

# SILVERDEV

## TECHNIQUES AVANCEES

*P r o g r a m m a t i o n   R P G   I L E*

**Notice relative aux droits d'auteurs.**

Les informations contenues dans ce document pourront faire l'objet de modifications sans préavis et ne sauraient en aucune manière engager **EXPERIA**. La fourniture du progiciel est régie par un octroi de licence ou un accord de confidentialité. Le progiciel ne peut être utilisé, copié ou reproduit sur quelque support que ce soit que conformément aux termes de cette licence ou de cet accord de confidentialité. L'acheteur ne peut effectuer des copies que dans le but de sauvegarde ou d'archivage.

Aucune partie du manuel et du progiciel ne peut être reproduite ou transmise par quelque moyen que ce soit, électronique ou mécanique, y compris par photocopie, enregistrement ou tout autre procédé de stockage, de traitement et de récupération d'informations, pour d'autres buts que l'usage personnel de l'acheteur sans permission expresse et écrite de la société **EXPERIA**.

IBM, AS/400, iSeries, System i, i5, Power I sont des marques déposées de International Business Machines Corporation. Windows est une marque déposée de Microsoft.

Tous les autres produits sont des marques déposées de leur société respective.

EXPERIA Europe  
4, rue L.Beridot  
Les jardins d'Epione  
38500 VOIRON  
FRANCE

## Table des matières

<b>Chapitre 1. Optimisation.....</b>	<b>6</b>	Introduction .....	36
Fonctions avec réponse .....	6	Exemple : .....	36
Cache des variables .....	6	Programme MDIFORM .....	37
Vidage du cache .....	8	Programme Child1 .....	39
Evénements déclenchés par code .....	8	Programme Child2.....	39
Cache des fenêtres .....	8	<b>Chapitre 8. Application multifenêtres</b>	<b>41</b>
Fonctions de type List .....	9	avec des onglets .....	<b>41</b>
Fonction sdGetSfl .....	9	Introduction .....	41
Code MD5 des icônes .....	9	Placer un composant dans un conteneur .....	41
Réutilisation de fenêtres .....	10	Sources .....	42
Compression.....	11	Programme PGM1 .....	42
Evenements locaux.....	11	Programme PGM2.....	43
sdAddNode .....	11	<b>Chapitre 9. MultiOccurrences d'une</b>	<b>45</b>
<b>Chapitre 2. Fonctions de la catégorie List</b>	<b>12</b>	fenêtre.....	<b>45</b>
Introduction .....	12	Introduction .....	45
Exemple : .....	12	Variable de fenêtre .....	45
sdGetControls, sdGetComponents .....	13	Evènements .....	45
<b>Chapitre 3. SFL : Recopie des données</b>	<b>14</b>	Stockage des données .....	46
en locales .....	<b>14</b>	<b>Chapitre 10. Multioccurrences ,</b>	<b>47</b>
sdGetSfl.....	14	programme de service	
sdGetSflRow.....	15	sdsrvlst .....	<b>47</b>
sdGetSflCol.....	15	Introduction .....	47
<b>Chapitre 4. Création dynamique de</b>	<b>17</b>	Service programme sdsrvlst .....	47
composants.....	<b>17</b>	Service programme srvchld .....	50
<b>Chapitre 5. Evenements locaux.....</b>	<b>19</b>	Programme Child .....	52
Introduction .....	19	Prototypes .....	55
Langage .....	19	Membre d'exportation .....	55
Structures et mots clefs .....	19	Répertoire de liage .....	55
Editeur .....	21	<b>Chapitre 11. Exemples et astuces .....</b>	<b>56</b>
Interactions avec evenements serveurs.....	22	« A partir de » dans sous fichiers .....	56
Fenêtres.....	23	Sous fichiers et F11.....	56
Ajout dynamique d'évènements locaux .....	23	Fenêtre d'attente .....	58
Exemples .....	23	Introduction : .....	58
<b>Chapitre 6. Applications Multifenêtres.....</b>	<b>30</b>	Fenêtre .....	59
Introduction .....	30	Code : .....	59
Organisation des fenêtres .....	30	Programme appelant : .....	60
Exemple de fenêtres flottantes : .....	30	Activer/désactiver éléments de menu	
Pile d'appels .....	31	OnPopup .....	61
Interactions entre programmes .....	31	Fenêtres de choix.....	61
Cas 1 : Sur une action dans ecran1, on souhaite		Contrôle de la compilation .....	63
effectuer une modification sur ecran2.....	32	Raccourcis claviers .....	63
Cas 2 : Sur une action dans ecran2, on souhaite		Modification d'une colonne de sous fichier .....	64
effectuer une modification sur ecran1 .....	33	Ajustement de la taille des composants .....	64
Paramètres .....	34	Align.....	64
<b>Chapitre 7. Applications MDI.....</b>	<b>36</b>	Anchors.....	65
		Profil utilisateur en cours .....	65
		Récupération du profil en cours dans les programmes...	66
		RPG.....	66

CLP .....	67
Déclencher un évènement par programme. ....	67
Envoyer un message entre fenêtres .....	67
Ouvrir un document sur le serveur .....	68
CSfl dynamique:.....	68
Survol d'un sfl .....	68
Eviter un appel à sdGet .....	69

## Chapitre 12. Applications multilingues ..... 70

Propriété Langs .....	70
Traduction des chaînes.....	70
Fichier PSVDDLO .....	72
Modifier la langue à l'exécution. ....	73
Déplacement d'un écran .....	73
Menu traduction .....	74
SdRtvMsg .....	74

## Chapitre 13. Fonctions catégorie images .. 76

sdSetImg.....	76
sdGetImg .....	78
sdLoadFromFile .....	79
sdAssignTo .....	80
sdAddImg.....	81

## Chapitre 14. Fonctions boîtes de dialogue 82

SdDialog .....	82
sdShowMessage.....	82
sdMsgDlg .....	83
Première solution : .....	83
Deuxième solution : .....	83
sdSelectFile .....	85
sdSelectDir .....	87
sdPrintDlg .....	87
sdSelectColor .....	87

## Chapitre 15. Fonctions catégorie fichiers PC ..... 89

sdGetDir.....	89
SdDelFile .....	90
sdDownLoad .....	90
SdForceDir.....	91
SdGetFileSize .....	92
SdSaveToFile .....	92
SdSelectFile.....	93
SdSetXFerBar.....	93
SdUpload .....	93
SdSelectDir .....	94
SdFileExists .....	94
SdDirExist .....	95
sdCopyFile .....	95

SdGetRealFolder.....	95
SdRenameFile.....	97

## Chapitre 16. Fonctions catégorie collections ..... 99

sdCollecClear .....	99
sdCollecAdd .....	99
sdCollecDelete .....	99
sdCollecIns.....	99

## Chapitre 17. Fonctions catégorie TStrings101

Introduction .....	101
Fonctions avec réponse .....	101
Cache des variables .....	102
Vidage du cache.....	103
Evénements déclenchés par code.....	104
Cache des fenêtres .....	104
Fonctions de type List .....	104
Fonction sdGetSfl.....	104
Code MD5 des icônes .....	105
Réutilisation de fenêtres.....	105
Compression .....	106
Evenements locaux .....	106
sdAddNode .....	106
sdSIAdd.....	107
sdSIClear.....	107
sdSIDelete.....	107
sdSIGet .....	108
sdSIInsert .....	108

## Chapitre 18. Fonctions catégorie ensembles ..... 109

sdSetSet.....	109
sdAddSet.....	110
sdDelSet.....	111
SdGetSet.....	111
SdUpdSet.....	112
SdIsInSet.....	112

## Chapitre 19. Fonctions diverses ..... 115

SdHideMainForm .....	115
SdCursor .....	115
SdApplySet .....	116
sdEnd .....	116
sdInnerEnabled .....	116
sdInnerReadOnly .....	117
sdGetClientIp .....	117
SdGetEnvVar .....	117
sdMsgDebug .....	118
sdSetKeepAlive .....	118

SdBringApplicationToFront .....	119	Conteneurs imbriqués .....	143
SdFlashApplication .....	119	Composant Splitter .....	144
<b>Chapitre 20. Liste des fenêtres et des événements .....</b>	<b>121</b>	Gridpanel, flowPanel, relativePanel .....	145
sdGetFormCount .....	121	Composant CScreen .....	145
sdGetFormHandle.....	121	Evènement onResize. ....	146
sdGetFormName .....	121	<b>Chapitre 25. ADO.....</b>	<b>147</b>
sdGetFormLib .....	122	Introduction .....	147
sdGetFormObj .....	122	CADOConnection .....	147
sdGetEventCount.....	123	CAdoQuery .....	148
sdGetEventName.....	124	<b>Chapitre 26. Exit points.....</b>	<b>150</b>
sdGetEventProc.....	124	SVDCTRLCPL.....	150
<b>Chapitre 21. Couleurs .....</b>	<b>126</b>	SVDEXCEPT.....	150
Designer .....	126	SVDSTRAPP.....	151
Modification à l'exécution .....	126	SVDSTRPGM.....	151
Choix de couleur.....	127	<b>Chapitre 27. Interaction avec des programmes externes. ....</b>	<b>153</b>
Couleurs prédéfinies .....	127	Lancement d'une application Silverdev depuis un programme sur pc. ....	153
<b>Chapitre 22. Ancrer des composants.....</b>	<b>131</b>	Lancement d'une application silverdev depuis un programme 5250.....	154
Introduction .....	131	Lancement d'une application Silverdev depuis une page html .....	157
Exemple de base .....	131	Lancement d'une application Silverdev depuis une application Silverdev .....	157
Fenêtre flottante.....	132	Lancement d'un programme depuis silverdev ..	158
Amélioration .....	132	Lancement d'une page web depuis une application Silverdev .....	160
<b>Chapitre 23. Opérations de drag and drop.134</b>		Insertion d'une application 5250 dans un programme silverdev.....	160
Introduction .....	134	Insertion d'une page web dans un programme silverdev.....	161
DragMode .....	134	<b>Chapitre 28. Programme de service sdsrvjson.....</b>	<b>162</b>
OnDragOver .....	135	Introduction : .....	162
OnDragDrop .....	135	Fonctions.....	162
Exemples .....	136	<b>Chapitre 29. WebBrowser .....</b>	<b>163</b>
Drag and drop d'un ClistBox vers un ClistBox .....	136	<b>Chapitre 30. Web services .....</b>	<b>164</b>
Drag and drop d'un ClistView vers un ClistView .....	137		
DragCursor et CCustomCursor .....	138		
OnDragOverEx, OnDragDropEx .....	139		
Afficher un popupmenu .....	140		
<b>Chapitre 24. Adaptation taille écran.....</b>	<b>142</b>		
Introduction .....	142		
Taille modifiable par l'utilisateur .....	142		
Align, anchors .....	142		
Anchors.....	143		

---

# Chapitre 1. Optimisation

Dans toutes les applications client-serveur, les temps de réponse, c'est le nerf de la guerre.

Ce chapitre vous présente les moyens utilisés par Silverdev pour optimiser les temps de réponse et vous donne quelques conseils pour améliorer vos applications.

---

## Fonctions avec réponse

L'appel aux fonctions demandant une réponse sont plus lentes car elles nécessitent un aller-retour sur le réseau.

Les fonctions ne renvoyant pas de valeur sont groupées dans un buffer.

C'est le cas par exemple des fonction `sdSet`, `sdSetCell` etc..

Ce buffer est envoyé au client à la fin du gestionnaire d'événement ou lors de l'appel d'une fonction qui renvoie une valeur (par exemple `sdGet`) ou encore à l'appel de la fonction `sdApplySet`.

A titre d'exemple :

Sur une machine de test, nous avons mesuré :

1000 appels à `sdGet` : Environ 3 secondes.

1000 appels à `sdset` Environ 15 millisecondes

---

## Cache des variables

Afin d'améliorer les performances des fonctions de type `sdGet`, à chaque événement, certaines valeurs de propriétés sont envoyées au serveur.

Ces valeurs sont mises en cache.

Les fonctions de type `sdGet` commencent par chercher dans ce cache.

Ce mécanisme est transparent pour le développeur, mais il permet d'améliorer nettement les performances.

Seules les propriétés qui ont la plus grande probabilité d'être interrogées sont mises en cache.

Par exemple, seule la propriété `text` du composant `CEdit` est mise en cache.

Dans le debugger :

Envoyer événement

Evenement : \_2.BtnOK.OnClick

-----

13/10/2007 15:01:59:218

-----

Envoi du cache

\_2.TITRE.text=Les robots

\_2.PRIX.value=4

\_2.STOCK.value=0

\_2.ACOMMANDER.value=Y

\_2.Themes.keySelected=2

\_2.Themes.text=Fantastique

\_2.IDAUT.value=17

\_2.NOMAUT.text=Dostoïevski

\_2.NOMEDI.text=j'ai Lu

\_2.IDEDI.value=4

\_2.DATEPARU.value=20040513

\_2.DATEPARU.DateIso=2004-05-13

-----

13/10/2007 15:01:59:234

-----

Execution Get

\_2.DATEPARU.ValidDate

Résultat : True

-----

13/10/2007 15:01:59:343

Execution Script

function Main() {

\_2.ModalResult=1;

}

-----

C	Eval	TITRE= sdGet (F1: 'TITRE': 'Text')
C	if	sdGetBool (F1: 'DATEPARU': 'ValidDate')=*off
C	eval	Message =Message + 'La date est -
C		invalide' + X'0D'
C	endif	
C	Eval	PRIX=sdGetNum (F1: 'PRIX': 'Value')
C	Eval	STOCK=sdGetInt (F1: 'STOCK': 'Value')
C	Eval	ACOMMANDER=sdGet (F1: 'ACOMMANDER':
C		'Value')
C	Eval	IDTHEME=sdGetNum (F1: 'Themes':

C		'KEYSELECTED')
C	Eval	IDAUT=sdGetInt(F1: 'IDAUT': 'Value')
C	Eval	IDEDI=sdGetInt(F1: 'IDEDI': 'Value')
C	eval	DATEPARU=sdGetDate(F1: 'DATEPARU')
C	callp	sdSetInt(F1: '*FORM': 'ModalResult': Mrok)

## Vidage du cache

Lorsque vous appelez une fonction qui n'effectue pas un get, le cache est vidé. Il faut donc veiller autant que possible à appeler les fonctions de types get en premier dans le gestionnaire d'évènement.

Exemple :

Le code suivant

```
*/EVENT Button4_OnClick
/free
row = sdGetInt(f1:'sfl1':'rowSelected');
sdSetBool(f1:'button1':'enabled':*off);
```

est plus efficace que le code suivant :

```
*/EVENT Button4_OnClick
/free
sdSetBool(f1:'button1':'enabled':*off);
row = sdGetInt(f1:'sfl1':'rowSelected');
```

## Evénements déclenchés par code

Lorsqu'un évènement est déclenché par code, le cache n'est pas envoyé par la partie cliente. (Dans le cas contraire, le cache pourrait ne pas être valide)

---

## Cache des fenêtres

Les fenêtres silverdev doivent être transmises du serveur vers le client.

Si celles ci contiennent des images, le transfert peut être assez long.

Les fenêtres créées sont donc enregistrées sur le disque du client.



Lors de la création d'une fenêtre, le client et le serveur commencent par discuter pour savoir si la fenêtre existe déjà sur le poste du client.

Les fenêtres sont stockées dans le répertoire /cache

Le nom du fichier cache est le code md5 de la fenêtre.

Lorsqu'une fenêtre est modifiée, et donc remise en cache, l'ancien cache de cette fenêtre est supprimé. La correspondance entre la provenance d'une fenêtre et son fichier cache est sauvegardée dans la base de registres.

---

## Fonctions de type List

Comme nous l'avons vu précédemment, les fonctions de type sdGet prennent un temps non négligeable.

La relecture de tous les éléments d'un composant de type treeview, memo, listbox, checklistbox...peut donc être assez long.

