

DynamicStructure

Version 1.1.0 – Septembre 2002

©Osmose Éditeur - techsupport@osmose.net



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>

Nouveautés/Changements de la version 1.1.0

Corrections de bugs

- ***ds_GetObjectComment*** ne renvoyait pas toujours le texte correct si le commentaire était stylisé
- ***ds_GetStyleSheet*** renvoyait parfois du « garbage » quand la police choisie était « Police système » ou « Police application »
- ***ds_GetObjectMethodIDs*** ne renvoyait pas d'informations sur les variables/champs inclus dans un groupe, et ne faisait pas de distinction entre variables process/interprocess.

Nouvelles fonctionnalités

- ***ds_GetObjectComments*** et ***ds_SetObjectComments*** acceptent un paramètre supplémentaire : un blob contenant le style du commentaire, s'il y en a un. Ce style peut être ensuite manipulé avec 4D Write.
- ***ds_GetObjectMethodIDs*** accepte un paramètre supplémentaire optionnel : un tableau (synchronisé avec les précédents) indiquant le N° de la page dans laquelle se trouve la variable/le champ.
- ***ds_SetStyleSheet*** : il est possible de fixer la police d'une feuille à « Police Système », « Police application » ou « Petite police système » en utilisant des caractères spéciaux.
- ***ds_GetStyleSheet*** et ***ds_SetStyleSheet*** acceptent des paramètres supplémentaires sous Mac OS X et 4D 68 pour gérer les nouvelles feuilles de style.

Nouvelles routines

- ***ds_GetPluginNames*** renvoie la liste des plug-ins installés, quel que soit le dossier Mac/Win4DX dans lequel ils se trouvent.
- ***ds_GetPluginRoutines*** renvoie la liste des routines d'un plug-in.
- ***ds_GetFormObjectMethodIDs*** fait la même chose que ***ds_GetObjectMethodIDs***, mais est limitée à un seul formulaire.
- ***ds_Preferences*** permet de modifier le comportement de DynamicStructure dans le cas de chargement de listes volumineuses sur des machines lentes.

Présentation

Il s'agit d'un outil extrêmement puissant qui va considérablement améliorer la productivité des développeurs 4D. De l'aide à la documentation automatique du développement à la manipulation dynamique des structures par programmation, DynamicStructure couvre de très nombreux domaines d'informations qui ne sont pas accessibles avec le langage 4D.

Compatibilité

DynamicStructure est compatible avec les versions de 4D et les systèmes d'exploitation suivants :

- 4D 6.7 et les systèmes d'exploitation supportés par 4D 6.7.
Note : il faut la version 6.7.2 au minimum. Version « la plus récente » recommandée (6.7.4 en septembre 2002)
- 4D 6.8 et les systèmes d'exploitation supportés par 4D 6.8.
- **4D Version 6.5:** *le plug-in est supporté à partir de la version 672 de 4D. Cependant, il a été testé avec 4D 659 pendant le bêta-tests public de mars 2002 et il n'a pas été relevé d'anomalies particulière ; il est probable qu'une version future de DynamicStructure soit officiellement compatible avec 4D 6.5 (utiliser la version 6.5.9r2). Si vous l'utilisez avec cette version de 4D, n'hésitez pas à nous faire connaître les problèmes éventuellement rencontrés et nous les corrigerons le plus vite possible, afin qu'une prochaine version de DynamicStructure supporte officiellement la 6.5. Merci également de nous faire savoir quelles routines vous utilisez sans problèmes en 659.*

Installation

Le plug-in se glisse dans un dossier Mac4DX ou Win4DX.

- Avec 4D 6.7 sous Mac OS comme sous Windows :
Utiliser DynamicStructure.4DX (et DynamicStructure.RSR sous Windows)
- Avec 4D 6.8
 - Sous Windows, utiliser DynamicStructure.4DX et DynamicStructure.RSR
 - Sous Mac (OS 9.2 comme OS X), utiliser la version carbonisée du plug-in : DynamicStructure.4CX

/

Sérialisation du plugin - Mode de démonstration

Une fois installé dans son dossier, le plug-in fonctionne en mode de démonstration jusqu'à ce qu'un numéro de licence valide soit entré en utilisant la routine `ds_Register` :

`ds_Register`(selector;licence) -> errorCode

selector Entier Type de licence

Licence Alpha N° de licence

Valeurs possibles pour Selector

1 : licence développeur Mac OS

2 : Licence développeur Windows

3 : licence de déploiement Runtime Compilé

4 : Licence de déploiement Runtime Compilé Illimité pour Une Application.

Si la licence est valide, la routine renvoie 0 (pas d'erreur) sinon, elle renvoie -30 000 et le plug-in continue de fonctionner en mode de démonstration.

Il y a trois types de licence :

- La *licence Développeur*, qui permet le développement en interprété sur une plateforme, et permet de tester en compilé. Il faut une licence par machine utilisant le plug-in (on peut utiliser la même licence pour plusieurs développements sur la même machine)
- La *license Runtime Compilé*, qui permet le fonctionnement en compilé uniquement et ne fait rien en interprété. C'est la licence à avoir si vous utilisez DynamicStructure en compilé (utilisation au runtime compilé pour un client spécifique par exemple). Il faut une licence par machine utilisant le pluug-in.
- Enfin, la *Runtime Illimité pour une Application*, qui permet le fonctionnement (sur une plateforme) en compilé mais convient aux cas où vous souhaitez diffuser un grand nombre de copies de votre application.

Les licences sont donc "par machine". En client serveur, il faut avoir une licence par poste utilisant le plug-in.

Note : Mac OS = une plateforme, quel que soit le système : OS 9 ou X

Exemples :

/

- Un développeur utilise DynamicStructure uniquement pour optimiser ses développements, mieux documenter son code, ... Dans ses développements livrés, DynamicStructure n'est jamais utilisé.
-> Il doit acquérir une *licence développeur* seulement
- Ce développeur utilise en plus DynamicStructure dans des développements compilés
-> Il doit acquérir une *licence développeur* (pour le développement) plus autant de licences *Runtime Compilé* qu'il y aura de machines utilisant le plug-in en compilé.
- Un développeur utilise DynamicStructure sur son Mac et sur son PC : il devra acquérir deux licences *développeur*, une par plateforme.
- DynamicStructure est utilisé avec 4D mono serveur web, dans une structure compilée : une seule licence *Runtime Compilé* est nécessaire

Pour toute question sur les licences, contactez infos@osmose.net ou consultez notre site web <http://www.osmose.net> ou ceux des distributeurs.

Limitations de la version de démonstration :

- Seuls 20 "Set" et 50 "Get" peuvent être effectués. Au-delà, le plug-in ne fait rien, et renvoie l'erreur "Plug-in non enregistré".
- Quand une routine s'est exécutée sans erreur, le plug-in renvoie -30 000 ("non enregistré" au lieu de 0.

Exemple de méthode de sérialisation :

```

C_ENTIER LONG($err)
C_ALPHA(255;$licence)
`

$licenceMac:="" ` <- Numéro de licence développeur Mac OS
$licenceWin:="" ` <- Numéro de licence développeur Windows
$licenceRUN:="" ` <- N° d elicence runtime
`

Si (Non(Application  compilee)) ` Interprété : Licence développeur
  Si (wBB_IsMac )
    $err:=ds_Register (1;$licenceMac) ` 1 = licence développeur Mac
  Sinon
    $err:=ds_Register (2;$licenceWin) ` 2 = licence développeur Windows
  Fin de si
Sinon ` Licence "runtime compilé"
  ` Si le numéro de licence commence par "UNL",
  ` c'est la licence "Illimitée pour une application". Sinon,
  ` c'est une licene de déploiement runtime compilé simple.
  Si ($licenceRUN="UNL@")
    $err:=ds_Register (4;$licenceRUN)
  Sinon
    $err:=ds_Register (3;$licenceRUN)
  Fin de si
  `
  ` NOTE : la licence développeur permet également des tests en compilé :
```

/

```
` Si (wßB_IsMac )  
` $err:=ds_Register (1;"DEV....")  
` Sinon  
` $err:=ds_Register (2;"DEV....")  
` Fin de si  
`
```

Fin de si**Si (\$err=-30000) ` -30 000 = plug-in non sérialisé****. . .****Fin de si**

Précautions d'emploi, principes d'utilisation

Les routines du plug-in *renvoyant* une information (ds_Getxxx) ne modifient *jamais* la structure. Elles peuvent être utilisées sans autres restrictions que les celles liées à leurs fonctionnalités : par exemple, on ne peut lire les paramètres de base depuis un 4D Client, ou récupérer le texte d'une méthode dans une base compilée.

