308	Error Reading Form	kdsERR_ErrorReadingForm
-30309	Bad Rulers Params	kdsERR_BadRulersParams
-30310	Duplicate Form Name	kdsERR_DuplicateFormName
-30311	Unknown Form Version (can’t parse form)	kdsERR_UnknnownFO4DVers
-30312	Form Parser Error (error while parsing the form)	kdsERR_FO4DParserError
-30313	ID Already Used	kdsERR_IDAlreadyUsed
-30314	Error while reading the form (« NullFromHandle »)	kdsERR_NullFormHandle
-30350	Max field count reached	kdsERR_MaxFieldCount
-30351	Duplicate table name	kdsERR_DuplicateTableName
-30351	Max table count reached	kdsERR_MaxTableCount
-30400	Can’t Change Component Object (protected)	kdsERR_CantChangeCompObject
-30401	Object Not Found	kdsERR_ObjectNotFound
-30402	Cant Get Protected Component Object	kdsERR_CantGetCompObject

-30403	Bad component ID	kdsERR_BadComponentID
-30500	Plug-in not found (bad name)	kdsERR_PluginNotFound