Les fonctions de la catégorie List permettent de gagner une temps considérable.

Le principe est de remonter toutes les informations intéressantes d'un composant, de les copier en mémoire afin de pouvoir les interroger en local. Pour plus de détail, voir (0)

---

## Fonction sdGetSfl

Afin de lire les valeurs dans un composant CSfl, utilisez la fonction sdGetSfl et les fonctions associées (sdGetSflStr,sdGetSflModified..)

Les données sont recopiées en mémoire sur le serveur.La lecture des données est ensuite très rapide.

---

## Code MD5 des icônes

Lors de l'affichage d'un répertoire dans MYDESK, différentes données sont envoyées à MyDesk.

Les informations concernant l'application et éventuellement une icône.

Les informations sont de type texte et ne représentent que quelques octets.

Les icônes sont beaucoup plus lourdes et représentent 90% des données à transférer.

Un mécanisme basé sur un code de hashage md5 est utilisé afin de n'envoyer les icônes qu'une seule fois.

Pour chaque application il existe sur le serveur un fichier à l'extension .app et éventuellement un fichier à l'extension .ico.

Le code MD5 de l'icône est stocké dans le fichier .app.

Ce fichier est envoyé en premier.

MyDesk teste l'existence de l'icône sur le disque.

Si l'icône existe déjà, elle est utilisée sinon, elle est téléchargée puis sauvée sur le disque avec comme nom, le code de hashage de l'icône.

Les icônes sont stockées dans le répertoire /icônes

Remarque :

*Ces icônes sont aussi utilisées lors de la création de raccourcis sur le bureau.*

## Réutilisation de fenêtres

Lorsque vous créez une fenêtre, des ressources sont utilisées côté client pour créer les composants de cette fenêtre.

Lorsque l'utilisateur ferme une fenêtre, celle ci est cachée, mais elle n'est pas détruite.

C'est à dire que les composants de la fenêtre et la fenêtre elle même utilisent de la mémoire côté PC.

La fenêtre sera détruite au moment de la fermeture du programme (côté pc)

Si vous souhaitez libérer la fenêtre au moment de sa fermeture, ajoutez le code suivant dans l'événement onclose de la fenêtre :

```
p ONCLOSE...
p
d B
d PI
D Parameters ds based(pevtinf)
D Win 5u 0
D Evt 48a
,* Variable
D Action 10i 0
,* 0 : ne rien faire
,* 1 : Cacher la fenêtre
,* 2 : Libérer la fenêtre
,* 3 : Minimiser la fenêtre
,*
c eval action = 2
c callp sdFreeForm(Win)
```

Que la fenêtre soit détruite ou non, n'oubliez pas de tester qu'une fenêtre n'existe pas déjà avant de la créer :

```
C      If      F1 <> 0
C      eval    F1 = sdcreateForm(F1REF)
C      endif
```

Si elle existe déjà, vous allez créer plusieurs fenêtres identiques qui utiliseront des ressources inutilement.

---

## Compression

Lors des premiers échanges de données, le temps des échanges est mesuré. En fonction du temps calculé, le serveur détermine si le client est distant ou sur le réseau local. Si le client est distant, les données sont compressées avant d'être transmises.

---

## Evenements locaux

Il est possible d'effectuer tous les traitements dans des evenements serveurs, mais le traitement des evenenemts en local est plus rapide que le traitement sur le serveur. Lorsqu'un événement se produit très souvent et que le traitement ne necessite pas l'utilisation d'objets sur le serveur, préférez les événements locaux.

---

## sdAddNode

La fonction sdAddNode2 est à préférer à la fonction sdAddNode car elle est beaucoup plus rapide et plus simple d'utilisation.

## Chapitre 2. Fonctions de la catégorie List

### Introduction

Certains composants possèdent une liste de valeurs, et il peut être long de parcourir l'ensemble de ces valeurs par les méthodes classiques. En effet, l'interrogation de la valeur d'une propriété prend un temps de l'ordre de 10 ms.

La fonction `sdGetList` permet d'interroger ces composants et de stocker les données sur le serveur. Cette fonction renvoie un pointeur qu'il faut ensuite utiliser dans les fonctions `sdGetListLabel`, `sdGetListKey`, `sdGetListSelected`, `sdGetListState` et `sdGetListCount`.

Les fonctions de type List permettent d'interroger des composants tels que `CCheckBox`, `CListBox`, `CComboBox`, `CTreeView`, `CListView` ou `CMemo`.

Il est ainsi beaucoup plus rapide de parcourir l'ensemble des valeurs de ces composants qu'avec les fonctions classiques `sdGet` et `sdSGet`.

### Exemple :

```
DList      s      *
DLabel     s      256    Varying
DKey       s      256    Varying
DSelected  s      N
DState     s      5u 0
Di         s      10u 0 inz(0)
C          eval    List=sdGetList(F1:'CheckBox1')
C          if      List <> *NULL
C          DOW     I < sdgetListCount(List)
C          eval    Label=sdGetListLabel(List:i)
C          eval    Key=sdGetListKey(List:i)
C          eval    Selected=sdGetListSelected(List:i)
C          eval    State=sdGetListState(List:i)
C          enddo
C          callp   sdFreeList(List)
C          endif
```

Utilisez la fonction `sdFreeList` pour libérer la mémoire allouée par la fonction `sdGetList`.

## sdGetControls, sdGetComponents

Les fonctions sdGetControls et sdGetComponents renvoient également un pointeur vers un objet du même type que celui renvoyé par sdGetList.

La fonction sdGetControls renvoie la liste de tous les contrôles contenus dans celui passé en paramètre.

La fonction sdGetComponents renvoie la liste de tous les composants dont le composant en paramètre est responsable de la destruction.

Pour une fenêtre créée avec le designer, tous les composants sur la fenêtre font partie de cette liste.

Le pointeur peut donc être utilisé comme paramètre dans les fonctions sdGetListLabel et sdGetListKey.

sdGetListLabel renvoie le nom du composant et sdGetListKey renvoie le type du composant.

### Exemple :

```
DList      s      *
DControlName  s      256      Varying
DTypeControl  s      256      Varying
Di          s      10u 0 inz(0)
C           eval    List=sdGetControls(F1: '*FORM')
C           if      List <> *NULL
C           DOW      I < sdgetListCount(list)
C           eval    ControlName=sdGetListLabel(List:i)
C           eval    TypeControl=sdGetListKey(List:i)
C           eval    i=i+1
C           enddo
C           callp    sdFreeList(List)
C           endif
```

## Chapitre 3. SFL : Recopie des données en locales

### sdGetSfl

```
D sdGetSfl      pr      *
D F1           5u 0 const
D component    30      varying value
D Filter       1  0 value options(*nopass)
```

Les fonctions sdGetCell, sdGetCellNum, etc.. permettent d'interroger le contenu des cellules dans un composant CSfl.

L'interrogation de toutes les cellules d'un sfl comprenant de nombreuses colonnes et de nombreuses lignes peut alors être assez long.

Afin d'optimiser les temps de réponses, utilisez la fonctions sdGetSfl.

Cette fonction recopie en local les données du composant CSfl.

Il est ensuite possible d'accéder aux données recopiées à l'aide des fonctions sdGetSflStr, sdGetSflNum, sdGetSflDate, sdGetSflTime, sdGetSflNull, sdGetSflModified, sdGetSflLines, sdGetSflColumns, sdGetSflColName.

Le paramètre Filter permet de choisir les lignes sélectionnées.

Les valeurs possibles de Filter sont :

Valeur	Constante définie dans H,SILVERDEV	Description
0	ALL_ROWS	Toutes les lignes
1	DIRTY_ROWS	Seulement les lignes modifiées
2	SELECTED_ROWS	Seulement les lignes sélectionnées.

La fonction sdGetSfl alloue de la mémoire. Pour libérer cette mémoire, utilisez la fonctions sdFreeSfl.

### Exemple :

```
*/EVENT Button2_OnClick
Dsfl      s      *
D i        s      10i 0
D MyDate   s      d
D MyNum    s      10  0
```

```

D MyStr          s          100    Varying
C                eval      sfl=sdgetsfl(f1:'sfl1':DIRTY_ROWS)
C                if        Sfl <> *NULL
C                for        i=1 to sdGetSflLines(Sfl)
C                if        not sdGetSflNull(Sfl:'MyDate':i)
C                eval      MyDate=sdgetSflDate(sfl:'MyDate':i)
C                endif
C                eval      MyNum=sdgetSflNum(sfl:'MyNum':i)
C                eval      MyStr=sdgetSflStr(sfl:'MyStr':i)
C                endfor
C                callp      sdFreeSfl(Sfl)

```

### sdGetSflRow

La fonction sdGetSfl remonte toutes les données de la machine cliente vers le serveur. Si vous n'avez besoin des données que d'une seule ligne, vous pouvez utiliser la fonction sdGetSflRow à la place de la fonction sdGetSfl.

La fonction sdGetSflRow renvoie un pointeur que vous pouvez utiliser dans les fonctions sdGetSflStr, sdGetSflNum, etc...

Il faut alors que le dernier paramètre de ces fonctions soit à 1.

### Exemple:

```

*/EVENT Button2_OnClick
Dsfl          s          *
D Row         s          10i 0
D MyDate      s          d
D MyNum       s          10 0
D MyStr       s          100    Varying
C            eval      Row = sdGetInt(F1:'sfl1':'RowSelected')
C            if        Row > 0
C            eval      sfl=sdgetsflRow(f1:'sfl1':Row)
C            if        Sfl <> *NULL
C            eval      MyNum=sdgetSflNum(sfl:'MyNum':1)
C            eval      MyStr=sdgetSflStr(sfl:'MyStr':1)
C            endif
C            endif
C            callp      sdFreeSfl(Sfl)

```

### sdGetSflCol

La fonction sdGetSflCol permet de récupérer toutes les valeurs d'une colonne d'un composant CSFL.

Les valeurs de la ligne sont recopiées dans une structure en local.  
Cette structure est identique à celle obtenu par la fonction sdGetSfl

**Exemple:**

```
Dsfl          s          *
D Row         s          10i 0
D MyDate      s          d
D MyNum       s          10 0
D MyStr       s          100 Varying
C             eval       sfl=sdgetsflCol (f1:'sfl1':'MyColumn')
C             if         Sfl <> *NULL

C             for        i=1 to sdGetSflLines(Sfl)
C             eval       MyNum=sdgetSflNum(sfl:'MyNum':i)
C             eval       MyStr=sdgetSflStr(sfl:'MyStr':i)
C             endfor
C             callp      sdFreeSfl(Sfl)
C             endif
```



## Chapitre 4. Création dynamique de composants

La création dynamique de composants se fait avec la fonction `sdCreate`.

Pour la création d'une fiche avec beaucoup de composants, il est possible de gagner en temps de réponse, en utilisant la fonction `sdLoadComponent`.

Cette fonction nécessite de créer un fichier au format XML.

Le schéma de ce fichier est le suivant :

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Control">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Control" type="Control" />
        <xs:element name="Data" type="xs:string" />
        <xs:simpleType>
          </xs:simpleType>
        </xs:element>
        <xs:element name="Event">
          <xs:simpleType>
            <xs:attribute name="Name" type="xs:string"/>
          </xs:simpleType>
        </xs:element>
        <xs:element name="Script">
          <xs:simpleType>
            <xs:attribute name="Language" type="xs:string"/>
            <xs:restriction base="xs:string">
              <xs:enumeration value="DelphiScript"/>
              <xs:enumeration value="JavaScript"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Le noeud racine doit être 'Control'.

Les noeuds 'Control' inclus contiennent les contrôles inclus.

Le noeud 'Data' contient les propriétés du composant.

Les noeuds 'Event' contiennent des événements du composant.

Les noeuds 'Script' contiennent des scripts à exécuter.

Le programme de service SDSRVCMP permet de créer le fichier plus facilement.

## Exemple :

```
path = '/home/user1/' + getQuallJobFileName+'.xml';
fd = openfile(path);
writeInFile(fd:'<Control Type="CPanel">');
writeInFile(fd:'<Data>');
writeInFile(fd:'object Panell1:CPanel' +CRLF);
writeInFile(fd:'caption = ' + encodeString('Panell1') +CRLF);
writeInFile(fd:'end' +CRLF);
writeInFile(fd:'</Data>');
writeInFile(fd:'<Control Type="CButton">');
writeInFile(fd:'<Data>');
writeInFile(fd:'object Button3:CButton' +CRLF);
writeInFile(fd:'caption = ' + encodeString('Button3') +CRLF);
writeInFile(fd:'end' +CRLF);
writeInFile(fd:'</Data>');
writeInFile(fd:'<Event Name="onClick"/>');
writeInFile(fd:'<Script Language ="JavaScript">');
writeInFile(fd:'self.button3.caption = "Hello";');
writeInFile(fd:'</Script>');
writeInFile(fd:'</Control>');
writeInFile(fd:'</Control>');
closeFile(fd);
sdLoadComponent (F1:'*FORM': '*FORM':path);
deleteFile(fd);
```

---

# Chapitre 5. Evenements locaux

---

## Introduction

Un programme silverdev peut être écrit entièrement sans utiliser les evenements locaux. Cette fonctionnalité n'est d'ailleurs apparue que très tard dans le logiciel silverdev.

L'utilisation des événements locaux est une fonctionnalité avancée.

Cependant, cette fonctionnalité apporte une plus grande puissance à Silverdev et permet d'optimiser les temps de réponse.

Il est important de bien comprendre quand utiliser un événement local et quand utiliser un événement serveur.

Lorsque qu'un gestionnaire d'événement ne nécessite pas d'interroger un objet du serveur (principalement la base de données), il est intéressant d'utiliser les événements locaux. Le traitement se fera plus rapidement, puisqu'il n'y aura pas d'échanges entre le serveur et le client.

Certains événements se produisent très souvent, et il n'est pas raisonnable d'utiliser les événements serveur dans ce cas.

---

## Langage

Certains événements possèdent des paramètres modifiables.

Il fallait donc utiliser un langage permettant de passer des paramètres de fonctions par référence. Javascript n'offrant pas cette possibilité, c'est le langage pascal qui a été choisi.

### Structures et mots clefs

#### Déclarations de variables

```
var  
i:integer;  
chaine :string;  
Reel :Float ;
```

**begin**

...

**opérateurs**

Egalité	Différent	Strictement Supérieur	Strictement Inférieur	Supérieur ou égal	Inférieur ou égal
=	<>	>	<	>=	<=

Affectation
:=

Ou logique	Et logique
Or	and

**conditions**

```

if (var1=0) or(var2 <> 5)then
begin
...
end
else
begin
...
end;

```

**Boucles**

```

for i:=0 to 10 do
begin
...
end;

```

```

while(i <10)do
begin
...
end ;

```

```

repeat
...
until (i >2);

```

break : sortir de la boucle

continue : passer à l'itération suivante

### Commentaires :

//commentaire sur une ligne

(\*commentaire sur plusieurs lignes\*)

{commentaires sur plusieurs lignes}

---

## Editeur

Le designer permet de saisir le code de l'événement.

Pour saisir le code de l'événement local, double-cliquez sur la ligne, un éditeur de code apparaît.

Lorsque du code est associé à un événement local, le nom de cet événement apparaît en gras et en bleu.