SI VOUS NE DEVEZ LIRE QU'UNE SEULE PARTIE, QUE CE SOIT CELLE-CI

- Toutes les routines *modifiant* une information changent *immédiatement* le contenu même du fichier de structure. Les modifications ne sont pas locales au process, elles sont « définitives » (jusqu'au prochain changement) exactement comme si l'objet avait été modifié depuis le mode structure.
- Il est donc **très important** que le développeur qui modifie un objet gère *lui-même* les conséquences de ses modifications et les accès concurrentiels. Notamment en environnement client serveur, et/ou s'il modifie des objets fondamentaux de l'environnement d'exécution, tels qu'une méthode associée à un trigger, une méthode associée à une méthode base, ... Car les changements qu'il provoque sont immédiatement appliqués à toute la base de données. L'utilisation de sémaphore devient vite indispensable pour sécuriser les accès.
- Enfin, en cours de développement, il faut **travailler sur des copies** de la structure dès lors que l'on développe un outil permettant de modifier dynamiquement la structure ouverte (ds_TextToMethod, ...)

En mode client serveur, les objets modifiés sont verrouillés le temps de la modification. S'ils étaient déjà utilisés sur un autre poste, le plug-in ne fait rien et renvoie l'erreur -30 200 (objet verrouillé). Il n'y a aucun verrouillage avec 4D Mono.

Problème commun à tous les plug-ins : l'interpréteur de 4D

En mode interprété, il peut arriver que les variables soient retypées après l'appel à un plug-in. Dans la définition de leurs routines, les plug-ins signalent à 4D les types de paramètres attendus : entier, texte, image, tableau, ... Lorsque l'on passe une variable à une routine de plug-in, au retour de l'appel, la variable aura pris le type défini par le plug-in.

Cela concerne tous les plu-ins et n'est pas spécifique à DynamicStructure.

/

Prenons l'exemple suivant : la routine *RoutineDePlug-in* attend un entier long comme 1^{er} paramètre :

C_REEL(\$leReel)

RoutineDePlug-in(\$leReel)

=> Après l'appel, \$leReel est devenu un entier long.

Cela peut entraîner, par la suite, de la perte d'information, car un entier long "pèse" 4 octets quand un réel en pèse 8. Si après l'appel vous tentez de mettre dans \$leReel une valeur supérieur à MAXLONG, 4D « alignera » la valeur.

Cela peut être particulièrement gênant avec les tableaux. La routine précédente attend un entier long comme paramètre. Imaginons que vous souhaitiez utiliser l'élément courant d'un tableau: vous utilisez donc le tableau par son nom...

TABLEAU TEXTE(monTab ;10)

... suite du code ...

RoutineDePlug-in(monTab)

=> après l'appel, monTab est devenu un entier long !

Le tableau monTab est devenu une variable de type entier long : le contenu du tableau est perdu. Dans ce genre de situation, il est préférable d'utiliser une variable "tampon" :

TABLEAU TEXTE(monTab ;10)

... suite du code ...

\$buffer:=monTab

RoutineDePlug-in(\$buffer)

Par contre, quand un tableau est attendu par un plug-in, il n'y a pas retypage : un tableau Entier ne se transforme pas en tableau texte ou en variable simple.

Enfin, ce problème n'existe pas en compilé.

Limitations de certaines routines

(se reporter à la description de chaque commande)

Routines imposant de quitter/relancer 4D immédiatement

ds_NewMethod

ds_SetMethodName

ds_DuplicateForm

/

Routines inutilisables en environnement compilé :

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

Routines inutilisables depuis 4D Client:

- ds_GetDatabaseProperties
- ds_SetDatabaseProperties
- ds_NewMethod
- ds_SetMethodName
- ds_DuplicateForm

Routines inutilisables depuis 4D Server:

- ds_NewMethod
- ds_SetMethodName
- ds_DuplicateForm

Points communs à toutes les routines

Toutes les routines du plug-in sans exception sont des fonctions retournant un code d'erreur : « errorCode », de type Entier long. Une valeur retournée de 0 signifie « pas d'erreur ». Toute autre valeur correspond à un code d'erreur du système, de 4D ou du plug-in lui-même (cf annexe « Codes d'erreur » dans ce cas).

Quand un paramètre passé doit être de type « Tableau numérique », vous pouvez utiliser indifféremment un tableau Entier, Entier long ou Réel

Quand un paramètre passé doit être de type « Tableau texte », vous pouvez passer indifféremment un tableau Alpha ou texte. Si la définition de longueur de chaîne d'un tableau alpha est trop petite, les chaînes reçues seront tronquées (exemple : un tableau alpha 5 pour récupérer les noms des méthodes)

Quand un paramètre attendu doit être de type « numérique », vous pouvez passer un entier, un entier long, ou un réel. Cependant, reportezvous à la section « Retypage des variables en interprété par 4D ».

Le plug-in ne renvoie jamais et ne modifie jamais les éléments des composants (il renvoie un code d'erreur dans ce cas).

DynamicStructure : Méthodes

/

Avec DynamicStructure, vous pouvez manipuler n'importe quel type de méthode : projet, objet, trigger, formulaire, base. Vous récupérez l'identifiant de la méthode et utilisez cet identifiant avec d'autres routines telles que ds_MethodToText, ds_GetObjectComment, ...

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

| | | | |
|-----------|-------------------|---|----------------------------------|
| Names | Tableau Texte | ← | Tableau de nom des méthodes |
| Filter | Alpha | → | Filtre de recherche des noms |
| Visible | Tableau numérique | ← | Visibilité |
| | Ou Booléen | | |
| Ids | Tableau numérique | ← | Id des méthodes |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Cette routine charge dans des tableaux synchronisés :

- Les noms de toutes les méthodes projet
- Leur visibilité
- Leurs Ids

Le paramètre filter permet de ne charger que les méthodes dont le nom commence par filter. Les tableaux Visible et IDs ainsi que le filtre sont optionnels, ils peuvent en pas être passés en argument.

Attention : après compilation, les Ids des méthodes sont modifiés.

Version 1.1 : il est possible de paramétrer DynamicStructure avec ds_Preferences, afin que les méthodes soient chargées plus ou moins rapidement et que le plugin donne plus ou moins souvent la main au scheduler de 4D. Charger de nombreuses méthodes sur une machine lente pouvait donner à l'utilisateur une impression de plantage pendant quelques secondes, le temps pour le plug-in de remplir les tableaux.

ds_MethodToText(methodID;code) ← errorCode

| | | | |
|-----------|-------------|---|----------------------------------|
| methodID | Entier Long | → | ID de méthode |
| code | Texte | ← | Le code |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Retourne le texte d'une méthode dont l'ID est passée en paramètre.

/

ds TextToMethod(newCode;methodID) ← errorCode

| | | | |
|-----------|-------------|---|----------------------------------|
| NewCode | Texte | → | Code de la méthode |
| methodID | Entier Long | → | Id de la méthode |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Modifie le code d'une méthode dont l'ID est passée en paramètre.

NOTE : si la méthode est ouverte en mode structure, le contenu de la fenêtre n'est pas mis à jour automatiquement. Pour retrouver le nouveau code, il faut choisir « Version enregistrée » dans le menu « Fichier ». Si vous refermez la fenêtre sans avoir choisi cette commande, le texte actuellement affiché devient le nouveau code de la méthode.

ds OpenMethod(methodName{ ;lineNum) ← errorCode

| | | | |
|------------|-------------|---|----------------------------------|
| methodName | Alpha | → | Nom de la méthode |
| lineNum | Entier | → | N° de ligne à surligner |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Ouvre, en mode structure, la méthode MethodName. Si lineNum est passé et valide, la ligne numéro lineNum est sélectionnée. Si l'utilisateur courant n'a pas accès au mode structure ou si la méthode est une méthode protégée d'un composant, une erreur est retournée.

ds OpenMethodByID(methodID{ ;lineNum) ← errorCode

| | | | |
|-----------|-------------|---|----------------------------------|
| methodID | Entier | → | ID de la méthode |
| lineNum | Entier | → | N° de ligne à surligner |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Ouvre, en mode structure, la méthode d'ID methodID. Si lineNum est passé et valide, la ligne numéro lineNum est sélectionnée. Si l'utilisateur courant n'a pas accès au mode structure ou si la méthode est une méthode protégée d'un composant, une erreur est retournée. Cette routine permet d'ouvrir les méthodes objets, base, formulaires dont les Ids sont récupérées par d'autres routines du plug-in (ds_GetObjectMethodIDs, ds_GetFormMethod, ...)

/

ds NewMethod(newMethodName {;ID}) ← errorCode

| | | | |
|---------------|-------------|---|----------------------------------|
| newMethodName | Alpha | → | Nom de la nouvelle méthode |
| ID | Entier | → | ID forcée (ou 0) |
| | | ← | ID de la méthode créée |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Crée une nouvelle méthode de nom newMethodName.

ID contient au retour l'ID de la méthode créée. Passer 0 ou -1 pour laisser 4D trouver une ID. Si vous passez une autre valeur que 0 ou -1, cette valeur sera utilisée pour forcer l'ID. Si cette ID existe déjà, la routine ne crée pas la méthode et renvoie un code d'erreur.

NOTE IMPORTANTE : Les versions actuelles de 4D (67/68) ne permettent pas à un plug-in d'informer 4D qu'une nouvelle méthode a été créée : il faut impérativement quitter/relancer 4D après avoir créé une nouvelle méthode de cette manière. Il est possible de créer plusieurs nouvelles méthodes d'affilée, puis de quitter. Il ne faut pas mettre dans les méthodes du code utilisant le nom d'une des nouvelles méthodes, il ne sera pas reconnu lors de la tokenisation. Le code suivant crée 50 méthodes et leur ajoute un commentaire en en-tête, rappelant qu'elles ont été créées avec ds_NewMethod.

```
` Create_50_Methods
C_ENTIER LONG($i;$err;$id)
C_TEXTE($prefix;$code;$name)
$i:=0
$prefix:="aMethod_"
$code:="" Crée par ds_NewMethod, "+Chaine(Date du jour)
$code:=$code+" - "+Chaine(Heure courante;h mn )+"`"+(" "*50)
Boucle ($i;1;50)
$name:=$prefix+Chaine($i;"00")
$id:=-1
$err:=ds_NewMethod ($name;$id)
Si ($err#0)
ALERTE("New method #"+Chaine($i)+" => error #"+Chaine($err))
$i:=50
Sinon
$err:=ds_TextToMethod ("` "+$name+$code;$id)
```

/

```

Si ($err#0)
ALERTE("ds_TextToMethod#" + Chaine($i) + " => error #" + Chaine($err))
$i:=50
Fin de si
Fin de si
Fin de boucle
,
QUITTER 4D
,

```

ds SetMethodName(oldName ;newName) ← errorCode

| | | | |
|-----------|-------------|---|-------------------------------------|
| oldName | Alpha | → | Méthode dont on veut changer le nom |
| newName | Alpha | → | Nouveau nom pour la méthode |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Modifie le nom de la méthode oldName. Si oldName n'est pas une méthde projet, si newName est le nom d'une méthode existante ou si newName est un nom invalide (vide ou de plus de 31 caractères), le routine ne fait rien et renvoie une erreur.

NOTE IMPORTANTE : Les versions actuelles de 4D (67/68) ne permettent pas au plug-in d'informer 4D que le nom d'une méthode a été modifiée: il faut impératiement quitter/relancer 4D après avoir modifié le nom d'une méthode de cette manière. Notez que le nouveau nom n'est pas reconnu lors de la tokenisation : si, après avoir modifié un nom, vous l'utilisez dans le code d'une méthode, il ne sera pas reconnu comme « méthode », mais comme variable process. Quitter/relancer 4D corrige le problème.

ds SetMethodVisible(methodName;isVisible) ← errorCode

| | | | |
|------------|-------------|---|--|
| methodName | Alpha | → | Nom de la méthode à modifier |
| isVisible | Numérique | → | Nouvelle visibilité (1 = visible, 0 = invisible) |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Modifie la visibilité de la méthode methodName. Une méthode invisible n'apparaît pas dans les différents dialogue de 4D : « Exécuter une méthode » en utilisation directe, éditeur de formule dans les états, ...

/

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

| | | | |
|-----------|---------------------|---|----------------------------------|
| IDs | Tableau numérique | ← | Ids des méthodes objets |
| varNames | Tableau alpha/texte | ← | Nom des variables |
| formIDs | Tableau numérique | ← | ID des formulaires |
| pageNums | Tableau numérique | ← | N° de page |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Cette routine charge dans des tableaux synchronisés des informations sur *toutes* les méthodes objets de la structure :

- Leurs IDs (utilisable avec ds_MethodToText par exemple)
- Les noms des objets propriétaires de la méthode (variable ou champ). Les variables interprocess sont préfixées du signe usuel selon la plateforme (« » sous Mac, « <> » sous windows)
- Les IDs des formulaires contenant l'objet
- Les N° des pages contenant l'objet

NOTE : Le nom du champ est formaté de la façon suivante : le numéro de table sur 3 caractères + Le numéro du champ sur 3 caractères également. Quand la table est celle du formulaire, le numéro de table vaut 0 (il est formaté "000").

Exemples :

champ N°12 de la table du formulaire : "000012"

champ N°8 de la table N°36 dans un formulaire d'une autre table : "036008"

Il vous appartient d'extraire les noms des champs.

Pour charger les Ids des objets d'un formulaire uniquement, utiliser

ds_GetFormObjectMethodIDs.

Version 1.1 : il est possible de paramétrer DynamicStructure avec ds_Preferences, afin que les méthodes soient chargées plus ou moins rapidement et que le plugin donne plus ou moins souvent la main au scheduler de 4D. Charger de nombreuses méthodes sur une machine lente pouvait donner à l'utilisateur une impression de plantage pendant quelques secondes, le temps pour le plug-in de remplir les tableaux.

ds_TokenizeStatement(text;tokens) ← **errorCode**

| | | | |
|------|-------|---|-------|
| text | Texte | → | Texte |
|------|-------|---|-------|

/

| | | | |
|-----------|-------------|---|----------------------------------|
| tokens | Texte | ← | Tokens |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Tokenise le texte passé en paramètre. Une seule ligne (séparateur : retour chariot) peut être tokenisée.

| | | | |
|---|-------------|---|----------------------------------|
| <u>ds ByteSwapTokens(tokens;swapped)</u> | | ← | errorCode |
| Tokens | Texte | → | Tokens originaux |
| swapped | Texte | ← | Tokens Byte swappés |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Transforme un token Mac en token PC et vice-versa, selon la plateforme d'exécution. Cela permet d'échanger ses tokens d'une plateforme à l'autre.

| | | | |
|---|-------------|---|----------------------------------|
| <u>ds DetokenizeStatement(tokens;text)</u> | | ← | errorCode |
| Tokens | Texte | → | Tokens |
| Text | Texte | ← | Texte |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Transforme un token en texte lisible.

| | | | |
|---------------------------------------|-------------|---|----------------------------------|
| <u>ds ExecuteToken(tokens)</u> | | ← | errorCode |
| Tokens | Texte | → | Tokens |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Execute le code tokenisé. Exécuter un token st plus rapide que'appeler «EXECUTER ». Si vous devez exécuter plusieurs fois la même instruction, l'exécution sera plus rapide si vous tokenisez l'expression puis exécutez les tokens.

DynamicStructure : Formulaires

/

ds_GetFormNames(names;tableNum{;lds{;formKinds{;input{;outPut{}}}) ←
errorCode

| | | | |
|-----------|-------------------|---|----------------------------------|
| Names | Tableau Texte | ← | Nom des Formulaires |
| TableNum | Entier | → | Numéro de la table |
| lds | Tableau numérique | ← | Tableau d' lds des Formulaires |
| formKinds | Tableau numérique | ← | Tableau des types de formulaire |
| input | Alpha/Texte | ← | Nom du formulaire entrée |
| output | Alpha/Texte | ← | Nom du formulaire sortie |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Charge les nom et les lds des formulaires de la table tableNum.

lds est optionnel et peut ne pas être passé.

formKinds est un tableau numérique synchronisé avec les précédents. Il renseigne sur le type de formulaire. Voici les codes des valeurs possibles retournées dans formKinds :

- 0 : aucun type particulier
- 1 : Formulaire détaillé
- 2 : Formulaire liste écran
- 3 : Formulaire impression détaillé
- 4 : Formulaire impression liste

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

| | | | |
|-----------|---------------------|---|----------------------------------|
| formID | Numérique | → | ID du formulaire |
| lds | Tableau numérique | ← | lds des méthodes objets |
| varNames | Tableau alpha/texte | ← | Nom des variables |
| pageNums | Tableau numérique | ← | N° de page |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Cette routine charge dans des tableaux synchronisés des informations sur les méthodes objets du formulaire formID:

- Leurs IDs (utilisable avec ds_MethodToText par exemple)
- Les noms des objets propriétaires de la méthode (variable ou champ). Les variables interprocess sont préfixées du signe usuel selon la plateforme (« » sous Mac, « <> » sous windows)
- Les IDs des formulaires contenant l'objet

- Les N° des pages contenant l'objet

NOTE : Le nom du champ est formaté de la façon suivante : le numéro de table sur 3 caractères + Le numéro du champ sur 3 caractères également. Quand la table est celle du formulaire, le numéro de table vaut 0 (il est formaté "000").

Exemples :

champ N°12 de la table du formulaire : "000012"

champ N°8 de la table N°36 dans un formulaire d'une autre table : "036008"

Il vous appartient d'extraire les noms des champs.

ds DuplicateForm(tableNum;nameOfOriginal;nameOfCopy) ← errorCode

| | | | |
|----------------|-------------|---|----------------------------------|
| tableNum | Entier | → | Numéro de Table |
| nameOfOriginal | Alpha | → | Nom du formulaire source |
| nameOfCopy | Alpha | → | Nom du formulaire créé |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Duplique le formulaire nameOfOriginal. La copie est placée dans la même table. La méthode formulaire ainsi que toutes les méthodes objet sont également dupliquées, de même que les propriétés du formulaire (options de redimensionnement, taquets, événements, ...)