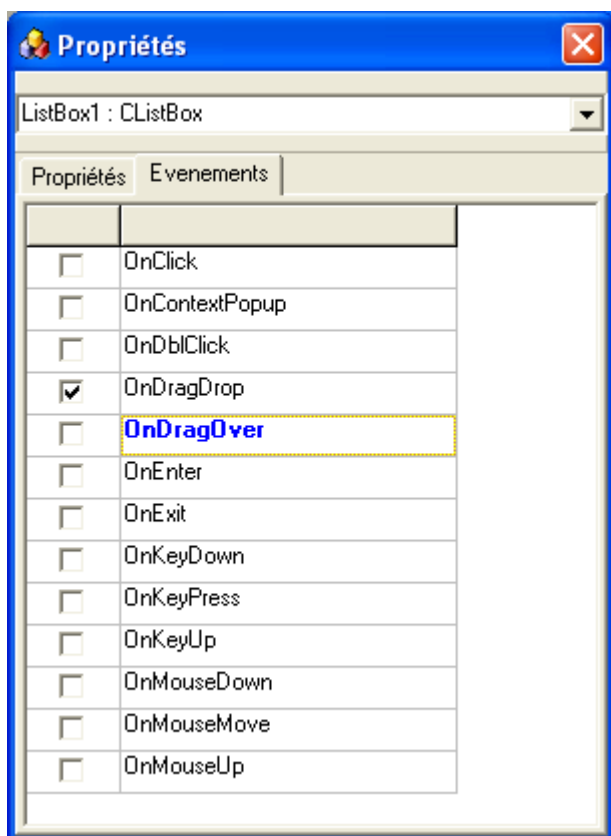


Figure 1

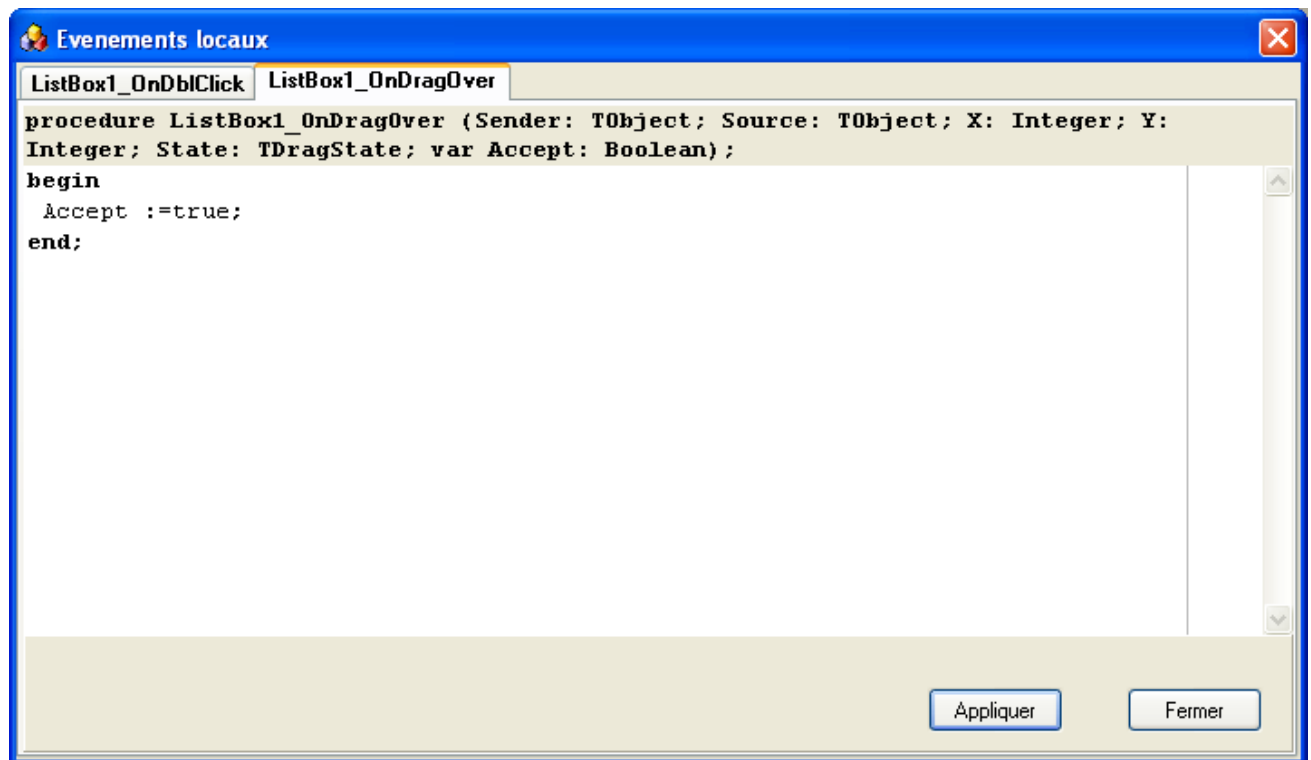


Figure 2

Le prototype de la fonction n'est pas modifiable.

Si le fond de l'éditeur est blanc, cela signifie que vous avez cliqué sur le bouton « Appliquer »  
Si le fond de l'éditeur est jaune, cela signifie que vous avez modifié le code, mais pas encore cliqué sur le bouton « Appliquer ».

Pour enlever un événement local, supprimez tout le code, et cliquez sur Appliquer.

---

## Interactions avec evenements serveurs

Il est possible d'utiliser pour un même événement un traitement en local ou un traitement sur le serveur.

Il est aussi possible de choisir de traiter l'événement à la fois en local et sur le serveur. Dans ce cas, il est possible d'annuler le traitement serveur depuis le traitement local en modifiant la variable booléenne SendEvt.

La variable SendEvt est une variable globale.

Ne déclarez pas de variable locale à la fonction portant ce nom, elle cacherait la variable globale SendEvt.

**Remarque** : Si vous écrivez un gestionnaire d'événement local et un gestionnaire d'événement serveur, affectez toujours une valeur à la variable SendEvt dans le gestionnaire d'événement local.

**Remarque :** Si vous ne demandez pas le signalement d'un événement au serveur (case à cocher dans le designer) l'affectation à true de la variable SendEvt n'aura aucun effet.

---

## Fenêtres

Dans le corps des fonctions, les objets doivent être précédés du nom de la fenêtre contenant l'objet. L'identifiant qui doit être utilisé pour les fenêtres est This Pour les autres objets, l'identifiant utilisé doit être la propriété name.

---

## Ajout dynamique d'évènements locaux

Si vous souhaitez ajouter des évènements à l'exécution pour des composants créés dynamiquement par exemple , utilisez la fonction sdAddLocalEvent.

**Prototype :**

```
d sdAddLocalEvent...
D                               pr
d  Form                        5u 0 const
d  Component                   30    varying value
d  Event                      30    varying value
d  Code                       1000   varying value
```

---

## Exemples

**Exemple 1 :**

**Description :**

L'événement ondragover est utilisé pour déterminer si l'utilisateur peut déplacer un elt en drag and drop

**Code**

```
procedure ListBox1_OnDragOver (Sender: TObject; Source: TObject; X: Integer; Y: Integer; State: TDragState; var Accept: Boolean);
begin
  Accept := (Source = This.listbox2);
end;
```

**Exemple 2 :****Description :**

L'évènement onchange d'un CEdit est envoyé lorsque le contenu dépasse dix caractères (utile pour les saisie avec douchettes à codes barres)

.

**Code**

```
procedure Edit1_OnChange (Sender: TObject);
begin
  SendEvt := (length(This.edit1.text) >= 10);
end;
```

**Exemple 3 :****Description :**

L'évènement onKeyPress est envoyé au serveur uniquement si la touche enter est utilisée

**Code**

```
procedure Edit1_OnKeyPress (Sender: TObject; var Key: Char);
begin
  SendEvt := (Key = #13); // touche enter
end;
```

**Exemple 4 :****Description :**

La zone de saisie passe en couleur



**Code**

```
procedure Edit1_OnEnter (Sender: TObject);
begin
  This.edit1.color := clred;
end;
```

```
procedure Edit1_OnExit (Sender: TObject);
begin
  This.edit1.color := clwhite;
end;
```

**Exemple 5 :****Description :**

Vérification de saisie de zones dans une fiche.

**Code**

```
procedure Button1_OnClick (Sender: TObject);
var
  msg:string;
begin
  msg:='';
  if (This.edit1.text='') then
  begin
    msg:=msg+'edit1 doit être renseigné'+#13#10;
  end;
  if (This.edit2.text='') then
  begin
    msg:=msg+'edit2 doit être renseigné'+#13#10;
  end;
  if (msg='') then
  begin
    SendEvt := true;
  end
  else
  begin
    messageDlg(msg,mtinformation,mbokcancel,0);
  end;
end;
```

**Exemple 6****Description :**

Une case à cochée est saisie lorsque l'utilisateur entre dans une zone d'édition.

**Code**

```
procedure NBOPER_OnEnter (Sender: TObject);
begin
  This.chk4.checked:=true;
end;
```

**Exemple 7****Description :**

Un composant CChart effectue une rotation selon le déplacement d'un trackbar.

**Code**

```
procedure TrackBar1_OnChange (Sender: TObject);
begin
  This.series1.RotationAngle:= This.TrackBar1.Position;
end;
```

**Exemple 8****Description :**

Rotation continue d'un composant CChart.

**Code**

```
procedure Timer1_OnTimer (Sender: TObject);
begin
  if(This.Series1.RotationAngle =359)then
  begin
    This.Series1.RotationAngle :=0;
  end
  else
  begin
    This.Series1.RotationAngle := This.Series1.RotationAngle+1;
  end;
```

```
end;
```

### Exemple 9

Voir **Erreur ! Source du renvoi introuvable.**

### Exemple 10

```
procedure TreeView1_OnCheckNode (Sender: TObject; Node: TNode; Value:
Integer; var Allow: Boolean);
begin
  // on autorise l'utilisateur à cocher
  // ssi le noeud n'a pas d'enfants
  Allow := node.count = 0;
end;
```

### Exemple 11

Voir « **A partir de** » dans sous fichiers

### Exemple 12

Voir **Activer/désactiver éléments de menu OnPopup**

### Exemple 13

```
if (MessageDlg('Confirmation ? ', mtConfirmation, mkset (mbYes,
mbNo), 0) = MrYes) then
begin
  ...
end;
```

Les valeurs possibles pour les boutons sont :

mbYes, mbNo, mbOK, mbCancel, mbAbort, mbRetry, mbIgnore, mbAll,  
mbNoToAll, mbYesToAll, mbHelp

Les valeurs possibles pour le retour sont :

mrYes, mrNo, mrNone, mrOk, mrCancel, mrAbort, mrRetry, mrIgnore, mrAll,  
mrNoToAll, mrYesToAll

**Exemple 14**

Survol d'un sfl : voir chapitre Survol d'un sfl

**Exemple 15**

Défilement d'un sfl lors du dragover :

```
procedure SFL1_OnDragOver (Sender: TObject; Source: TObject; X: Integer;
Y: Integer; State: TDragState; var Accept: Boolean);
begin
  if (( X + 10) > This.SFL1.ClientWidth)then
  begin
    SendMessage(This.SFL1.Handle,276,1,0)
  end
  else if( x < 10)then
  begin
    SendMessage(This.SFL1.Handle,276,0,0)
  end;
  if (( Y + 10) > This.SFL1.ClientHeight)then
  begin
    SendMessage(This.SFL1.Handle,277,1,0)
  end
  else if( y < 10)then
  begin
    SendMessage(This.SFL1.Handle,277,0,0)
  end;
end;
```

**Exemple 16****Description**

Pour afficher les bandes de ruptures,alors qu'il n'y a pas de ligne de détail, vous pouvez procéder comme suit :

Ajoutez un champ dans la bande de détail que l'on nommer ici toPrint.  
Modifiez la propriété visible de ce champ à false.

Dans le programme, affectez la valeur à Y lorsque les valeurs sont à afficher, et à N lorsqu'il ne faut pas les afficher.

Utilisez l'evt local onPrintBand de la fiche :

**Code :**

```
procedure myDetailBand_OnPrintBand (sender: TObject; var print:
Boolean);
begin
  print := This.toPrint.getvalue()='Y';
end;
```

---

# Chapitre 6. Applications Multifenêtres

---

## Introduction

Lorsqu'un programme silverdev appelle un autre programme silverdev, il y a alors deux fenêtres affichées à l'écran.

Le nombre de fenêtres affichées n'est pas limité.

Contrairement à une application 5250, l'utilisateur peut accéder à plusieurs fenêtres en même temps.

La première fenêtre affichée est la fenêtre principale.

Lorsque la fenêtre principale est fermée par l'utilisateur, le processus sur la partie cliente et le job sur le serveur se terminent.

---

## Organisation des fenêtres

Le comportement par défaut est que chaque fenêtre est indépendante.

Les fenêtres sont alors dites flottantes.

Exemple de fenêtres flottantes :

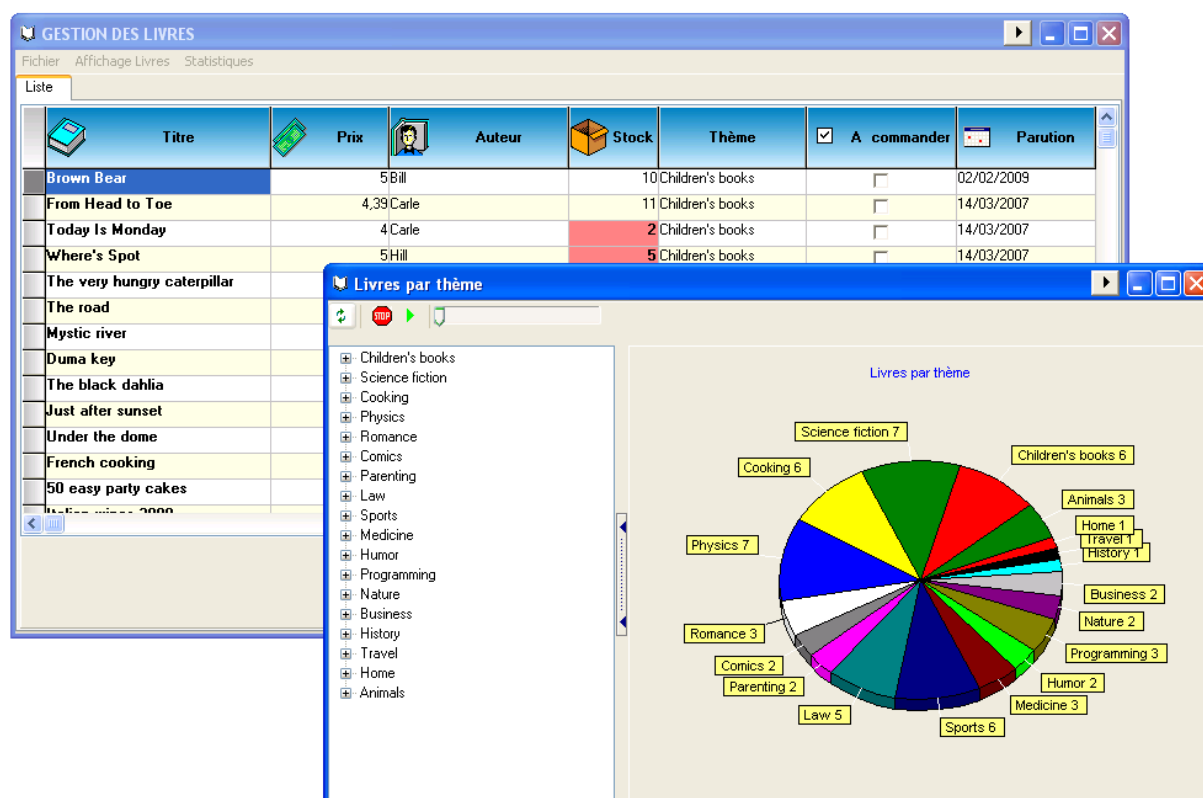


Figure 3

Il est possible d'organiser les fenêtres avec l'architecture mdi ou dans des onglets comme indiqué dans les deux chapitres suivants.

## Pile d'appels

Lorsque le premier programme est lancé, il appelle la fonction `sdStart` qui entre dans une boucle d'écoute des événements serveur.

Lorsqu'un second programme est appelé, ce programme est chargé en mémoire, puis, le programme sort de la pile d'appel. Il est toujours chargé en mémoire, et les handlers d'événements seront rappelés lors des événements correspondants.

## Interactions entre programmes

Le fait que l'utilisateur ait accès à plusieurs écrans en même temps apporte une situation nouvelle par rapport à la programmation 5250. Il est possible que vous vouliez qu'une action sur un écran entraîne un changement sur un autre écran.

Si on suit la règle un écran un programme, cela signifie qu'il faut depuis un programme exécuter du code qui se situe dans un autre programme.

Soit pgm1 et pgm2 deux programmes, ayant les écrans respectifs ecran1 et ecran2  
le programme pgm1 appelant le programme pgm2.

### Cas 1 : Sur une action dans ecran1, on souhaite effectuer une modification sur ecran2

Le programme pgm2 doit prendre en paramètres un pointeur de procédure et assigner ce pointeur avec l'adresse de la fonction de pgm2 qui doit être exécuté dans pgm1

Pgm 1 :

```

D ptrfnc      s      *   procptr

D fnc         pr      extproc(ptrfnc)
...

C             call    'PGM2'
C             parm    ptrfnc
...

/Free
  Callp fnc();
/end-free

```

Pgm 2 :

```

D ptrfnc      s      *   procptr

D fnc         pr

C      *entry  plist
C             parm    ptrfnc

/Free
  ptrfnc = %paddr('FNC');
/end-free

P fnc         B
D             pi

```



...

P fnc

E

## Cas 2 : Sur une action dans ecran2, on souhaite effectuer une modification sur ecran1.

Le programme pgm1 doit passer l'adresse de la fonction qui sera exécuté dans pgm2 :

Pgm 1 :

```

D ptrfnc      s      *   procptr

D fnc         pr      extproc(ptrfnc)

/Free
  ptrfnc = %paddr('FNC');
/end-free

C              call    'PGM2'
C              parm    ptrfnc

P fnc         B
D              pi
D parm1       10u 0
D parm2       10
...
P fnc         E

```

Pgm 2 :

```

D ptrfnc      s      *   procptr

D fnc         pr      extproc(ptrfnc)
D parm1       10u 0
D parm2       10

C      *entry   plist
C              parm    ptrfnc

/Free
  Callp fnc();

```

```
/end-free
```

Remarque 1 : l'argument passé à la fonction intégrée %paddr doit être en majuscule. Cela vient du fait que le compilateur rpg convertit le source en majuscules.

Remarque 2 : Les deux dernières solutions peuvent être combinées.

Remarque 3 : Il peut être tentant de ne passer en paramètre entre les programmes uniquement les handles de fenêtres (F1) et de saisir en dur les noms de composants de écran1 dans pgm2. Cela est une mauvaise idée car pgm1 et pgm2 se trouvent alors fortement liés.

## Paramètres

La fonction externe peut avoir des paramètres. Il faut dans ce cas définir correctement le prototype.

Le cas 1 devient :

Pgm 1 :

```
D ptrfnc          s          *   procptr

D fnc             pr          extproc(ptrfnc)
D parm1           10u 0
D parm2           10
...

C                call        'PGM2'
C                parm        ptrfnc
...

/Free
  Callp fnc(arg1:arg2) ;
/end-free
```

Pgm 2 :

```
D ptrfnc          s          *   procptr
```

```
D fnc          pr
D parm1          10u 0
D parm2          10

C      *entry      plist
C          parm          ptrfnc

/Free
  ptrfnc = %paddr('FNC');
/end-free

P fnc          B
D          pi
D parm1          10u 0
D parm2          10
...
P fnc          E
```

Attention, le rpg n'ayant pas de pointeurs typés, le compilateur ne détectera pas d'erreur si l'adresse assignée à ptrfnc est l'adresse d'une fonction avec des paramètres différents, mais une erreur se produira à l'exécution.

---

# Chapitre 7. Applications MDI

---

## Introduction

Une application MDI est une application ou il existe une fenêtre mère et des fenêtres filles.

Les fenêtres filles ne peuvent pas sortir du cadre de la fenêtre mère et le menu de la fenêtre mère est toujours accessible.

Des fonctions permettent de ranger les fenêtres en mosaïque.

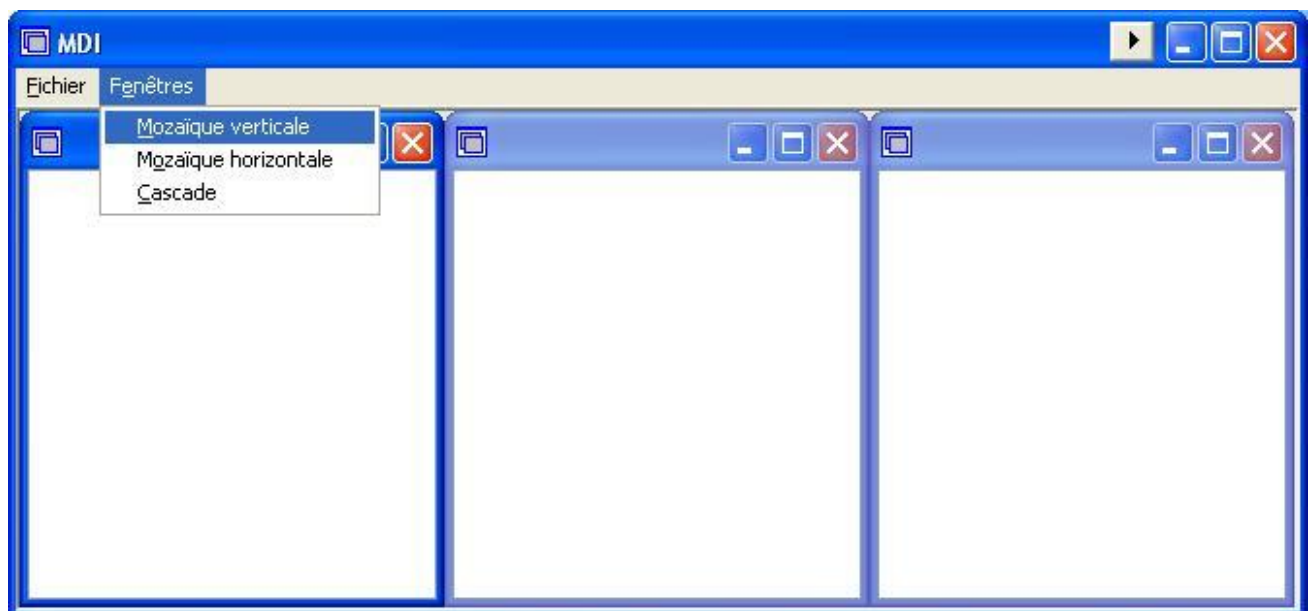


Figure 4

Pour qu'une fenêtre soit mère, il faut modifier sa propriété `formstyle` à `fsMDIFORM`.  
Pour qu'une fenêtre soit fille, il faut modifier sa propriété `formstyle` à `fsMDICHILD`.

Une fenêtre fille ne peut pas exister s'il n'existe pas une fenêtre mère.

Une fenêtre fille ne peut pas être invisible. Si l'utilisateur clique sur la croix d'une fenêtre fille, la fenêtre se réduit au lieu de se fermer. Pour réellement la fermer, libérez la fenêtre dans son événement `onclose`.

---

## Exemple :

L'exemple suivant est composé de 3 programmes contenant chacun une fenêtre.

Les deux programme Child1 et Child2 ne peuvent pas être appelés seuls puisque leur fenêtre est de type fsMdiChild.

## Programme MDIFORM

```

h dftactgrp(*no)
h bnmdir('SILVERDEV')
/copy h,silverdev
*
*
d F1                S                5u 0
d F1REF             S                21A  INZ('*LIBL/MDIFORM')

d Init              pr

d MnuChild1_OnClick...
d                  pr
d  PevtInf          *  const options(*nopass)

d MnuChild2_OnClick...
d                  pr
d  PevtInf          *  const options(*nopass)

d MnuVert_OnClick...
d                  pr
d  PevtInf          *  const options(*nopass)

d MnuHor_OnClick...
d                  pr
d  PevtInf          *  const options(*nopass)

d MnuCascade_OnClick...
d                  pr
d  PevtInf          *  const options(*nopass)

c                  eval  *inRT = *on
c                  callp sdStart(%paddr('INIT'))

p Init              B
d                  PI
c                  If    F1 = 0
c                  eval  F1 = sdcreateForm(F1REF)
c                  callp sdSetCallBack(F1
c                        : 'MnuChild1.OnClick'
c                        : %paddr(
c                          'MNUCHILD1_ONCLICK'))
c                  callp sdSetCallBack(F1
c                        : 'MnuChild2.OnClick'

```

```

c                                :%paddr (
c                                'MNUCHILD2_ONCLICK'))
c                                callp    sdSetCallBack (F1
c                                : 'MnuVert.OnClick'
c                                :%paddr (
c                                'MNUVERT_ONCLICK'))
c                                callp    sdSetCallBack (F1
c                                : 'MnuHor.OnClick'
c                                :%paddr (
c                                'MNUHOR_ONCLICK'))
c                                callp    sdSetCallBack (F1
c                                : 'MnuCascade.OnClick'
c                                :%paddr (
c                                'MNUCASCADE_ONCLICK'))
C                                Endif
c                                callp    sdShow (F1)
p                                E
pMnuChild1_OnClick...
p                                B
d                                PI
d PevtInf                        *    const options(*nopass)
C                                call    'CHILD1'
p                                E

pMnuChild2_OnClick...
p                                B
d                                PI
d PevtInf                        *    const options(*nopass)
C                                call    'CHILD2'
p                                E
pMnuVert_OnClick...
p                                B
d                                PI
d PevtInf                        *    const options(*nopass)
C                                callp    sdSet (F1: '*FORM': 'TileMode': 'tbVertical')
C                                callp    sdTile (F1)
p                                E
pMnuHor_OnClick...
p                                B
d                                PI
d PevtInf                        *    const options(*nopass)
C                                callp    sdSet (F1: '*FORM': 'TileMode': 'tbHorizontal')
C                                callp    sdTile (F1)
p                                E
pMnuCascade_OnClick...
p                                B
d                                PI
d PevtInf                        *    const options(*nopass)
C                                callp    sdCascade (F1)

```

p

E

## Programme Child1

```

h dftactgrp(*no)
h ACTGRP('SILVERDEV')
h bnmdir('SILVERDEV')
/copy h,silverdev
d F1          S          5u 0
d F1REF       S          21A  INZ('*LIBL/CHILD1')

d OnClose     pr
d PevtInf          *    const options(*nopass)
c
c          eval      *inRT = *on
C          If        F1 = 0
c          eval      F1 = sdcreateForm(F1REF)
c          callp     sdSetCallBack(F1
c                  : 'OnClose'
c                  : %paddr(
c                  'ONCLOSE'))
C          endif
C          Callp     sdShow(F1)
pOnClose     B
d            PI
D Parameters  ds          based(pevtinf)D Win
5u 0
D Evt          48a
,* Variable
D Action      10i 0
,* 0 : ne rien faire
,* 1 : Cacher la fenêtre
,* 2 : Libérer la fenêtre
,* 3 : Minimiser la fenêtre
,*d PevtInf          *    const options(*nopass)
c          eval      action = 2
c          callp     sdFreeForm(Win)

p          E

```

## Programme Child2

Idem Child1, remplacer

d F1REF	S	21A	INZ('*LIBL/CHILD1')
---------	---	-----	---------------------

par

d F1REF	S	21A	INZ('*LIBL/CHILD2')
---------	---	-----	---------------------





## Chapitre 8. Application multifenêtres avec des onglets

### Introduction

Une solution alternative à une application mdi est de mettre les fenêtres dans des onglets comme dans l'image suivante :

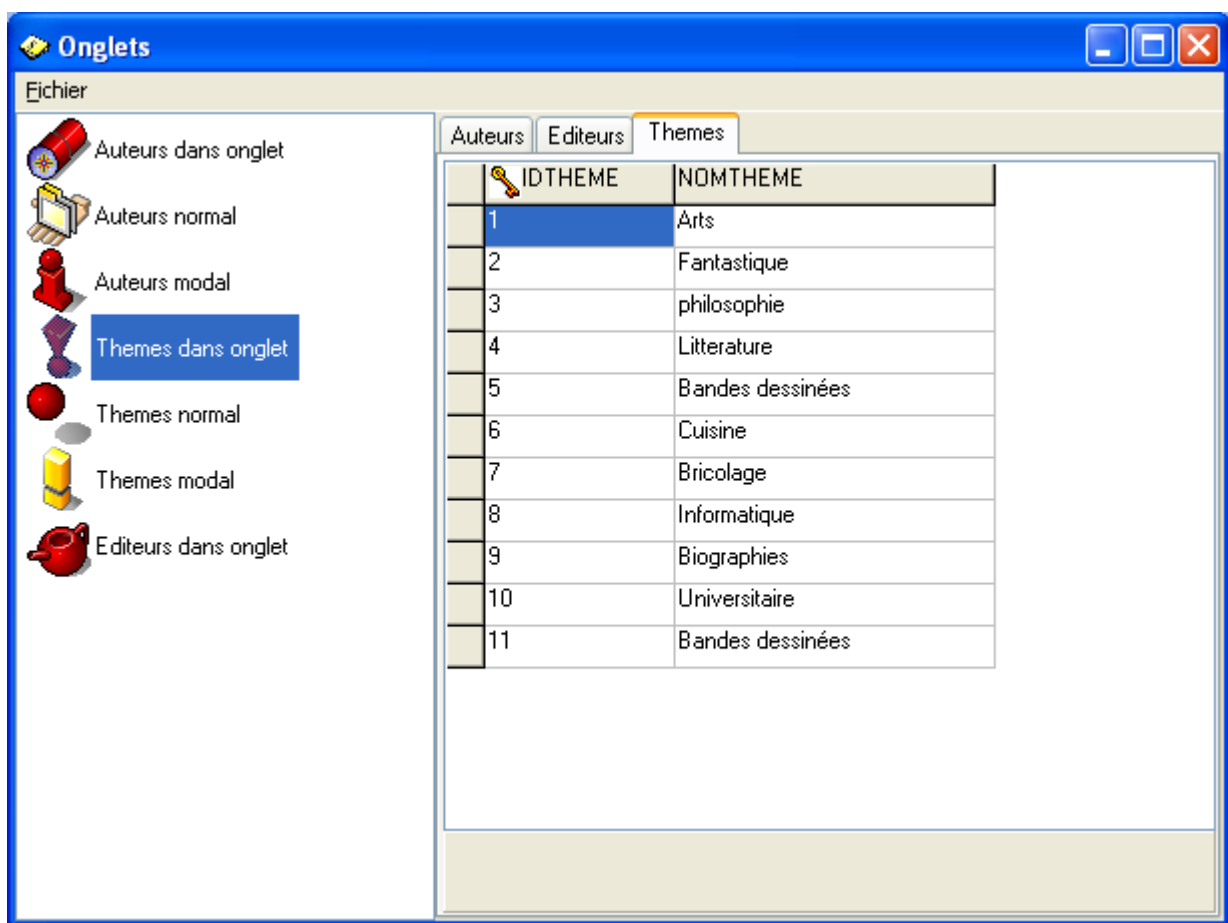


Figure 5

### Placer un composant dans un conteneur

Pour placer un composant dans un autre composant à l'exécution, il faut utiliser la fonction `sdSetCtrl` et la propriété `parent`.