LIMITATION IMPORTANTE : Après avoir appelé cette routine, il est nécessaire de quitter/relancer 4D, car la liste des formulaires de la table n'est pas automatiquement mise à jour.

ds GetFormEvents(formID;events) ← errorCode

| | | | |
|-----------|-------------|---|----------------------------------|
| formID | Entier | → | ID du Formulaire |
| theEvents | Entier Long | ← | Événements cochés |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Renvoie dans theEvents la liste des événements cochés pour le formulaire. theEvents est renvoyé sous la forme d'un entier long dont chaque bit indique si l'événement est coché ou non. Il suffit d'utiliser les constantes 4D du thème « Événements formulaire » pour connaître l'état de tel ou tel événement. Une valeur de -1 signifie « tous les événements sont cochés ». Pensez à déclarer theevents en entier long explicitement.

/

Exemple : savoir si l'événement « Sur chargement » est coché.

C_ENTIER LONG(\$events)

\$err := **ds_GetFormEvents**(id ;\$events)

Si(\$events ?? Sur chargement)

...Sur chargement est coché...

ds SetFormEvents(formID;newEvents) ← errorCode

| | | | |
|-----------|-------------|---|----------------------------------|
| FormID | Entier | → | ID du Formulaire |
| Events | Entier Long | → | Nouvelles valeurs des événements |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Modifie les événements cochés pour le formulaire. Reportez-vous à ds_GetFormEvents.

**ds GetFormRulers(tableNum;formName;header;detail;break;footer;
moreHeaders;moreBreaks) ← errorCode**

| | | | |
|-------------|-------------------|---|-------------------------------------|
| tableNum | Entier | → | Numéro de table |
| formName | Alpha | → | Nom de formulaire |
| header | Entier | ← | Position entête |
| detail | Entier | ← | Position corps |
| freak | Entier | ← | Position rupture |
| footer | Entier | ← | Position pied de page |
| moreHeaders | Tableau numérique | ← | Tableau des Entêtes supplémentaires |
| moreBreaks | Tableau numérique | ← | Tableau des Ruptures suplémentaires |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Renvoie les valeurs des taquets d'un formulaire. Si aucun en-tête ou rupture supplémentaire n'est posé dans le formulaire, les tableaux moreHeaders et moreBreaks sont retournés de taille 0.

**ds SetFormRulers(tableNum;formName;header;detail;break;
footer;moreHeaders;moreBreaks) ← errorCode**

| | | | |
|----------|--------|---|-------------------|
| tableNum | Entier | → | Numéro de table |
| formName | Alpha | → | Nom de formulaire |

/

| | | | |
|-------------|-------------------|---|--------------------------------------|
| header | Entier | ← | Position entête |
| detail | Entier | ← | Position corps |
| break | Entier | ← | Position rupture |
| footer | Entier | ← | Position pied de page |
| moreHeaders | Tableau numérique | ← | Tableau des Entêtes supplémentaires |
| moreBreaks | Tableau numérique | ← | Tableau des Ruptures supplémentaires |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Définit les taquets d'un formulaire. La routine renvoie une erreur s'il y a une abération dans les positions (par exemple une rupture située avant un en-tête). Vous ne pouvez (comme dans l'éditeur de formulaire) passer plus de 9 ruptures supplémentaires et pas plus de 10 en-têtes supplémentaires.

ds GetFormMenubar(tableNum;formName;menuBar) ← errorCode

| | | | |
|-----------|-------------|---|-------------------------------------|
| tableNum | Entier | → | Numéro de table |
| formName | Alpha | → | Nom de formulaire |
| menubar | Entier | ← | Numéro de la barre de menu associée |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

renvoie dans menuBar le numéro de la barre de menu associée à un formulaire. Un chiffre négatif signifie que la case « Barre de menus active » est cochée.

ds SetFormMenubar(tableNum;formName;newMenuBar) ← errorCode

| | | | |
|------------|-------------|---|-------------------------------------|
| tableNum | Entier | → | Numéro de table |
| formName | Alpha | → | Nom de formulaire |
| newMenuBar | Entier | → | Numéro de la barre de menu associée |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Définit la barre de menu associée à un formulaire. Passer une valeur négative revient à cocher la boîte « Barre de menus active ».

ds GetFormMethod(tableNum;formName;methodID) ← errorCode

| | | | |
|----------|--------|---|-----------------|
| tableNum | Entier | → | Numéro de table |
|----------|--------|---|-----------------|

/

| | | | |
|-----------|-------------|---|---|
| formName | Alpha | → | Nom de formulaire |
| methodID | entier | ← | ID de la méthode formulaire (0 = pas de méthode formulaire) |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Renvoie dans methodID l'ID de la méthode formulaire associée au formulaire formName de la table numéro tableNum. Si le formulaire ne possède aucun méthode formulaire, methodID vaut 0.

ds SetFormMethod(tableNum;formName;newMethodID) ← errorCode

| | | | |
|-----------|-------------|---|----------------------------------|
| tableNum | Entier | → | Numéro de table |
| formName | Alpha | → | Nom de formulaire |
| methodID | entier | → | ID de la méthode à associer |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Permet de changer de méthode formulaire associée au formulaire formName.

NOTE IMPORTANTE : la routine ne vérifie pas que newMethodID correspond à une méthode existante. Pour ne plus associer de méthode au formulaire, passer 0 dans methodID. Notez cependant que dans ce cas, l'ancienne méthode formulaire, s'il y en avait une et que son ID ne correspondait pas à celui d'une méthode projet, reste dans la structure, comme « orpheline ».

ds GetFormInfos(formID;

resizable;fixedWidth;fixedHeight;
 largeurMini;largeurMaxi;hauteurMini;hauteurMaxi;
 hMarginOrWidth;vMarginOrHeight;
 nomObjetDelimiteur;windowTitle;platForm
 inheritedTableNum ;inheritedFormName) ← errorCode

| | | | |
|-------------|--------|---|----------------------|
| FormID | Entier | → | Numéro du formulaire |
| Resizable | Entier | ← | Retaillable |
| FixedWidth | Entier | ← | Largeur fixée |
| FixedHeight | Entier | ← | Longueur fixée |
| LargeurMini | Entier | ← | largeur minimum |
| LargeurMaxi | Entier | ← | largeur maximum |

/

| | | | |
|--------------------|-------------|---|-------------------------------------|
| HauteurMini | Entier | ← | Longueur minimum |
| HauteurMaxi | Entier | ← | Longueur maximum |
| HmarginOrWidth | Entier | ← | Marge horizontale ou Largeur |
| VmarginOrHeight | Entier | ← | Marge verticale ou Longueur |
| NomObjetDelimiteur | Alpha | ← | Nom de l'objet délimiteur |
| WindowTitle | Alpha | ← | Titre de la fenêtre |
| PlatForm | Entier | ← | |
| inheritedTableNum | Entier | ← | N° de la table du formulaire hérité |
| inheritedFormName | Alpha | ← | Nom du formulaire hérité |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Cette routine permet de charger la plupart des propriétés du formulaire.

Selon les options cochées, les paramètres prendront des valeurs différentes, parfois négatives.

DynamicStructure : Tables

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

| | | | |
|-----------|-------------|---|---|
| tableNum | Entier | → | Numéro de table |
| triggerID | Entier | ← | ID de la méthode trigger. 0 : pas de méthode trigger |
| onSaveNew | Entier | ← | <div style="display: inline-block; vertical-align: middle;"> <div style="display: inline-block; vertical-align: middle;">Trigger activé (1) ou désactivé (0)</div> </div> |
| onSave | Entier | ← | |
| onDelete | Entier | ← | |
| onLoad | Entier | ← | |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Cette routine retourne dans triggerID l'ID de la méthode trigger (0 s'il n'y en a pas pour la table). Cet ID peut être utilisée avec les autres routines du plug-in (ds_MethodToText, ds_GetObjectComments, ...)

Les autres valeurs renseignent sur l'état d'activation ou non des triggers pour tel ou tel événement moteur. Une valeur renvoyée à 1 signifie que le trigger est activé, sinon la valeur vaut 0. Notez que depuis 4D 6.7, ces informations (sauf l'ID e la méthode trigger) sont déjà accessibles par le langage 4D.

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

| | | | |
|-----------|-------------|---|---|
| tableNum | Entier | → | Numéro de table |
| triggerID | Entier | → | ID de la méthode à utiliser ou -1 |
| onSaveNew | Entier | → | <div style="display: inline-block; vertical-align: middle;"> <div style="display: inline-block; vertical-align: middle;">Activer (1) ou désactiver (0) le trigger -1 : ne pas modifier</div> </div> |
| onSave | Entier | → | |
| onDelete | Entier | → | |
| onLoad | Entier | → | |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Modifie :

- la méthode à exécuter lorsque le trigger est appelé. Passer 0 signifie « pas de méthode trigger »
- L'activation/désactivation du trigger pour tel ou tel événement moteur

Un paramètre à -1 signifie « ne pas modifier ».

/

NOTE IMPORTANTE : la routine ne vérifie pas que triggerID correspond à une méthode existante. Pour ne plus associer de méthode au formulaire, passer 0 dans triggerID. Notez cependant que dans ce cas, l'ancienne méthode trigger, s'il y en avait une et que son ID ne correspondait pas à celui d'une méthode projet, restera dans la structure, comme « orpheline ».

ds TablesIDs(tabIDs) ← errorCode

| | | | |
|-----------|-------------------|---|----------------------------------|
| TabIDs | Tableau numérique | ← | Listes des ID des tables |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Donne la liste des IDs des tables. Sert à lire les commentaires Insider avec ds_GetObjectComments.

ds SetTableName(tableNum;newName) ← errorCode

| | | | |
|-----------|-------------|---|----------------------------------|
| TableNum | Entier | → | N° de table |
| newName | Alpha | → | Nouveau nom pour la table |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Modifie le nom de la table numéro tableNum.