#### Exemple 1 :

Pour placer un composant dans un panel :

```
c          callp      sdSetCtrl (F1: 'ListBox1': 'parent': F1: 'Panel1')
```

Pour que le composant occupe toute la place dans le panel :

```
C          callp      sdSet (F1: '*FORM': 'align': 'alclient')
```

### Exemple 2 :

Pour placer une fenêtre F1 dans un panel de la fenêtre F2 :

```
c          callp      sdSetCtrl (F1: '*FORM': 'parent': F2: 'Panel1')
```

### Exemple 3 :

Création d'un onglet dans une fenêtre F1 et insertion d'une fenêtre F2 dans l'onglet :

```
c          callp      sdcreate (F1: 'TTabSheet': 'Tab1')
c          callp      sdSetCtrl (F1: 'Tab1': 'pagecontrol':
c                      F1: 'pagecontrol1')
c          callp      sdSetCtrl (F2: '*FORM': 'parent': F1: 'Tab1')
C          callp      sdSet (F2: '*FORM': 'BorderStyle': 'bsNone')
C          callp      sdSet (F2: '*FORM': 'align': 'alclient')
```

---

## Sources

Ci dessous, l'exemple complet :

### Programme PGM1

```
*/EVENT ListView1_OnDbClick
* -----*
* Description :                               *
* -----*
D Parameters      ds          based(pevtnf)
D Win              5u 0
D Evt              48a
*
D pgm              s          20      inz (*LIBL/PGM2')
c                  call      pgm
C                  parm      F1
```

```

*/EVENT MnuClose_OnClick
* -----*
* Description :
* -----*
Parameters      ds              based(pevtfinf)
Win              5u 0
Evt              48a
*
nameTab          s              30      varying
DnumWin          s              5u 0
C                eval          NumWin = sdGetInt(F1:'Pagecontrol1':
C                'ActivePage.Tag')
C                callp          sdClose(numWin)

```

## Programme PGM2

```

*/BLOCK RPGSPCID
* ----- RPGSPCID : Data descriptions (D Spec.)
D tab            s              30      varying
DFParent         s              5u 0

*/BLOCK RPGPARM
* ----- RPGPARM : Program parameters
C      *entry     plist
C                parm              FParent

*/BLOCK RPGINITEND
* ----- RPGINITEND : End of Initializations procedure
c                eval          tab = 'Tab'+%char(F1)
c                callp          sdcreate(FParent:'TTabSheet':tab)
C                callp          sdSetInt(FParent:Tab:'Tag':F1)
c                callp          sdSetCtrl(FParent:tab:'pagecontrol':
c                fParent:'pagecontrol1')
c                callp          sdSetCtrl(F1:'*FORM':parent:FParent:
c                Tab)
C                callp          sdSet(F1:'*FORM':BorderStyle:'bsNone')
C                callp          sdSet(F1:'*FORM':align:'alclient')
*/BLOCK RPGBEFORESHOW
* ----- RPGBEFORESHOW: Before show(F1)
C                callp          sdSetBool(Fparent:Tab:'TabVisible':*on)
C                callp          sdSetCtrl(Fparent:'pageControl1':
C                'ActivePage':Fparent:Tab)

*/EVENT OnClose
* -----*
* Description :
* -----*
D Parameters      ds              based(pevtfinf)

```

```
D Win          5u 0
D Evt          48a
  * Modifiable
D Action       10i 0
  * 0 : ne rien faire
  * 1 : Cacher la fenêtre
  * 2 : Libérer la fenêtre
  * 3 : Minimiser la fenêtre
  *
C              callp      sdSetBool (FParent:Tab: 'TabVisible' :*off)
```

*Remarque : Un programme du même type, un peu plus complexe se trouve dans les programmes de démo, (voir sources de SDDMTAB1, SDDMTAB2, SDDMTAB3, SDDMTAB4)*

---

# Chapitre 9. MultiOccurrences d'une fenêtre

---

## Introduction

Le présent chapitre explique comment réaliser une application avec une fenêtre ayant plusieurs occurrences. Cela signifie que à chaque appel d'un programme, une nouvelle fenêtre sera créée. Le même programme gère alors plusieurs fenêtres. Les multiples occurrences peuvent être flottantes ou organisées en fenêtres mdi ou en onglet comme indiqué dans les deux chapitres précédents.

---

## Variable de fenêtre

Silverdev génère le code suivant :

```
/Free
  If F1 = 0 ;
    F1 = sdCreateForm(F1REF) ;
/End-Free
```

C'est-à-dire qu'une fenêtre n'est créée qu'au premier appel.

Nous allons ajouter dans le moment afterShow le code suivant :

```
/BLOCK RPGAFTERSHOW
// ----- RPGAFTERSHOW : After show(F1)

C                               eval      F1 = 0
```

Ainsi, à chaque appel du programme, F1 sera égal à zéro, et la fonction sdCreateForm sera ré appelée.

---

## Evènements

Dans le paragraphe précédent, nous avons vu que la valeur de la variable F1 n'est pas conservée.

Il ne faut donc plus utiliser la variable F1 dans les évènements.

Chaque évènement possède des paramètres. Ces paramètres sont variables selon les évènements, mais deux paramètres sont toujours présents.

Il s'agit des paramètres Win et evt

```
~/EVENT OnClose
,* -----*
,* Description :
,* -----*
D Parameters      ds          based(pevtinf)
D Win              5u 0
D Evt              48a
```

Utilisez toujours le paramètre Win dans les évènements pour les fenêtre à multiples occurrences.

---

## Stockage des données

Imaginons à présent que nous avons besoin de stocker des valeurs pour chaque fenêtres. Un numéro de client, un chemin sur l'ifs etc...

Puisque la fenêtre peut être instanciée plusieurs fois, il faut conserver autant de fois ces valeurs qu'il y a de fenêtres en ayant un moyen de retrouver ces valeurs à l'aide du numéro de fenêtre passé dans l'évènement.

On peut imaginer plusieurs implémentations : un tableau statique (le nombre d'occurrences sera alors limité), un fichier de base de données, un tableau dynamique ou une liste chaînée.

Nous vous proposons une implémentation utilisant un service programme dédié. Voir le chapitre suivant.

---

# Chapitre 10. Multioccurrences , programme de service sdsrvlst

---

## Introduction

Nous allons créer un programme child qui affichera les données d'un client. Plusieurs fenêtres créées par le programme child peuvent exister en même temps.

Un service programme que l'on appellera srvcld permettra d'avoir toutes les informations sur les fenêtres existantes. Ce service programme conservera des informations spécifiques à chaque fenêtre.

Le service programme srvcld utilisera un service programme fourni avec silverdev, le programme de service sdsrvlst

sdsrvlst ressemble beaucoup à la classe vector de java ou c++.

La programmation avec sdsrvlst s'apparente d'ailleurs beaucoup à de la programmation objet.

Les champs de la structure DsList peuvent faire penser à des propriétés, les fonctions de sdsrvlst peuvent faire penser à des méthodes.

Des "objets" sont créés avec %alloc, puis ajoutés à un "objet" de type DsList de la même façon que vous créeriez un objet en c++ pour l'ajouter à un objet de type Vector.

Il ne s'agit que d'une analogie, pas d'une véritable programmation objet, et donc, (malheureusement) pas d'héritage ni de polymorphisme.

---

## Service programme sdsrvlst

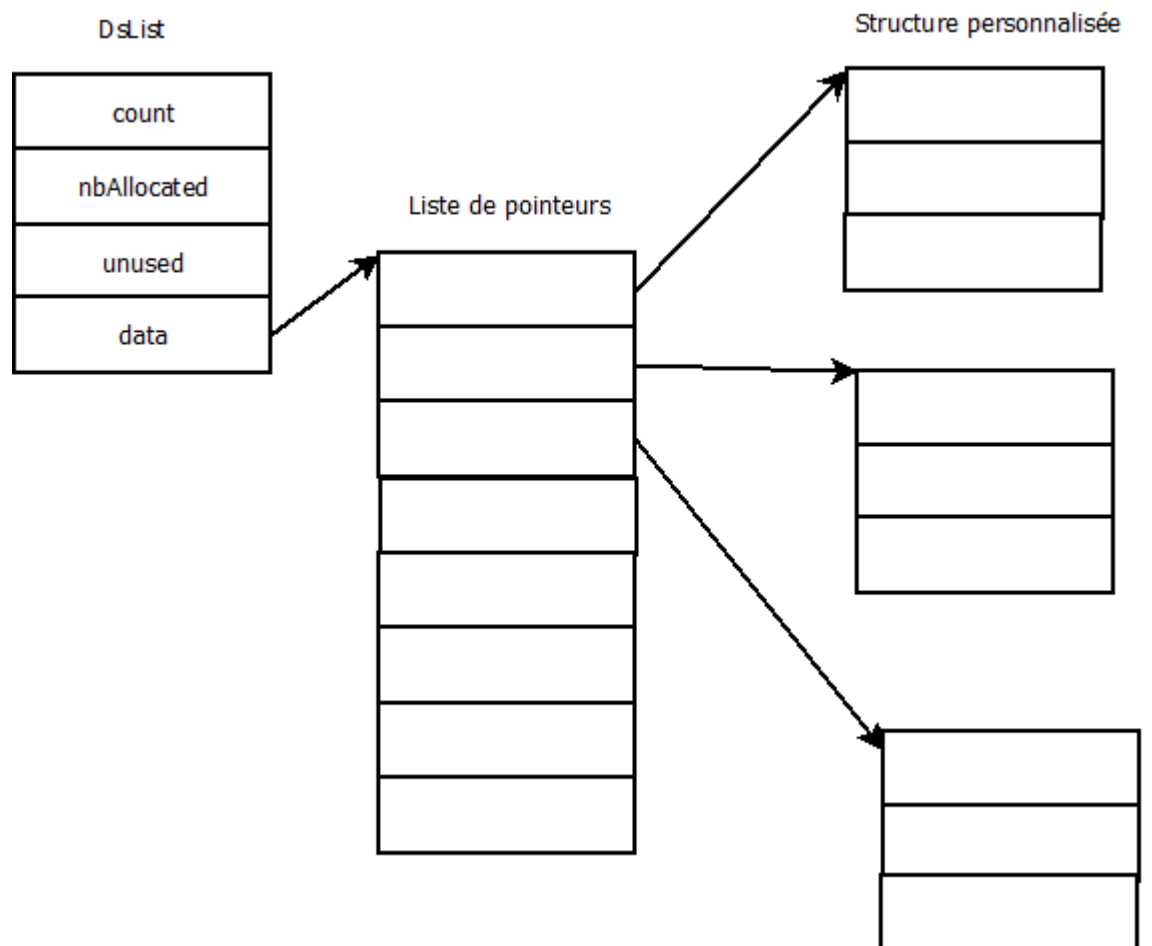
Le service programme sdsrvlst est fourni avec silverdev. Un répertoire de liage sdsrvlst est fourni, et les prototypes sont dans silverdev,h,sdsrvlst.

Ce service programme permet de gérer des listes de pointeurs. Il contient toutes les fonctions nécessaires pour ajouter un élément, supprimer un élément, parcourir tous les éléments, échanger deux éléments, trier etc..

Fonctions du service programme sdsrvlst :

addItem	Permet d'ajouter un pointeur dans la liste.
clearList	Permet de supprimer tous les éléments de la liste. Le paramètre freeltems permet de préciser si la mémoire sur les pointeurs doit être désallouée par la fonction.
Deleteltem	Permet de supprimer un élément de la liste. Le paramètre

	freeltem permet de préciser si la mémoire sur le pointeur doit être désallouée par la fonction.
DeleteltemIdx	Permet de supprimer un pointeur de la liste à partir de son index.
exchangeltems	Permet d'échanger deux pointeurs de la liste
exchangeltemsIdx	Permet d'échanger deux pointeurs de la liste à partir de leurs index
getltem	Permet de récupérer un élément à partir de son index . Avec le champ count de DsList, cette fonction permet de parcourir tous les éléments de la liste.
indexOf	Permet de retrouver l'index d'un élément. Renvoie -1 si le pointeur n'est pas dans la liste.
SortList	Permet de trier une liste. Cette fonction prend un paramètre de type callback.(Voir exemple suivant)



Le champ count peut être lu directement. (il ne faut pas le modifier)



Le champ allocated est utilisé en interne.

Le champ unused sert à aligner le champ data sur 16 octets

Le champ data pointe sur des données allouées dynamiquement.

La liste de pointeurs est une zone de données allouée dynamiquement. Si besoin, le module réalloue une zone plus grande .(si un addItem est effectué et qu'il n'y a plus de place libre)

Dans la liste de pointeurs, on trouve tous les 16 octets un pointeur pointant sur une structure de donnée elle aussi allouée dynamiquement.

La fonction clearList va remettre le champ count à zéro, libérer la mémoire pointée par le champ data et si le second paramètre est à \*on, la mémoire allouée pour chaque structure personnalisée sera libérée.

### Exemple :

#### Déclaration de la structure personnalisée :

D dsClient	ds	qualified template
D id	10	0
D name	50	varying
D adress	250	varying

#### Déclaration d'une variable de type client (pointée par ptrclient)

D client	ds	likeDs (DsClient)
D		based (ptrClient)

#### Déclaration de la liste des clients:

D clients	ds	likeDs (dsList) inz
-----------	----	---------------------

#### Ajout d'un client dans la liste :

/free
ptrClient = %alloc(%size(DsClient)) ;
clear client;
client.name = NAME;
addItem(clients:ptrClient) ;
/end-free

#### Relecture de la liste des clients :

```

D i          s          10u 0
/free
for i = 1 to clients.count;
  ptrClient = getItem(clients:i);
  sdSetCell(F1:'sflClients':'name':i:client.name);
endfor;
/end-free

```

### **Exemple d'utilisation de sortList :**

La fonction sortList permet de trier une liste.

Cette fonction prend en paramètre une fonction callback.

```

P procCompare      B
d                  pi          N
D ptr1            *      value
D ptr2            *      value

D child1          ds          likeDs (DsChild)
D                  based(ptr1)

D child2          ds          likeDs (DsChild)
D                  based(ptr2)
/free
return child1.numCli > child2.numCli;
/end-free

```

Appel de la fonction sortlist :

```
callp sortlist(children:%paddr(procCompare));
```

---

## **Service programme srvchld**

Le service programme srvchld sera constitué d'un membre pour les prototypes H/srvchld, d'un membre pour la définition des fonctions qrpglesrc/srvchld, d'un membre de définition des exportations qsrvsrvc/srvchld et d'un répertoire de liage srvchld.

Le membre qrpglesrc/srvchld contient une variable de type DsList:

D children	ds	likeDs (DsList)
------------	----	-----------------

Le type DsList est defini dans sdsrvlst.

Cette liste contiendra des éléments de type DsChild:

D DsChild	ds	qualified based(ptrDummy)
D handle		5u 0
D numClient		10 0

Ici, la structure DsChild est très simple, mais elle peut contenir beaucoup plus de champs.

Elle peut contenir des chaines, des "file descriptor" (si la fenêtre ouvre un fichier ifs par exemple), des pointeurs, des structures, et même des structures de type DsList.

La première fonction que nous allons définir dans srvchld est celle permettant la création d'une nouvelle fenêtre.

P CreateNewChild...		
P	B	
D	pi	*
D child	ds	likeDs (DsChild)
d		based(ptrChild)
/free		
eval ptrchild = %alloc(%size(DsChildQuery));		
clear child;		
call additem(children:ptrChild):		
return ptrChild;		
/end-free		
P CreatenewChild...		
P	E	

Nous avons utilisé la fonction addItem du service programme sdsrvlst pour ajouter la structure créée dans la liste.

Si on continue l'analogie avec la programmation objet, la fonction createNewChild peut être considérée comme le constructeur de DsChild.

La seconde fonction que nous allons définir dans srvchld est celle permettant l'ouverture d'une fenêtre client.

P openChild	B	export
D	pi	
D numCli		10 0
D i	s	10u 0
D pWin	s	5u 0
D child	ds	likeDs (DsChildQuery)

```

D                                     based(ptrChild)
/free
  for i = 1 to children.count;
    eval ptrChild = getItem(children:i);
    if child.numCli = numCli;
      callp sdShow(child.handle);
    return ;
  endif;
endfor;
callp CHILD(pWin);
eval ptrchild = createNewChild;
eval child.handle = pWin;
eval child.numCli= numCli;
/end-free
C
P openChild      E

```

Nous avons utilisé le champ count de DsList et la fonction getItem de sdsrvlst pour parcourir les éléments existants dans la liste.