Particularité : le changement de nom est pris en compte immédiatement partout *sauf* dans l'explorateur.

ds SetTableVisible(tableNum;isVisible) ← errorCode

| | | | |
|-----------|-------------|---|------------------------------------|
| tableNum | Entier | → | Numéro de la table |
| isVisible | Entier | → | Rendre visible (1) ou invisible(0) |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Modifie la visibilité de la table numéro tableNum.

Rappel : pour savoir si une table est visible, on utilisera 4D Pack, plug-in fournit avec 4D.

ds SetTableDeletion(tableNum;physicallyDeleted) ← errorCode

/

| | | | |
|-------------------|-------------|---|------------------------------------|
| tableNum | Entier | → | Numéro de la table |
| physicallyDeleted | Entier | → | Suppression physique (1) ou non(0) |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Modifie le statut « Suppression physique » de la table.

Rappel : pour connaître le statut de la table, on utilisera 4D Pack, plug-in fournit avec 4D.

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

| | | | |
|-----------|-------------|---|----------------------------------|
| TableNum | Entier | → | N° de table |
| Top | Entier | ← | Position point haut |
| Left | Entier | ← | Position point gauche |
| Bottom | Entier | ← | Position point bas |
| Right | Entier | ← | Position point droit |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Retourne les coordonnées (dans la fenêtre de structure) d'une table dont on donne le numéro.

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

| | | | |
|-----------|-------------|---|----------------------------------|
| TableNum | Entier | → | N° de table |
| Top | Entier | → | Position point haut |
| Left | Entier | → | Position point gauche |
| Bottom | Entier | → | Position point bas |
| Right | Entier | → | Position point droit |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Modifie, dans la fenêtre de structure, les coordonnées d'une table dont on passe le numéro.

ds_GetTableColor(tableNum;isAutomatic;colorIndex) ← errorCode

| | | | |
|----------|--------|---|-------------|
| TableNum | Entier | → | N° de table |
|----------|--------|---|-------------|

/

| | | | |
|-------------|-------------|---|--|
| IsAutomatic | Entier | ← | Couleur par défaut (1) ou non (0) |
| ColorIndex | Entier | ← | Index de la couleur (si isAutomatic = 0) |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Envoie des information sur la couleur (dans la fenêtre de structure) d'une table. Si isAutomatic vaut 1, la couleur est celle « par défaut ». Sinon, colorIndex correspond au numéro dans la palette 256 couleurs de 4D. La numérotation des couleurs commence à 0.

ds_SetTableColor(tableNum;automatic;newColorIndex) ← errorCode

| | | | |
|-------------|-------------|---|-----------------------------------|
| TableNum | Entier | → | N° de table |
| IsAutomatic | Entier | → | Couleur par défaut (1) ou non (0) |
| ColorIndex | Entier | → | Couleur |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Définit la couleur d'une table dans le fenêtre de structure. Si isAutomatic vaut 1, 4D utilisera la couleur par défaut. Sinon, il faut passer dans colorIndex l'indice de la couleur dans la palette 256 couleurs de 4D (valeurs e 0-255)

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

| | | | |
|-------------|-------------|---|----------------------------------|
| TableNum | Entier | → | N° de table |
| FieldNum | Entier | → | N° de champ |
| isAutomatic | Entier | ← | Couleur par défaut |
| ColorIndex | Entier | ← | Couleur |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Récupère la couleur d'un champ d'une table. Si isAutomatic vaut 1, la couleur est celle « par défaut ». Sinon, colorIndex correspond au numéro dans la palette 256 couleurs de 4D. La numérotation des couleurs commence à 0

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

| | | | |
|----------|--------|---|-------------|
| TableNum | Entier | → | N° de table |
|----------|--------|---|-------------|

/

| | | | |
|-------------|-------------|---|----------------------------------|
| FieldNum | Entier | → | N° de champ |
| isAutomatic | Entier | → | Couleur Auto ? |
| ColorIndex | Entier | → | Couleur |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Change la couleur d'un champ d'une table. Si isAutomatic vaut 1, 4D utilisera la couleur par défaut. Sinon, il faut passer dans colorIndex l'indice de la couleur dans la palette 256 couleurs de 4D (valeurs e 0-255)

**ds SetFieldAttributes(tableNum;fieldNum;isMandatory;isEnterable;noModif;
indexUnique;invisible) ← errorCode**

| | | | |
|-------------|-------------|---|---|
| TableNum | Entier | → | N° de table |
| FieldNum | Entier | → | N° de champ |
| isMandatory | Entier | → | 1 = donne l'attribut Obligatoire, 0 le retire |
| isEnterable | Entier | → | 1 = donne l'attribut saisissable, 0 le retire |
| noModif | Entier | → | 1 = donne l'attribut modifiable, 0 le retire |
| IndexUnique | Entier | → | 1 -> le champ devient indexé-unique |
| Invisible | Entier | → | 1 -> rend le champ invisible |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Modifie les attributs d'un champ. Les valeurs courantes sont récupérables avec les commandes 4D.

**ds SetFieldRelation(tableNum;fieldNum;relatedTable;relatedField) ←
errorCode**

| | | | |
|--------------|-------------|---|----------------------------------|
| TableNum | Entier | → | N° de table |
| FieldNum | Entier | → | N° de champ |
| RelatedTable | Entier | → | N° de lable liée |
| RelatedField | Entier | → | N° du champ lié |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Change les relations d'un champ.

Si les types ne sont pas adaptés, la routine ne fait rien (par exemple on ne peut relier un champ entier long avec un champ d'un autre type que Entier long)

/

**ds_SetFieldProperties(tableNum;fieldNum;newName;newType;
newStringSize;newList) ← errorCode**

| | | | |
|---------------|-------------|---|--|
| TableNum | Entier | → | N° de table |
| FieldNum | Entier | → | N° de champ |
| NewName | Alpha | → | Nouveau Nom du Champ. "" = ne pas changer |
| NewType | Entier | → | Nouveau type. -1 = ne pas modifier |
| NewStringSize | Entier | → | Longueur (2-80) pour les champs Alpha. -1 = ne pas changer |
| NewList | Alpha | → | Nom de l'énumération associée. "" = ne pas changer |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Change les propriétés d'un champ : nom, type, énumération associée On ne peut pas modifier les champs de type soustable, ni transformer un champ non-soustable en un champ soustable. Si newName n'est pas vide et qu'un champ de même nom existe déjà dans la table, la routine renvoie une erreur et ne fait rien

Vous passez dans newType une valeur du thème « Constantes : Types des Champs et Variables » :

| | |
|----|--------------------|
| 0 | Est un champ alpha |
| 1 | Est un numérique |
| 2 | Est un texte |
| 3 | Est une image |
| 4 | Est une date |
| 6 | Est un booléen |
| 8 | Est un entier |
| 9 | Est un entier long |
| 11 | Est une heure |
| 30 | Est un BLOB |

DynamicStructure : Menus

/

ds MenuBarsID(tabIDs) ← errorCode

| | | | |
|-----------|-------------------|---|----------------------------------|
| TabIDs | Tableau numérique | ← | IDs des barres de menus |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Récupère la liste des Ids des barres de menus. Le tableau est synchronisé avec le numéro de la barre.

ds MenusIDs(menuBarID;menuIDs) ← errorCode

| | | | |
|-----------|-------------------|---|----------------------------------|
| menuBarID | Entier Long | → | ID de la barre de menu |
| menuIDs | Tableau numérique | ← | IDs des menus |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Récupère la liste des Ids des menus de la barre menuBarID.

ds GetMenuBarPICT(menuBarNum;menuBarPict) ← errorCode

| | | | |
|-------------|-------------|---|----------------------------------|
| menuBarNum | Entier Long | → | Numéro de la barre de menu |
| menuBarPict | Image | ← | Image de la barre |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Récupère l'image de la barre de menu numéro menuBarNum. Si aucune image n'est associée à cette barre de menus, menuBarPict est retournée vide.

ds SetMenuBarPICT(menuBarNum;newPict) ← errorCode

| | | | |
|------------|-------------|---|----------------------------------|
| menuBarNum | Entier Long | → | Numéro de la barre de menu |
| newPict | Image | → | Nouvelle Image pour cette barre |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Change l'image de la barre de menu numéro menuBarNum.