Une fonction dans srvchld permettra de retrouver le pointeur permettant d'accéder aux informations de cette fenêtre.

```

P getChild      B      export
D              pi      *
D handle              5u 0
D i                s    10i 0
D child            ds    likeds(DsChildQuery)
d                  based(ptrChild)
/free
  for i = 1 to childrenQuery.count;
    eval ptrChild = getItem(children:i);
    if child.handle = handle;
      return ptrchild;
    endif;
  endfor;
  return *NULL;
/end-free
P getChild      E

```

## Programme Child

Le programme child sera un programme silverdev. Il permet de concevoir la fenêtre.

Le programme child contient une variable F1 comme tous les programmes silverdev, mais cette variable n'est pas utilisable dans les événements.

En effet, la variable F1 contient la valeur de la dernière fenêtre créée, mais plusieurs fenêtres créées par ce programme peuvent cohabiter en même temps dans l'application.

Lorsque le programme child crée une fenêtre, il faut que l'appelant récupère l'identifiant créé pour l'assigner à la structure DsChild associée.

On ajoute donc la variable F1 dans les paramètres d'entrée du programme :

```

$*/BLOCK RPGPARM
  // ----- RPGPARM :
c      *entry      plist
C                      parm          F1

```

Le programme child utilisera le service programme srvchld pour avoir accès à aux informations des fenêtres.

Le source du programme devra donc comporter l'instruction :

```
h bnmdir('SRVCHLD')
```

Utilisation du répertoire de liage srvchld

ainsi que :

```
/copy H,srvchld
```

copie des prototypes du service programme srvchld.

Pour chaque évènement se produisant dans le programme on n'utilisera pas la variable F1, mais le paramètre win de l'évènement.

Exemple :

```

~*/EVENT Button1_OnClick
,* ----- *
,* Description :
,* ----- *
D Parameters      ds                      based(pevtinf)
D Win              5u 0
D Evt              48a
,*
,* based variables
D child            ds                      likeDs(DsChild)
D                  based(ptrChild)
/Free

```

```

eval ptrChild = getchild(win);
if ptrChild <> *NULL;
    // you can use child.numCli
endif;
/end-free

```

Si une fenêtre est supprimée, dans l'évènement onclose par exemple, utilisez la fonction sdDeleteltem de sdsrvlst pour supprimer les informations de la fenêtre :

```

~*/EVENT OnClose
,* -----*
,* Description :
,* -----*
D Parameters      ds          based(pevtnf)
D Win              5u 0
D Evt              48a
,* Variable
D Action           10i 0
D child            ds          likeds (DsChild)
d                  based(ptrChild)
/free
eval ptrChild = getchild(Win);
if ptrChild = *NULL;
    eval action = 2;
    callp sdFreeForm(child.handle);
    callp deleteItem(childrenQuery:ptrChild:*on);
endif;
/end-free

```

Nous avons utilisé la fonction deleteItem du service programme sdsrvlst.

Remarque : Le troisième paramètre indique si la fonction doit désallouer la mémoire de la structure de type DsChild. Si ce paramètre est à false, le pointeur sur la structure est enlevé de la liste, mais la mémoire allouée pour ce pointeur n'est pas libérée.

Deux cas où vous pourriez souhaiter mettre ce paramètre à false.

Cas 1 : Si un élément est dans plusieurs listes, lors de la destruction de cet élément vous devez le retirer de toutes les listes le contenant, mais la désallocation ne doit être faite qu'une seule fois. On dit parfois qu'une liste possède ou pas l'objet. Dans ce cas, l'appel à addItem sera sans doute sortie du constructeur.

Cas 2 : Si la structure DsChild est plus complexe et contient elle-même des pointeurs, vous aurez besoin d'écrire une fonction spécifique pour la destruction des objets. (Comme un destructor en C++), vous devrez alors appeler deleteItem avec \*off en troisième paramètre.

---

## Prototypes

Un membre dédié aux prototype n'est pas obligatoire, mais est conseillé pour une programmation plus propre.

Le membre sera inclus par un /copy dans les programmes qui utiliseront le service programme (en plus de bnmdir).

Le membre sera aussi inclus par un /copy dans le membre qrpglesrc/srvchld.

---

## Membre d'exportation

Le membre d'exportation du service programme srvchld n'est pas obligatoire. Si vous ne le créez pas, vous devez préciser EXPORT \*ALL dans la commande crtsrvpgm. (la valeur par défaut du paramètre export est \*srcfile)

L'utilisation d'un membre d'exportation a l'avantage de pouvoir gérer les signatures du service programme.

Voici à quoi peut ressembler votre source d'exportation :

```
STRPGMEXP  PGMLVL(*CURRENT) SIGNATURE('SRVCHLD')
export(createNewChild)
export(OpenChild)
export(getChild)
export(FreeChild)
ENDPGMEXP
```

---

## Répertoire de liage

Créez un répertoire de liage srvchld avec la commande crtbnmdir, puis ajoutez le service programme srvchld dans le répertoire de liage avec la commande WRKBNDDIRE BNDDIR(SRVCHLD) ou ADDBNDDIRE.

Le répertoire de liage peut être ajouté dans les programmes qui utilisent le service programme en carte h.

---

# Chapitre 11. Exemples et astuces

---

## « A partir de » dans sous fichiers

Pour se placer sur une ligne du sous fichier lors de la saisie d'un texte dans un CEdit, utilisez les événements locaux :

```
procedure Edit1_OnChange (Sender: TObject);
var
  cpt:integer;
  temp:string;
begin
  for cpt:= 1 to This.SFL1.Lines do
  begin
    temp:= This.SFL1.CellValue['titre',cpt];
    if(UpperCase(temp) >= UpperCase(_1.Edit1.Text)) or(cpt
= _1.SFL1.Lines) then
    begin
      This.SFL1.GotoCell('titre',cpt,false);
      break;
    end;
  end;
end;
```

---

## Sous fichiers et F11

Dans les applications 5250, vous étiez sans doute habitué à modifier la vue des sous fichiers avec la commande F11.

Le sous fichier de silverdev possède un ascenseur horizontal, il est donc possible de complètement s'affranchir de ces différentes vues.

Pourtant certains utilisateurs souhaiteront encore avoir différentes vues sur le sous fichier.

La solution que nous préconisons, est d'utiliser un seul sous fichier. Lorsque l'utilisateur fera basculer les vues, vous rendrez visibles ou non les colonnes.

---

C	callp	sdSetBool(F1:'sf11':
---	-------	----------------------

---



C

`'colbyname('col2').visible':*ON)`

La modification de la vue pourra être effectuée par exemple sur l'une des opérations suivantes :

- \_Popupmenu sur le sous fichier.
- \_Raccourci F11 (utilisez un Taction)
- \_Des boutons
- \_Des onglets

Dans le cas des onglets, utilisez un CPageControl.

Réduisez celui ci en hauteur pour qu'on ne voit que les tabsheet, mettez sa propriété align à alTop, puis placez le sous fichier en alclient juste dessous. Sur l'événement onshow des tabsheet, modifiez les visibilité des colonnes.

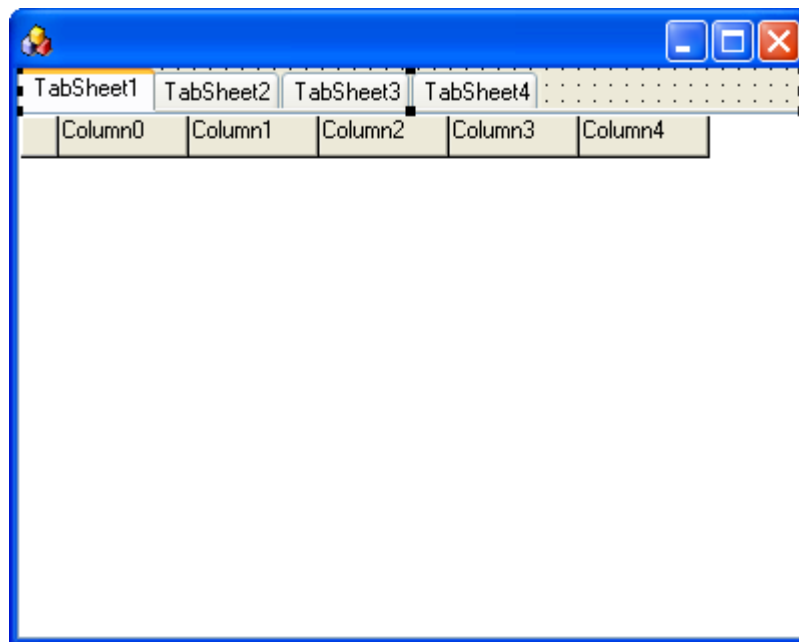


Figure 6

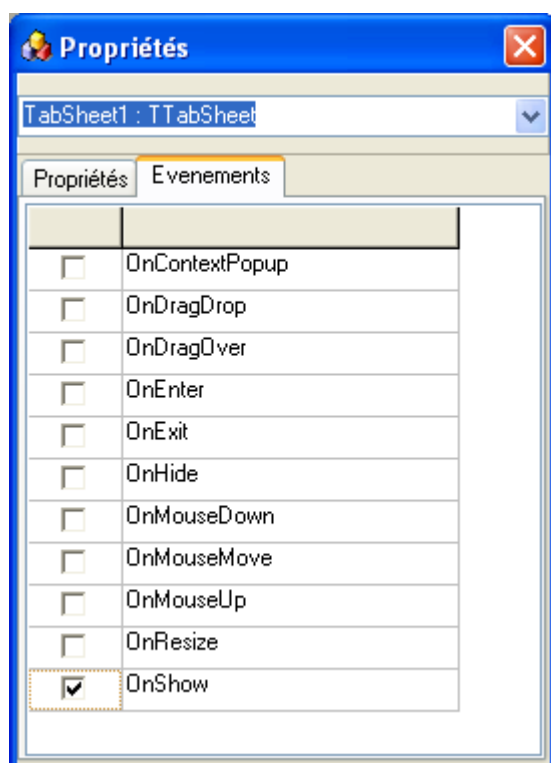


Figure 7

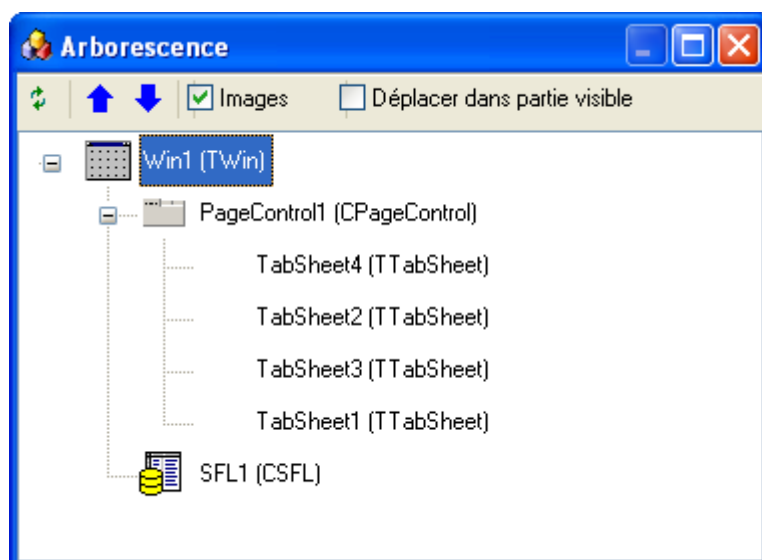


Figure 8

---

## Fenêtre d'attente

### Introduction :

Dans l'exemple qui suit, nous allons réaliser une fenêtre d'attente qui sera réutilisable par plusieurs programmes.

## Fenêtre

Commencez par créer une fenêtre contenant une progressBar et un label.

Modifiez la propriété name de la progressBar à Bar1.

Modifiez les propriétés borderIcons à [ ] et position à poScreenCenter.

Modifiez les propriétés de Label1 : align = alClient, alignment = taCenter et Layout = tlCenter

Cochez l'évènement OnAfterShow de la fenêtre.

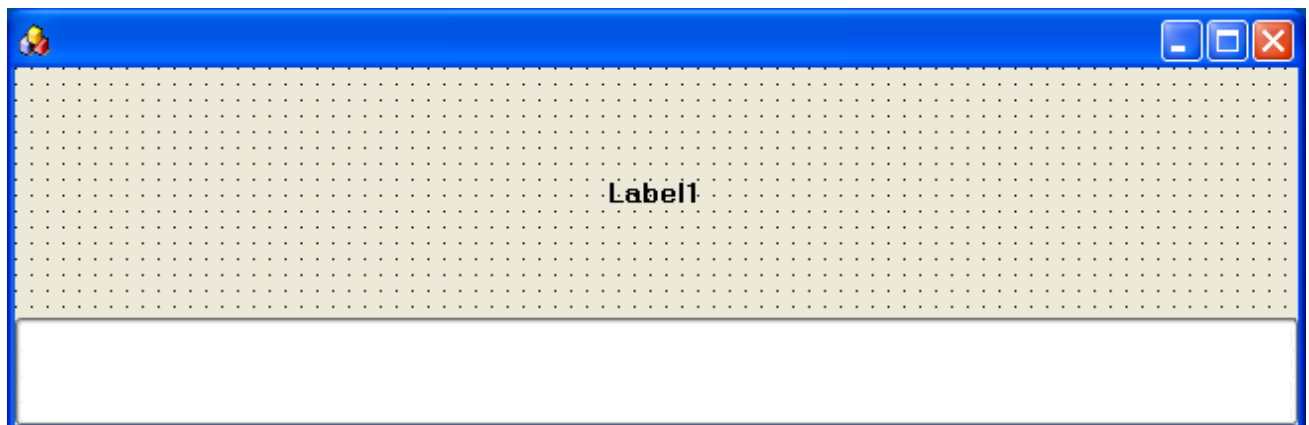


Figure 9

## Code :

```

*/BLOCK RPGSPCID
* ----- RPGSPCID : Data descriptions (D Spec.)
Dpfonc          s          *   procptr
DFonc           pr          extproc(pfonc)
DFx              5u 0
C   *entry       plist
C               parm          PFONC
*/BLOCK RPGBEFORESHOW
* ----- RPGBEFORESHOW: Before show(F1)
C               callp         sdCursor(*off)
*/BLOCK RPGAFTERSHOW
* ----- RPGAFTERSHOW : After show(F1)
C               callp         sdSetInt(F1:'Bar1':'Position':0)
C               callp         sdCursor(*on)
*/EVENT OnAfterShow
* -----*
* Description :                      *
* -----*
D Parmameters    ds          based(pevtinf)

```

D Win		5u 0
D Evt		48a
C	callp	fonc(F1)
C	callp	sdClose(F1)

**Programme appelant :**

```

*/BLOCK RPGSPCID
* ----- RPGSPCID : Data descriptions (D Spec.)
DTraitement      pr
DAttente          5u 0

*/BLOCK RPGPROCDEF
* ----- RPGPROCDEF : User procedures
PTraitement      B
D                pi
DAttente          5u 0
D i              s      10i 0
C                callp  sdSetInt(FAttente:'Bar1':'Max':10)
C                for   i = 1 to 10
C                ...
C                callp  sdSetInt(FAttente:'Bar1':'position':i)
C                callp  sdSetString(FAttente:'Label1':'caption'
C                        %char(i) + '/'10')
C                callp  sdRefresh(FAttente:'Label1')
C                callp  sdApplySet
C                endfor
PTraitement      E

*/EVENT Button1_OnClick
D Parmameters      ds                based(pevtinf)
D Win              5u 0
D Evt              48a
D pgm              s      20      inz('ATTENTE')
D ptraitement      s      *      procptr
*
C                eval   ptraitement = %paddr(TRAITEMENT)
C                call   pgm
C                parm                    ptraitement

```

**Remarque :**

Il est aussi possible d'utiliser les fonctions `sdStrLongProcess` et `sdEndLongProcess` et d'inclure le traitement dans la fenêtre. Cette méthode est plus simple, mais le programme doit être réécrit pour chaque application.

## Activer/désactiver éléments de menu OnPopup

```

procedure TForm1.PopupMenu1Popup(Sender: TObject);
var
  Statut:string;
  row:integer;
begin
  row:= This.CSFL1.RowSelected;
  Statut:='';
  if(row >=1) then
  begin
    Statut:= This.CSFL1.GetCellValue('statut',row);
  end;
  This.menuitem1.Enabled := (row >=1) and ((Statut ='1') or (Statut =
'2'));
end;

```

## Fenêtres de choix

Lorsqu'une fenêtre doit être utilisée pour que l'utilisateur effectue un choix, créez un programme avec une fenêtre en modal.

Pour cela, sur le nom du programme, choisissez l'option 8, et sélectionnez mode

```

SilverDev                               Programmes SilverDev
                                           22/10/07  14:34:46

Nom programme . . . . . : SVCHXAGE      Créé par : ADUVAL      Le 2007-02-15

Description . . . . . : Choix Agenda
Fenetre principale (SDF): SVCHXAGE      Biblio. : *LIBL
Mode affiche fenetre *MODAL      (*NOMODAL, *MODAL, *NOSHOW)
Type de fin . . . . . : RT              (LR ou RT)

```

affichage fenêtre \*MODAL.

Le générateur crée une variable nommée \$F1ModRes.

Ajoutez cette variable dans la liste des paramètres du programme à appeler, ainsi que les valeurs à récupérer.

```

*/BLOCK RPGPARM
* ----- RPGPARM : Program parameters
C      *ENTRY          PLIST
C              PARM                      $F1ModRes
C              PARM                      PIDAGE          5 0

```

Le générateur génère le code suivant :

```

c              Eval      $F1ModRes = sdShowModal (F1)

```

Sur l'événement ou l'utilisateur fait son choix (double clic sur un sfl, ou bouton), modifiez la propriété modalresult de la fenêtre.

```

*/EVENT btnOk_OnClick
* -----
* Description :
* -----
DRow          s          10i 0
DWIDAGE       s          5 0
DWdate        s          8 0
C              eval      Row = sdGetInt(F1:'sfl1':
C              'RowSelected')
C              if        Row > 0
C              eval      WIdAge=sdGetCellNum(F1:'sfl1':
C              'IDAGE':row)
C              eval      PIDAGE=WIDAGE
C              callp     sdSetInt(F1:'*FORM':
C              'ModalResult':MrOk)
C              endif

```

Dans le programme appelant, utilisez le paramètre ModRes, pour savoir si l'utilisateur a effectué un choix ou est sorti de la fenêtre modale en cliquant sur la croix en haut à droite.

```

DpmodRes      s          10u 0
DNewIDAGE     s          5 0
DNbAges       s          10 0
C      K1      klist
C              kfld          WUSER
C              kfld          WIDAGE
C              call          'SVCHXAGES'
C              parm          PmodRes
C              parm          NBAGES
C              if            PModRes=MrOk

```

**Remarque:** Si un sous fichier doit être chargé dans une fenêtre modale, appelez le chargement dans le moment BEFORESHOW. Le moment AFTERSHOW a lieu une fois que la fenêtre est fermée.

**Remarque :** il est possible d'ajouter un bouton dont la propriété modalResult est à mrCancel. Ce bouton fermera automatiquement la fenêtre modale et dans le programme appelant, vous récuperez le fait que l'utilisateur a annulé.

---

## Contrôle de la compilation

Au moment de la compilation d'un écran, le serveur de silverdev recherche l'exit program associé à l'exit point SVDCTRCPLSDF.

Pour ajouter un exit program à un exit point, utilisez la commande WRKREGINF.

Il s'agit d'un programme dont les paramètres doivent être comme ci dessous :

PGM	PARM(&USER &LIB &USRSPC &RET &TEXTRET)
DCL	VAR(&USER) TYPE(*CHAR) LEN(10)
DCL	VAR(&LIB) TYPE(*CHAR) LEN(10)
DCL	VAR(&USRSPC) TYPE(*CHAR) LEN(10)
DCL	VAR(&RET) TYPE(*CHAR) LEN(1)
DCL	VAR(&TEXTRET) TYPE(*CHAR) LEN(256)

Le paramètre User est le profil connecté.

Le paramètre Lib est la bibliothèque où l'écran va être compilé.

Le paramètre Usrspc est le nom de l'écran qui va être compilé

Le paramètre Ret doit renvoyer Y ou N (N, la compilation sera bloquée)

Le paramètre TextRet est un texte qui sera affiché dans le designer dans le cas ou Ret est à N

---

## Raccourcis claviers

Les applications 5250 offrent la possibilité d'utiliser des raccourcis tels que F1,F2 etc..

Vous serez sans doute tenté d'offrir à vos utilisateurs de tels raccourcis dans vos applications.

La meilleure solution est d'utiliser un composant TAction.

Ce composant possède une propriété nommée shortcut.

Il s'agit de la combinaison de touches qui déclenche l'action.

L'utilisation du raccourci déclenche l'évènement onexecute.

## Modification d'une colonne de sous fichier

Pour modifier une propriété d'une colonne de sous fichier à l'exécution, prenez exemple sur le code suivant :

```
C          callp      sdSetBool(F1:'sf11':
C                      'colByName('Titre').ColumnField.ReadOnly':
C                      *ON)
```

ou encore :

```
C          callp      sdSetBool(F1:'sf11':
C                      sdCol('Titre')+'.ColumnField.ReadOnly':
C                      *ON)
```

## Ajustement de la taille des composants

Si vous laissez à l'utilisateur la possibilité d'agrandir les fenêtres, l'emplacement et la taille des composants ne conviendront plus.

Exemple :

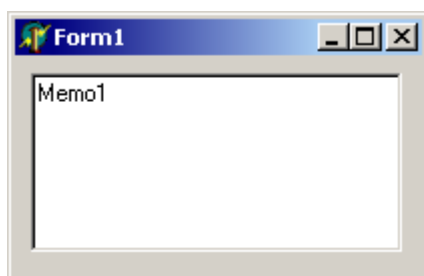


Figure 10

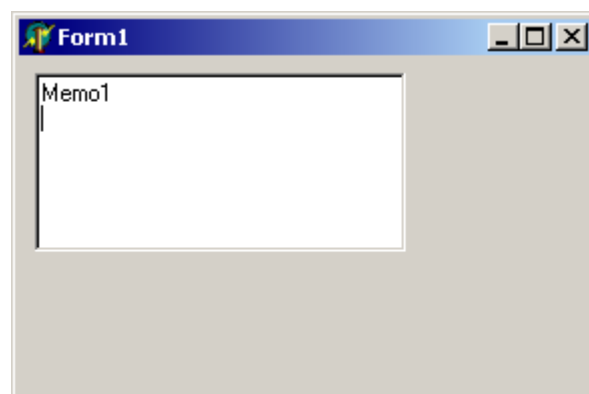


Figure 11

### Align

Cette propriété commune à tous les composants prend les valeurs suivantes :

alClient	Le contrôle remplit la zone client de son parent. Si un autre contrôle occupe déjà une partie de cette zone, le contrôle nouveau se redimensionne pour prendre le reste de la zone.
alLeft	Le contrôle se place sur le bord gauche de son parent et prend toute la hauteur de son parent. Sa largeur n'est pas modifiée.
alRight	Le contrôle se place sur le bord droit de son parent et prend toute la hauteur de son parent. Sa largeur n'est pas modifiée.
alBottom	Le contrôle se place en bas de son parent et prend toute la largeur de son parent. Sa hauteur n'est pas modifiée.



alTop	Le contrôle se place en haut de son parent et prend toute la largeur de son parent. Sa hauteur n'est pas modifiée.
alNone	Le contrôle reste à l'emplacement où il a été mis. Valeur par défaut.

Remarque :

*Il est tout à fait possible d'avoir plusieurs composants avec la propriété alLeft.(par exemple)*

*Ils seront alors les uns à côté des autres. Dans ce cas, utilisez le composant CSplitter dans l'onglet Supplément pour laisser la possibilité à l'utilisateur de modifier lui même la largeur des composants.*

**Anchors**

Si vous souhaitez que votre composant rest à 1cm du bord quel que soit la taille de la fenêtre, la propriété align ne vous convient plus.

Dans ce cas, utilisez la propriété anchors.

La propriété anchors est un ensemble qui peut inclure les 4 valeurs akLeft,akRight,akTop,akBottom.

Par défaut, les composants incluent les valeurs akLeft et akTop.

Cela signifie que les composants restent toujours à la même distance du haut et de la partie droite de leur contrôle parent.

Si vous souhaitez que des boutons restent en bas à droite d'une fenêtre, vous devez donc mettre la propriété anchors à [akRight,akBottom]

Si on inclut toutes les valeurs, le composant s'agrandit en même temps que la fenêtre.

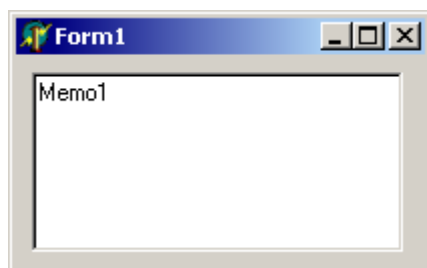


Figure 12

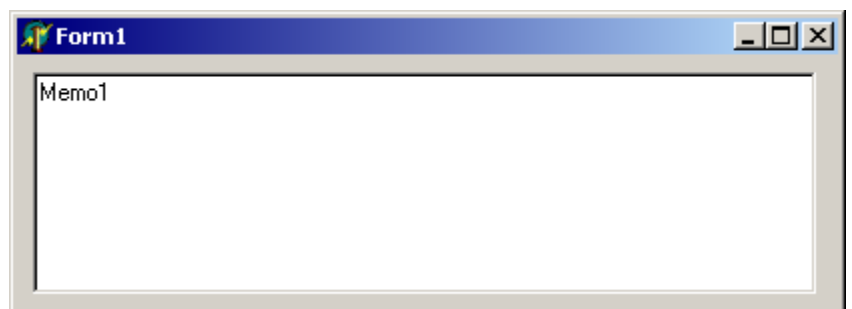


Figure 13

---

## Profil utilisateur en cours

Les travaux SilverDev fonctionnant dans un serveur TCP/IP pré démarré sous un profil standard, le profil utilisateur indiqué dans le nom de travail ne correspond pas forcément au profil réellement utilisé par le travail.

En effet au démarrage de la connexion, le profil du travail est changé (par des API Système) en celui indiqué par l'utilisateur final.

Le profil utilisateur du travail ne reflète pas cette modification.

Le profil utilisateur réellement utilisé est appelé : « **profil utilisateur en cours** » (ou Profil en cours)

Il est possible de voir ce profil par l'option 1 (Etat du travail ) de la commande DSPJOB (ou WRKJOB) :

```

                                Etat d'un travail
                                Système: EXPERI
Travail:  SDWRKBCH      Utilisateur:  QPGMR      Numéro:  646703

Etat du travail . . . . . :  ACTIF
Profil utilisateur en cours . . . . . :  JEANMICHEL
Identité de l'utilisateur du travail . . . :  JEANMICHEL
Défini par . . . . . :  *DEFAULT
Entrée dans le système:

```

Ici, bien que le nom de travail indique l'utilisateur QPGMR, le profil réellement utilisé pour la gestion des droits est JEANMICHEL

## Récupération du profil en cours dans les programmes

### RPG

La SDS (System Data Structure) permet de récupérer le nom du profil utilisateur en cours aux positions 358 à 367 :

```

d psds          sds
d  pgmStatus    *status
d  pgmProc      *proc
d  pgmRoutine   *routine
d  pgmExcpId    40      46
d  pgmLibrary   81      90
d  pgmMsgDta    91     170
d  pgmJob       244     269
d  pgmJobName   244     253
d  pgmJobUser   254     263
d  pgmJobNumber 264     269
d  pgmQJob      244     269
d  pgmCurUser  358     367

```

Attention la variable pgmJobUser renvoie le nom de l'utilisateur qui a démarré silverdev.

**CLP**

Le paramètre CURUSER de la commande RTVJOBA permet de récupérer en CLP le nom du profil utilisateur en cours :

```
RTVJOBA CURUSER(&CURUSER)
```

---

## Déclencher un évènement par programme.

Pour déclencher le onclick d'un bouton :

```
C          callp      sdMethod(F1:'Button1':'Perform':
C          'BM_CLICK':'1':'0')
```

**Remarque** : Le recours à cette méthode est sans doute le signe d'une mauvaise architecture dans vos programmes.

---

## Envoyer un message entre fenêtres

Il est possible d'envoyer un message d'une fenêtre à une autre à l'aide de la fonction sdSendMsgWin, dans le but par exemple de déclencher une procédure contenue dans un autre programme.

Programme 1 :

```
C          callp      sdSendMsgWin(F1:1:10:'REFRESH')
```

Programme 2 :

```
*/EVENT OnMsgWin
D Parameters      ds              based(pevtinf)
D Win              5u 0
D Evt              48a
  * Non modifiable
D From              5u 0
D Code              10u 0
D Datas             100      varying
C          select
C          when      Datas ='REFRESH'
C          callp      Refresh
...
```

C

endsl

*Remarque 1: Cet évènement est envoyé du serveur vers le serveur. Il ne transite pas par la partie cliente.*

## Ouvrir un document sur le serveur

Pour ouvrir un document qui se trouve sur l'ifs, copiez le dans le répertoire temp du pc et ouvrez le comme dans l'exemple suivant :

```
DPath      s      1000      varying
C          eval      Path      ='%TEMP%\test.txt'

C          eval      Path      =sdGetRealFolder(Path )
C          callp      sdDownload('/home/Doc.txt':
C                      Path)
C          callp      sdExecutePc('open':Path:
C                      ':'W_NORMAL)
```

## CSfl dynamique:

Le programme sddmsql dans silverdemo utilise des requêtes sql dynamique et crée les colonnes en fonction de la requête.

## Survol d'un sfl

L'exemple suivant permet d'afficher une texte dans un memo lors du survol d'une cellule de sous fichier. **Cet exemple utilise les évènements locaux.**

La fenetre comprend un mémo nommé memo1 et un un sfl nommé sfl1.

Le sfl comprend une colonne col1 et une colonne cachée col2.

Lors du survol de la colonne col1, on affiche dans le memo le contenu de la colonne col2.

Le code local de l'évènement onMouseMove du sfl est le suivant :

```
procedure SFL1_OnMouseMove (Sender: TObject; Shift: TShiftState; X:
Integer; Y: Integer);
var
```

```

row:integer;
col:string;
begin
  This.memo1.lines.clear();
  This.sfl1.getCellAt(x,y,row,col);
  if(row>0) and (col='col1')then
    begin
      This.memo1.lines.add(This.sfl1.getCellValue('col2',row));
    end;
  End;

```

## Eviter un appel à sdGet

Au lieu d'écrire :

D path	s	250	varying
c	eval	path = sdGet(F1:'IfsDialog1':'fileName')	
c	callp	sdSetString(F1:'edtPath':'text':path)	

Il est possible d'écrire:

c	callp	sdSetCtrl(F1:'edtPath':'text':F1:
c		'ifsDialog1.fileName')

---

## Chapitre 12. Applications multilingues

Silverdev propose une fonctionnalité pour obtenir des écrans multilingues, ainsi qu'une fonction pour rechercher un message dans un fichier de message.

---

### Propriété Langs

Sélectionnez la propriété de type collection `OptionGeneration.Langs`.

Ajoutez un élément à cette collection. (ou plusieurs si vous souhaitez traduire dans plusieurs langues).

Affectez la propriété `LangId` de cet élément. Cette propriété doit comporter trois caractères.