/

ds GetMenuInfos(menuBarNum;menuRange;menuTitle;labels;methods) ←
 errorCode

| | | | |
|------------|---------------|---|----------------------------------|
| menuBarNum | Entier Long | → | Numéro de la barre de menu |
| menuRange | Entier Long | → | Numéro du menu |
| MenuTitle | Texte | ← | Titre du menu |
| labels | Tableau Texte | ← | Liste des libellés |
| methods | Tableau texte | ← | Liste des méthodes |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Renvoie les informations du menu numéro menuRange dans la barre numéro menuBarNum. Labels et methods sont des tableaux synchronisés renvoyant respectivement les libellés des items et le nom de la méthode 4D associée.

ds SetMenuInfos(menuBarNum;menuRange;menuTitle;labels;methods) ←
 errorCode

| | | | |
|------------|---------------|---|--|
| menuBarNum | Entier Long | → | Numéro de la barre de menu |
| menuRange | Entier Long | → | Numéro du menu |
| MenuTitle | Texte | → | Nouveau titre du menu ("" = ne pas modifier) |
| labels | Tableau Texte | → | Liste des libellés |
| methods | Tableau texte | → | Liste des méthodes |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Modifie les informations du menu numéro menuRange dans la barre numéro menuBarNum.

NOTE IMPORTANTE : labels et methods doivent être synchronisés et contenir le même nombre d'éléments que le menu original. Il n'est pas possible, avec cette routine, de diminuer ou d'augmenter le nombre d'items du menu.

/

DynamicStructure : Base

/

ds GetDatabaseProperties(selector,value) ← errorCode

| | | | |
|-----------|-------------|---|----------------------------------|
| Selector | Entier Long | → | selecteur |
| Value | Entier Long | ← | valeur du selecteur |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Récupère la valeur d'un paramètre de la base de données. Il s'agit des informations accessibles dans le dialogue « Propriétés de la base ». Les valeurs possibles pour selecteur sont définies dans des constantes du plug-in :

| Constante | Valeur | Information renvoyée |
|--------------------------|--------|---|
| kds_ StartupEnvironment | 1 | 1 = Démarrage mode structure 2 = Démarrage moe Utilisation Directe 3 = Démarrage mode Menus créés |
| kds_ ProgressIndicator | 9 | 0 = Nombres 1 = Thermomètres |
| kds_DragDropHighLight | 10 | 0 = aucun effet 1 = cadre seul 2 = motif 3 = les deux |
| | | |
| kds_MandatoryLogFile | 5 | 0 = historique non obligtoire 1 = historique obligatoire |
| kds_DeletionControl | 3 | 0 = pas de contrôle d'intégrité référentielle 1 = contrôle d'intégrité référentielle actif |
| kds_AutomaticTransaction | 4 | 0 = pas de transactions automatiques en saisie 1 = transactions automatiques en saisie |
| kds_ArobaselsChar | 11 | 0 = ne pas considérer @ comme un caractère 1 = considérer @ comme un caractère |
| | | |
| kds_4DOpenAllowed | 2 | 0 = ne pas autoriser les connexions 4D Open 1 = autoriser les connexions 4D open |
| kds_UserList | 6 | 0 = ne pas afficher la liste des utilisateurs 1 = afficher la liste des utilisateurs |
| kds_SortUserList | 7 | 0 = ne pas trier la liste de sutilisateurs 1 = trier la liste de sutilisateurs |
| | | |

/

| | | |
|---------------------|----|---|
| kds_FlushData | 8 | Délait d'écriture du cache |
| kds_CacheMin | 12 | Valeur du réglage Cache minimum |
| kds_CacheMax | 13 | Valeur du réglage Cache Maximum |
| kds_UseNewMemOnMac | 14 | 0 = ne pas utiliser le nouveau système d'allocation mémoire sous Mac 1 = l'utiliser |
| kds_WinBlockCount | 15 | Nombre de blocs mémoire sous Windows |
| kds_WinOneBlockSize | 16 | Taille d'un bloc sous Windows |

L'appel de cette routine sous Mac avec les sélecteurs kds_WinBlockCount ou kds_WinOneBlockSize renvoie une erreur.

ds_SetDatabaseProperties(selector;newValue) ← errorCode

| | | | |
|-----------|-------------|---|----------------------------------|
| Selector | Entier Long | → | Selecteur |
| newValue | Entier Long | → | Nouvelle valeur du selecteur |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Modifie la valeur d'un paramètre de la base de données.

Reportez-vous à ds_GetDatabaseProperties pour connaître les valeurs possibles pour selector.

On ne peut pas changer la valeur de « kds_ArobaselsChar ».

Sous Mac, utiliser les sélecteurs kds_WinBlockCount ou kds_WinOneBlockSize renvoie une erreur. Et ne modifie rien.

ds_GetDatabaseMethodIDs(onStartup;

```

onServerStart;
onExit;
onServerShutDown;
onServerOpenConnexion;
onWebConnexion;
onServerCloseConnexion;
onWebLog) ← errorCode

```

| | | | |
|-----------|--------|---|---|
| OnStartup | Entier | ← | ID de la méthode base « Sur Ouverture » |
|-----------|--------|---|---|

/

| | | | |
|------------------------|-------------|---|--|
| OnServerStart | Entier | ← | ID de la méthode base « Sur démarrage serveur » |
| OnExit | Entier | ← | ID de la méthode base « sur fermeture » |
| OnServerShutDown | Entier | ← | ID de la méthode base « Sur arrêt serveur » |
| OnServerOpenConnexion | Entier | ← | ID de la méthode base » sur ouverture connexion serveur» |
| OnWebConnexion | Entier | ← | ID de la méthode base » sur connexion web » |
| OnServerCloseConnexion | Entier | ← | ID de la méthode base » sur fermeture connexion serveur» |
| OnWebLog | Entier | ← | ID de la méthode base » Sur authentification web» |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Retourne les Ids de toutes les méthodes Base. -1 signifie « Pas de méthode base ».

ds_SetDatabaseMethodIDs(onStartup;

```

onServerStart;
onExit;
onServerShutDown;
onServerOpenConnexion;
onWebConnexion;
onServerCloseConnexion;
onWebLog) ←    errorCode

```

| | | | |
|-----------------------|--------|---|--|
| OnStartup | Entier | → | ID de la méthode base « Sur Ouverture » |
| OnServerStart | Entier | → | ID de la méthode base « Sur démarrage serveur » |
| OnExit | Entier | → | ID de la méthode base « sur fermeture » |
| OnServerShutDown | Entier | → | ID de la méthode base « Sur arrêt serveur » |
| OnServerOpenConnexion | Entier | → | ID de la méthode base » sur ouverture connexion serveur» |

/

| | | | |
|------------------------|-------------|---|--|
| OnWebConnexion | Entier | → | ID de la méthode base » sur connexion web » |
| OnServerCloseConnexion | Entier | → | ID de la méthode base » sur fermeture connexion serveur» |
| OnWebLog | Entier | → | ID de la méthode base » Sur authentification web» |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Change les Ids de toutes les méthodes de la Base de Données.

Passer -1 signifie « pas de méthode base ». Attention : si vous passez -1 comme paramètre d'une méthode base, l'ancien code n'est pas supprimé et devient « orphelin » dans la structure.

ds GetStyleSheets

Cette routine charge des des tableaux l'ensemble des feuilles de styles et leurs spécifications. Les tableaux sont synchronisés, chaque ligne correspond à un style. Les feuilles de style ont été modifiées à partir de la version 6.8 de 4D afin de tenir compte de Mac OS X et de Windows XP. La routine accepte ainsi deux syntaxes différentes, variant sur le nombre de paramètres uniquement.

Syntaxe avec 4D 6.7

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

| | | | |
|--------------|-------------------|---|----------------------------------|
| names | Tableau Texte | ← | Noms des feuilles de style |
| macFontNames | Tableau Texte | ← | Noms des polices Mac |
| macFontSizes | Tableau Numérique | ← | Tailles des polices Mac |
| macFontFaces | Tableau Numérique | ← | Styles des polices Mac |
| ntFontNames | Tableau Texte | ← | Noms des polices WinNT |
| ntFontSizes | Tableau Numérique | ← | Tailles des polices WinNT |
| ntFontFaces | Tableau Numérique | ← | Styles des polices WinNT |
| winFontNames | Tableau Texte | ← | Noms des polices Win98 |
| winFontSizes | Tableau Numérique | ← | Tailles des polices Win98 |
| winFontFaces | Tableau Numérique | ← | Styles des polices Win98 |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

/

Syntaxe avec 4D 6.8

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

| | | | |
|--------------|-------------------|---|----------------------------------|
| names | Tableau Texte | ← | Noms des feuilles de style |
| macFontNames | Tableau Texte | ← | Noms des polices Mac |
| macFontSizes | Tableau Numérique | ← | Tailles des polices Mac |
| macFontFaces | Tableau Numérique | ← | Styles des polices Mac |
| ntFontNames | Tableau Texte | ← | Noms des polices Windows NT |
| ntFontSizes | Tableau Numérique | ← | Tailles des polices Windows |
| ntFontFaces | Tableau Numérique | ← | Styles des polices Windows |
| osxFontNames | Tableau Texte | ← | Noms des polices Mac OS X |
| osxFontSizes | Tableau Numérique | ← | Tailles des polices Mac OS X |
| osxFontFaces | Tableau Numérique | ← | Styles des polices Mac OS X |
| xpFontNames | Tableau Texte | ← | Noms des polices Windows XP |
| xpFontSizes | Tableau Numérique | ← | Tailles des polices Windows XP |
| xpFontFaces | Tableau Numérique | ← | Styles des polices Windows XP |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

ds_SetStyleSheet

Cette routine permet de modifier une feuille de style. Compte tenu des nouvelles feuilles de style apparues avec 4D 6.8, elle admet deux syntaxes, variant uniquement sur le nombre de paramètres.

Sous 4D 6.7 :

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

| | | | |
|-------------|--------|---|---------------------------------------|
| name | Alpha | → | Nom de la feuille de style à modifier |
| newName | Alpha | → | Nouveau nom de la feuille de style |
| macFontName | Alpha | → | Nom de la police Mac |
| macFontSize | Entier | → | Taille de la police Mac |
| macFontFace | Entier | → | Style de la police Mac |
| ntFontName | Alpha | → | Nom de la police WinNT |
| ntFontSize | Entier | → | Taille de la police WinNT |

/

| | | | |
|-------------|-------------|---|----------------------------------|
| ntFontFace | Entier | → | Style de la police WinNT |
| winFontName | Alpha | → | Nom de la police Win98 |
| winFontSize | Entier | → | Taille de la police Win98 |
| winFontFace | Entier | → | Style de la police Win98 |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Sous 4D 6.8 :**ds_SetStyleSheet**(name;newName;

macFontName;macFontSize;macFontFace;

ntFontName;ntFontSize;ntFontFace;

osxFontName; osxFontSize; osxFontFace ;

xpFontName;xpFontSize;xpFontFace) <- errorCode

| | | | |
|-------------|-------------|---|---------------------------------------|
| name | Alpha | → | Nom de la feuille de style à modifier |
| newName | Alpha | → | Nouveau nom de la feuille de style |
| macFontName | Alpha | → | Nom de la police Mac |
| macFontSize | Entier | → | Taille de la police Mac |
| macFontFace | Entier | → | Style de la police Mac |
| ntFontName | Alpha | → | Nom de la police Windows NT |
| ntFontSize | Entier | → | Taille de la police Windows NT |
| ntFontFace | Entier | → | Style de la police Windows NT |
| osxFontName | Alpha | → | Nom de la police Mac OS X |
| osxFontSize | Entier | → | Taille de la police Mac OS X |
| osxFontFace | Entier | → | Style de la police Mac OS X |
| xpFontName | Alpha | → | Nom de la police Windows XP |
| xpFontSize | Entier | → | Taille de la police Windows XP |
| xpFontFace | Entier | → | Style de la police Windows XP |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Polices « spéciales »

Sous 4D 67 uniquement (la version 4D 6.8.2, version active lors de la sortie de DynamicStructure 1.1, ne permet pas cette manipulation), il est possible de demander les polices « Police application », ... en utilisant un caractère spécial comme nom de la police :

| <u>Pour la police...</u> | <u>...utiliser :</u> |
|--------------------------|----------------------|
| Police système | Caractere(1) |
| Police application | Caractere(2) |
| Grande police système | Caractere(1) |
| Petite police système | Caractere(2) |

/

ds GetFilters(names;values) ← errorCode

| | | | |
|-----------|-------------------|---|----------------------------------|
| Names | Tableau Texte | ← | Noms des formats/filtres |
| Values | Tableau numérique | ← | Valeurs des formats/filtres |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Récupère la liste des formats/filtres et leurs valeurs dans des tableaux synchronisés

ds SetFilter(filterName;newName;newValue) ← errorCode

| | | | |
|------------|-------------|---|-----------------------------------|
| filterName | Alpha/Texte | → | Nom du format/filtre à modifier |
| Newname | Alpha/Texte | → | Nouveau nom pour le format/filtre |
| Value | Alpha/Texte | → | Valeur du filtre |
| errorCode | Entier Long | ← | errorCode |

Modifie le format/filtre filterName

ATTENTION : les filtres sont stockés dans les formulaires par leur nom, pas par une ID. Si vous changez le nom d'un filtre utilisé dans un formulaire, le changement ne sera pas automatiquement reporté dans tous les formulaires l'utilisant.

DynamicStructure : Divers

/

ds_GetListNames(names{;IDs}) ← errorCode

| | | | |
|-----------|-------------------|---|----------------------------------|
| Names | Tableau Texte | ← | Tableau de nom des énumérations |
| Ids | Tableau numérique | ← | Tableau de IDs des énumérations |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Permet de récupérer les noms et les IDs de l'ensemble des énumérations de la structure.

IDs est optionnel et peut ne pas être passé

ds_GetListUserModifiable(listID;isModifiable) ← errorCode

| | | | |
|--------------|-------------|---|----------------------------------|
| ListID | Entier | → | ID de l'énumération |
| IsModifiable | Entier | ← | 1 = l'énumération modifiable |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Renvoi 1 dans isModifiable si l'énumération d'ID listID est modifiable, 0 dans le cas contraire.

ds_SetListUserModifiable(listID;isModifiable) ← errorCode

| | | | |
|--------------|-------------|---|--|
| ListID | Entier | → | ID de l'énumération |
| IsModifiable | Entier | → | 1 = l'énumération modifiable, 0 = non modifiable |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Modifie le statut « Énumération modifiable » de l'énumération d'ID listID.

ds_GetComponentNames(tabNoms;tabIDs) ← errorCode

| | | | |
|-----------|-------------------|---|----------------------------------|
| TabNoms | Tableau Texte | ← | Tableau de nom des composants |
| TabIDs | Tableau numérique | ← | Tableau d'Ids des composants |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

/

Retourne les nom des composants installés et leurs Ids.

ds_GetObjectComments

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

| | | | |
|-----------------|-------------|---|----------------------------------|
| commentSelector | Entier | → | Selecteur |
| Id | Entier | → | Id de l'objet |
| fieldID | Entier | → | N° du champ |
| objectComment | Texte | ← | Commentaire |
| blobStyl | BLOB | ← | BLOB décrivant le style du texte |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Renvoie le commentaire d'un objet. Il s'agit du commentaire global, écrit dans Insider ou dans 4D.

Les valeurs possibles pour commentSelector sont :

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

L'ID attendue est l'ID renvoyée par les diverses routines du plug-in (ds_GetMethodNames, ds_getFormNames, ds_TableIDs, ...).

Pour récupérer les commentaires des champs, il faut passer l'ID de la table (pas son numéro) et le numéro du champ.

Ajout version 1.1 : blobStyl

Si blobStyl est passé, il contiendra àèu retour de fonction la description du style du commentaire. Cette fonctionnalité est accessible uniquement dans le cadre d'une stylisation du texte depuis l'Explorateur de 4D, pas depuis 4D Insider. BlobStyl contiendra une description au format « styl » Macintosh. Pour exploiter ce style, il faut utiliser le presse-papiers et 4D Write (notamment sous Windows). En pratique, il suffit d'écrire le texte du commentaire dans le presse-papiers puis d'y ajouter le blob, en précisant « styl » somme type de données :

/

```

C_BLOB($styl)
FIXER TAILLE BLOB($styl;0)
$comment:=" "
$L_err:=ds_GetObjectComment(kds_ObjectComment ;methodID;0;$comment;$styl)
EFFACER PRESSE PAPIERS
ECRIRE TEXTE DANS PRESSE PAPIERS($comment)
AJOUTER A PRESSE PAPIERS("styl";$styl)
` le texte peut maintenant être collé dans 4D Write, y compris sous Windows.

```

ds_SetObjectComments

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

| | | | |
|-----------------|-------------|---|----------------------------------|
| commentSelector | Entier | → | Selecteur |
| Id | Entier | → | Id de l'objet |
| fieldID | Entier | → | Id du champ |
| newComment | Texte | → | Commentaire |
| newStyle | BLOB | → | Style du commentaire |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Ajoute un commentaire à un objet. Passer une chaîne vide dans newComment supprime le commentaire. Voir ds_GetObjectComments pour les valeurs possibles de selector.

Ajout version 1.1 : blobStyle

Le BLOB blobStyle, s'il est passé, doit contenir une description du style du texte au format 'styl' macintosh. Ce format est accessible (y compris sous Windows) avec l'aide de 4D Write, via le presse-papiers. En pratique, on peut écrire un commentaire dans 4D Write, le sélectionner et le copier, puis lire le presse-papiers pour modifier le commentaire en conservant le style. Notez que si newStyle n'est pas passé, le commentaire n'aura aucun style du tout, le précédent – s'il y en avait un – sera perdu.

Exemple après copie dans 4D Write :

```

C_BLOB($styl)
FIXER TAILLE BLOB($newStyle;0)
C_TEXTE($newStyle)
$newComment:=Lire texte dans presse papiers
LIRE PRESSE PAPIERS("styl";$newStyle)
$L_err:=ds_SetObjectComment(kds_ObjectComment ;methodID;0;$newComment;$newStyle)

```

```
ds_GetTipList(tipNames;tipIDs) ← errorCode
```

/

| | | | |
|-----------|-------------------|---|-------------------------------------|
| TipNames | Tableau Texte | ← | Tableau des noms des bulles d'aides |
| TipIDs | Tableau numérique | ← | Tableau des IDs des bulles d'aides |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Retourne la liste des bulles d'aides ainsi que leurs Ids.

ds_SetTip(tipID;newName;newValue) ← errorCode

| | | | |
|-----------|-------------|---|----------------------------------|
| TipID | Entier | ← | Id de la bulle d'aide |
| NewName | Alpha | → | Nouveau nom |
| NewValue | Alpha | → | Nouveau libellé |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Modifie une bulle d'aide.

ds_GetObjectModifDate(selector;ID;dateModif;heureModif) ← errorCode

| | | | |
|------------|-------------|---|----------------------------------|
| selector | Entier | → | type d'objet |
| ID | Numérique | → | ID de l'objet |
| dateModif | Date | ← | Date de modification |
| heureModif | Heure | ← | Heure de modification |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Renvoie la date et l'heure de dernière modification de l'objet. Les valeurs possibles pour selector sont :

- 1 : méthode
- 2 : Table
- 3 : Formulaire
- 4 : barre de menus
- 5 : Menu
- 6 : Bulle d'aide

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.

TABLEAU TEXTE(names;0)
TABLEAU BOOLEEN(visible;0)

```

TABLEAU ENTIER(IDs;0)
C_DATE($modifDate;$date)
C_HEURE($modifTime)
` Chargement de toutes les méthodes projet
` Get all project methods
$err:=ds_GetMethodNames (names;"";visible;IDs)
$date:=Date du jour-7
` Pour chaque méthode...
` For each method...
Boucle ($i;Taille tableau(IDs);1;-1)
` ...lire la date de modification...
` ...get the modification date...
$err:=ds_GetObjectModifDate (1;IDs{i};$modifDate;$modifTime)
` ...si elle est < date voulue, on retire la méthode de la liste.
` ...if it is < date checked, remove the method from the list.
Si ($modifDate<$date)
SUPPRIMER LIGNES(names;$i)
SUPPRIMER LIGNES(visible;$i)
SUPPRIMER LIGNES(IDs;$i)
Fin de si
Fin de boucle

```

ds_GetPluginNames(pluginNames) <- errorCode

| | | | |
|-------------|---------------------|---|----------------------------------|
| pluginNames | Tableau alpha/texte | ← | Noms des plug-ins |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Renvoie la liste des noms de tous les plug-ins actifs, quelle que soit leur localisation (un plug-in peut être situé dans un dossier Mac/Win4DX lui-même is à divers endroits, se reporter à la documentaiton de 4D).

Notez que la routine pourra charger les « faux plug-ins », et que ceux-ci n'ont parfois pas de nom.

ds_GetPluginRoutines(pluginName ;pluginRoutines{;selector}) <- errorCode

| | | | |
|----------------|---------------------|---|----------------------------------|
| pluginName | Alpha/texte | → | Nom du plug-in |
| pluginRoutines | Tableau alpha/texte | ← | Noms des routines de ce plug-in |
| selector | numérique | → | Format des chaînes. |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Renvoie la liste des routines du plug-in nommé pluginName. Selon la valeur du paramètre selector, les noms pourront être plus ou moins complets :

0 : noms bruts (avec paramètres).

Exemple : "ds_GetMethodNames(&Y;&S;&Y;&Y):L"

1 : noms bruts précédés du thème avec un slash.

/

Exemple : "Méthodes/ds_GetMethodNames(&Y;&S;&Y;&Y):L"

2 : noms sans les paramètres

Exemple : "ds_GetMethodNames"

4 : noms sans les paramètres, précédés du thème avec un slash

Exemple : "Méthodes/ds_GetMethodNames"

Si le plug-in est un « faux plug-in » et qu'il ne contient aucune ressource « STR# » décrivant des routines, la ds_GetPluginRoutines renvoie l'erreur -192 (« Ressource introuvable »).

ds_Preferences(selector ;value) <- errorCode

| | | | |
|--------------|-------------|---|----------------------------------|
| prefSelector | Entier | → | Valeur à lire/écrire |
| prefvalue | Entier long | → | Nouvelle valeur |
| | | ← | Valeur actuelle |
| errorCode | Entier Long | ← | Code d'erreur (0 = pas d'erreur) |

Cette routine permet de lire/écrire des préférences influant sur le comportement de DynamicStructure. Apparue en version 1.1, elle possède deux sélecteurs possibles :

- 1 Modifier la valeur de fréquence d'appel au Scheduler
- 1 Lire la valeur de fréquence d'appel au Scheduler

Ce sélecteur change le comportement des routines ds_GetMethodNames et ds_GetObjectMethodIDs. Sur des machines lentes et/ou si le nombre d'objets à charger est important, le temps de chargement peut être relativement long, de l'ordre de plusieurs secondes. Il est important que le plugin puisse donner la main à 4D de temps en temps, exactement comme on le fait lorsque l'on utilise APPELER 4D. Si on fixe la valeur de ce sélecteur à 0 (ce qui signifie « ne jamais appeler le scheduler de 4D ») et que le temps de chargement est long, l'utilisateur pourra avoir le sentiment que sa machine est gelée, les autres process n'auront pas la main, etc. avant que le plugin ait terminé. La valeur par défaut est 30. Au delà de 0, plus la valeur donnée sera élevée, moins 4D aura la main.

DynamicStructure : Codes d'erreur

| Valeur | Signification | Constante |
|--------|---|-------------------------------|
| 0 | Pas d'erreur | kdsERR_NoError |
| -5 | Ressource (méthode, commentaire, ...) non trouvée | kdsERR_ResNotFound |
| -15000 | Tableau d'un type invalide | kdsERR_BadArrayType |
| -30000 | Non enregistré | kdsERR_NotRegistered |
| -30001 | Temps Dépassé | kdsERR_TimeBombed |
| -30002 | Mauvaise Version De 4D | kdsERR_Bad4DVersion |
| -30003 | Au Moins 4D 67 | kdsERR_NeedAtLeast67 |
| -30004 | Erreur Chargement Structure | kdsERR_CouldNotLoadStruct |
| -30005 | Pas Sur Windows | kdsERR_NotOnWindows |
| -30006 | Pas Sur Mac | kdsERR_NotOnMac |
| -30007 | Pas Sur 4D Client | kdsERR_NotOn4DClient |
| -30008 | Pas En Compilé | kdsERR_NotOnCompiled |
| -30009 | Selecteur Invalide | kdsERR_BadSelector |
| -30010 | Le blob reçu est invalide | kdsERR_InvalidBlob |
| -30011 | La routine ne marche qu'avec 4D monoposte | kdsERR_Only4DMono |
| -30100 | Paramètre Invalide (erreur interne au plugin) | kdsERR_OutOfRangeParameter |
| -30101 | Mauvais Passage de Paramètre | kdsERR_ParamErr |
| -30102 | 32000 caractères dans le texte, impossible d'en mettre plus | kdsERR_32000CharReached |
| -30200 | Objet Verrouillé | kdsERR_4DResLocked |
| -30201 | Nom Invalide | kdsERR_InvalidName |
| -30300 | ID De Feuille De Style Invalide | kdsERR_BadStyleSheetID |
| -30301 | Nom Méthode Existe | kdsERR_MethodNameExists |
| -30302 | Nom Méthode Non Trouvé | kdsERR_MethodNameNotFound |
| -30303 | Pas En Mode Graphe | kdsERR_NoFlowChart |
| -30304 | Longueur De Champ Alpha Invalide | kdsERR_InvalidStringFieldSize |
| -30305 | Type De Champ Invalide | kdsERR_InvalidFieldKind |
| -30306 | GetVariable : Seulement Compilé | kdsERR_GetVarOnlyCompiled |
| -30307 | Formulaire Introuvable | kdsERR_FormNotFound |
| -30308 | Erreur Lecture Formulaire | kdsERR_ErrorReadingForm |
| -30309 | Taquets Invalides | kdsERR_BadRulersParams |
| -30310 | Formulaire Existe Déjà | kdsERR_DuplicateFormName |
| -30311 | Version De Formulaire Inconnue | kdsERR_UnknnownFO4DVers |
| -30312 | Erreur Lors De L' Analyse Du Formulaire | kdsERR_FO4DParserError |
| -30313 | ID Déjà Utilisée | kdsERR_IDAlreadyUsed |
| -30314 | Erreur lecture formulaire (« NullFromHandle ») | kdsERR_NullFormHandle |
| -30350 | Nombre maximal de champs atteint | kdsERR_MaxFieldCount |
| -30351 | Nom de table déjà existant | kdsERR_DuplicateTableName |
| -30352 | Nombre maximal de tables atteint | kdsERR_MaxTableCount |
| -30400 | Objet de Composant non Modifiable (protégé) | kdsERR_CantChangeCompObject |

/

| | | |
|--------|---|--------------------------|
| -30401 | Objet Introuvable | kdsERR_ObjectNotFound |
| -30402 | Objet de Composant non Chargeable (protégé) | kdsERR_CantGetCompObject |
| -30403 | ID de composant invalide | kdsERR_BadComponentID |
| -30500 | Plugin introuvable (mauvais nom) | kdsERR_PluginNotFound |