---

### Traduction des chaînes

Après avoir compilé l'écran, utilisez le menu Outils/Localisation.

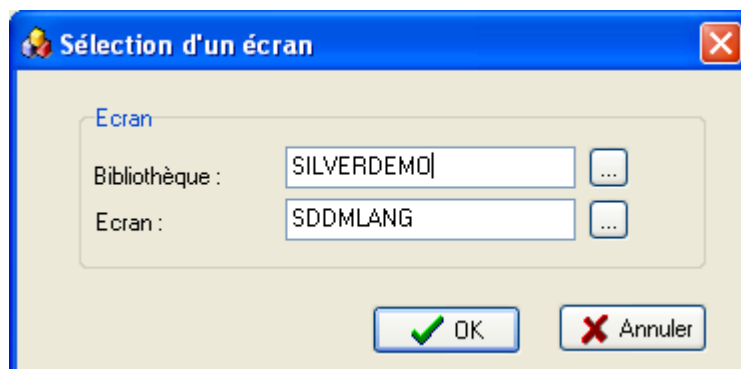


Figure 14

S'il y a plusieurs éléments dans la collection `OptionGeneration.Langs`, la liste des `langId` est affichée dans une fenêtre.

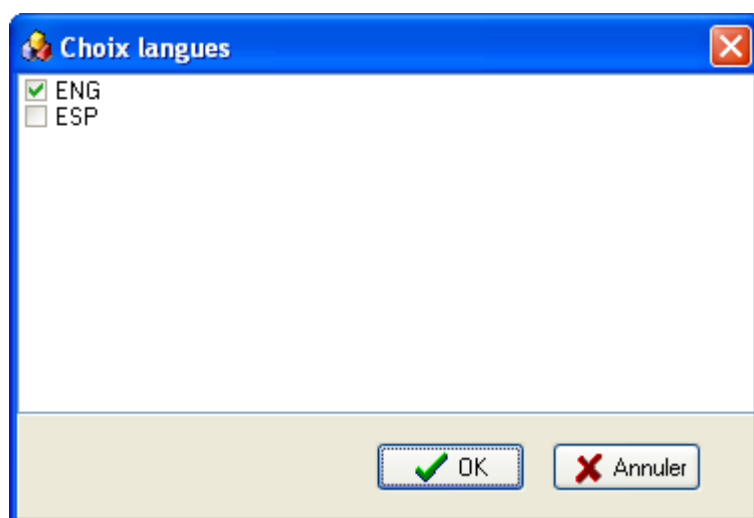


Figure 15

Cochez les langues que vous souhaitez voir apparaître dans la fenêtre suivante :

Traductions				
SILVERDEMO/SDDMLANG				
Propriété	Chaine	Traduction générale ENG	Traduction spécifique ENG	Chaine traduite ENG
Caption	Démonstration de traduction	Translation demonstration		Translation demonstration
Label1.Caption	Bonjour	Hello		Hello
ComboBox1.Items(0)	janvier	january		january
ComboBox1.Items(1)	février	february		february
ComboBox1.Items(2)	mars	march		march
ComboBox1.Items(3)	avril	april		april
ComboBox1.Items(4)	mai	may		may
ComboBox1.Items(5)	juin	june		june
ComboBox1.Items(6)	juillet	july		july
ComboBox1.Items(7)	août	august		august
ComboBox1.Items(8)	septembre	september		september
ComboBox1.Items(9)	octobre	october		october
ComboBox1.Items(10)	novembre	november		november
ComboBox1.Items(11)	décembre	december		december
ComboBox1.Text	mars	march		march
Button1.Caption	Anglais	English		English
Button2.Caption	Français	French		French
ListBox1.Items(0)	lundi	monday		monday
ListBox1.Items(1)	mardi	tuesday		tuesday
ListBox1.Items(2)	mercredi	wednesday		wednesday
ListBox1.Items(3)	jeudi	thursday		thursday
ListBox1.Items(4)	vendredi	friday		friday
ListBox1.Items(5)	samedi	saturday		saturday
ListBox1.Items(6)	dimanche	sunday		sunday

Figure 16

Les colonnes Traduction générale et traduction spécifique sont modifiables, les autres sont en lecture seule.











La colonne Propriété liste toutes les propriétés de type chaîne qui peuvent nécessiter une traduction.

Les chaînes originelles dans la fenêtre sont affichées dans la colonne Chaîne.

Les traductions sont affichées dans la colonne Chaîne traduite.

Les chaînes sont traduites automatiquement en se basant sur le fichier PSVDDLO.

## Fichier PSVDDLO

Fichier : <b>PSVDDLO</b>		Bibliothèque : <b>SILVERDEV</b>	Description : <b>Dictionnaire traduction</b>
Type : <b>PF</b>		Unique : <b>Oui</b>	
Nom	Type	Colhdg	Edit Code
<b>RSVDDLO</b>	<b>R</b>		
  1 LANGID	CHARACTER (3)	LANGID	
  2 USRSPC	CHARACTER (10)	USRSPC	
  3 USRSPCLIB	CHARACTER (10)	USRSPCLIB	
  4 PROPERTY	VAR CHARACTER (250)	PROPERTY	
  5 STRING	VAR CHARACTER (250)	STRING	
TRANSLAT	VAR CHARACTER (250)	TRANSLAT	

**Figure 17**

La chaîne à traduire est recherchée en priorité avec les champs usrspc, usrspclib et property correspondant aux valeurs de l'écran, puis avec ces valeurs à blanc.

Les deux colonnes suivantes sont en saisie, elle permettent de modifier le fichier psvddlo.


La colonne traduction générale renseigne le fichier psvddlo avec les champs usrspc, usrspclib et property à blanc. Ces valeurs seront donc utilisées pour tous les écrans.

La colonne traduction spécifique renseigne le fichier psvddlo avec les champs usrspc, usrspclib, et property prenant les valeurs de l'écran en cours. Ces valeurs seront donc utilisées uniquement pour cet écran.

Après avoir modifié le fichier psvddlo, il faut recompiler l'écran.

Le bouton  permet de recompiler l'écran.



Le bouton  permet de change de langue.

## Modifier la langue à l'exécution.

Pour modifier langue d'un 'écran pendant l'exécution, utilisez la fonction sdTranslate.

```
D sdTranslate      pr
D F1              5u 0 const
D LangId          3      value
```

Le paramètre LangId correspond à la valeur renseignée dans la collection Langs de la fenêtre.

```
c                  callp      sdTranslate(F1: 'ENG')
```

Pour revenir à la langue d'origine passez \*blanks au paramètre langId

Pour modifier tous les écrans, utilisez la fonction sdTranslateAll.

```
D sdTranslateAll  pr
D LangId          3      value
```

Tous les écrans de l'application seront alors traduits. Les écrans créés après l'appel à sdTranslateAll seront traduit automatiquement.

Attention : Si vous utilisez sdTranslateAll, il est plus prudent de l'appeler au début du programme. En effet, si la valeur d'une chaîne a été changé par programme, ce n'est pas la nouvelle chaîne qui sera traduite, mais la chaîne affectée lors de la création de l'écran.

## Déplacement d'un écran

Si vous renommez un écran, ou le déplacez un écran d'une bibliothèque à une autre, les informations de traduction sont conservées, mais lors de la prochaine compilation, les informations peuvent ne pas être trouvées dans psvddlo. Pensez alors à modifier ou copier les enregistrements de PSVDDLO.

Exemples :

Ecran déplacé de la bibliothèque SILVERDEMO vers la bibliothèque DEMO

```
UPDATE SILVERDEV/PSVDDLO SET USRSPCLIB = 'DEMO' WHERE USRSPC
```

= 'SDDMLANG' and USRSPCLIB ='SILVERDEMO'

Ecran copié de la bibliothèque SILVERDEMO vers la bibliothèque DEMO

```
INSERT INTO PSVDDLO (LANGID, USRSPC, USRSPCLIB, PROPERTY,
STRING, TRANSLAT) SELECT LANGID, USRSPC,'DEMO', PROPERTY,
STRING, TRANSLAT FROM PSVDDLO WHERE USRSPC= 'SDDMLANG' and
USRSPCLIB='SILVERDEMO'
```

---

## Menu traduction

En effectuant un click droit dans la barre des tâche, l'utilisateur accède au menu traduction.

La liste des toutes les traduction possibles est alors affichée.

Lorsque l'utilisateur sélectionne une langue, le résultat est identique à l'appel de sdTranslateAll.

---

## SdRtvMsg

### Prototype :

```
d sdRtvMsg...
d                                pr          1000a    varying
d  msgfile                      20a    const
d  msgid                        7a    const
d  msgdta                      1000a    const options(*varsize:*nopass)
d  msgdtalen                   10i 0    const options(*nopass)
```

### Description :

La fonction sdRtvMsg permet d'obtenir le texte de 1<sup>er</sup> niveau d'un message, en utilisant éventuellement des données de substitution. La fonction utilise l'API QMHRTVM à cet effet.

Le paramètre MsgFile, obligatoire, est un nom d'objet qualifié, nom d'objet sur 10 caractères puis nom de bibliothèque sur 10 caractères, représentant un fichier de message (type \*MSGF). SdRtvMsg, parce-qu'elle utilise QMHRTVM est susceptible de retourner un message provenant d'un autre fichier de message si une substitution de fichier de message (OVRMSGF) est active. Les valeurs spéciales \*CURLIB et \*LIBL peuvent être utilisées comme nom de bibliothèque.

Le paramètre MsgId, obligatoire, précise l'identificateur de message à extraire.

Le paramètre `MsgDta`, facultatif, contient les valeurs de substitution pour le message. Le format de ce paramètre dépend du message extrait.

Le paramètre `MsgDtaLen`, facultatif, indique la longueur du paramètre `MsgDta`. Si ce paramètre n'est pas passé ou que sa valeur est égale à zéro, le paramètre `MsgDta` n'est pas pris en compte.

La valeur retournée contient le texte de 1<sup>er</sup> niveau formaté

Si l'identificateur de message n'est pas trouvé, la valeur retournée contient le texte de 1<sup>er</sup> niveau du message CPF2419 de QSYS/QCPFMSG.

Toute autre erreur rencontrée lors de l'appel de `QMHRTVM` est retournée à l'appelant sous la forme d'une exception.

## Chapitre 13. Fonctions catégorie images

### sdSetImg

Pour charger une image depuis l'ifs vers un composant pouvant recevoir une image (CImage, CBitBtn, CSpeedButton..)

#### Prototype :

d sdSetImg	pr	10i 0	
d form			like (tWHandle)
d component		30	varying value
d propertyPath		256a	varying value
d TypeImage		3	value
d path		256a	varying value

#### Description

La fonction sdSetImg permet de charger une image dans un objet de type TBitmap , TPicture ou TIcon

TypeImage prend les valeurs : 'bmp', 'jpg' ou 'ico' .

Le fichier indiqué par le paramètre path doit être du type indiqué par le paramètre TypeImage.

Type de l'objet Valeurs possibles de TypeImage

TBitmap 'bmp'

TIcon 'ico'

TPicture 'bmp', 'jpg' ou 'ico'

Paramètres :

#### Valeurs de retour :

0 : Tout s'est bien passé

-1: Le fichier spécifié n'existe pas ou vous n'avez pas les droits. L'image n'est pas chargée.

-2: Le fichier envoyé ne semble pas être du type indiqué ou le composant ne peut pas charger ce type d'image.

(Vous envoyez une image dont le format ne correspond pas au format indiqué par le paramètre TypeImage)

### Paramètres

Component : Composant contenant l'image

Propertypath : Propriété du composant capable de contenir une image. Cette propriété doit être du type TBitmap , TPicture ou TIcon.

TypeImage :Indique le type de l'image qui est envoyée.

path : Chemin sur l'ifs de l'image à charger

### Exemple 1 :

```
C          callp      sdSetImg (form:'BitBtn1':'Glyph':
C          'bmp':'/Images/test.bmp')
```

### Exemple 2 :

```
PLoadImage      B

D                PI
DNumLivre        4  0
DNomImage        s      1000    varying
D retour        s      10i 0
C                eval      NomImage='/Home/Images/'+
C                SdIntToStr(NumLivre)+' .jpg'
C                eval      Retour =
C                sdSetImg(F1:'Image1':
C
C                'Picture':'jpg':NomImage)
*Si le chargement ne s'est pas bien passé
*On supprime l'image qui pouvait être dans le composant
C                if      retour <> 0
C                callp      sdSet      (F1:'Image1':
C                'Picture.graphic':'Null')
C                endif
```

### Exemple 3 :

Chargement d'une icône dans une fiche :

```
C          callp      sdSetImg(form:'*FORM':'Icon':
C          'ico':'/Images/test.ico')
```

**Exemple 4 :**

Chargement d'un bmp dans un CImage:

```
C          callp      sdSetImg (form:'Image1':'picture.Bitmap':
C          'bmp':'/Images /test.bmp')
```

ou

```
C          callp      sdSetImg (form:'Image1':'picture':
C          'bmp':'/Images /test.bmp')
```

**Exemple 5 :**

Chargement d'un jpg dans un CImage:

```
C          callp      sdSetImg (form:'Image1':'picture':
C          'jpg':'/Images /test.bmp')
```

---

## sdGetImg

**Prototype :**

```
d sdGetImg      pr          10i 0
d pform          5u 0
d component      30a  varying value
d propertyPath   256a  varying value
d path           256a  varying value
```

**Description :**

Permet de récupérer une image qui est stockée dans un composant et de la sauvegarder sur un fichier de l'ifs.

Le paramètre propertyPath est une propriété qui doit être du type TBitmap , TPicture ou TIcon

**Valeur de retour :**

0 : Tout s'est bien passé

-1 : L'image dans le composant est vide, et donc le fichier sur l'ifs n'a pas été créé/modifié.

-2 : Le nom du fichier ifs est incorrect ou vous n'avez pas les droits suffisants

**Paramètres :**

pForm :Identifiant de la fenêtre contenant le composant "component".

Component : Composant contenant l'image.

propertyPath : Propriété de type TPicture, TBitmap ou TIcon contenant l'image.

path :Chemin sur l'ifs pour stocker l'image récupérée

**Exemple :**

```
PWriteImage      B
D                PI
D Retour         s          10i 0
D NameImg        s          500    varying
C                eval      nameImg = '/Home/Images/' +
C                sdIntToStr(IDLIVRE)+' .jpg'
C                eval      Retour =
C                sdGetImg(F1: 'Image1':
C                'picture':NameImg)
C                *Si l'image est vide et qu'il y a un fichier sur l'ifs
C                *pour cet enregistrement, on le supprime
C                if        (retour = -1) and access(NameImg:W_OK)=0
C                callp     UnLink(NameImg)
C                endif
```

---

**sdLoadFromFile****Prototype :**

```
d sdLoadFromFile pr          10i 0
d pform          5u 0 const
d Component      30    varying value
d FileName       1000   varying value
```

**Description :**

Permet de charger une image depuis le disque du client.

S'applique aux objets du type TPicture , TStringrs

**Exemple :**

```
PSelectImg      B
D                PI
```

```

D path          s          1000    varying
C              if          sdSelectFile('*.*jpg |*.jpg':
C                      Path:seOpenPicture)
C              callp       sdLoadFromFile(F1:'Image1.picture':
C                      path)
C              endif
P              E

```

## sdAssignTo

### Prototype

```

d sdAssignTo      pr
d pform1          5u 0
d Object1         50a  varying value
d pform2          5u 0
d Object2         50a  varying value

```

### Description :

Copie le contenu de l'objet Object1 dans l'objet Object2. Object1 et Object2 doivent être de même type.

### Exemple :

Copie de l'objet TPicture d'un CImage dans un autre.

```

PEvtGdeImg      B
D              PI
* -- Insert your code here ...
C              if          FicheImg = 0
C              eval        FicheImg =sdCreateform('*LIBL/FICHEIMG')
C              endif
C              callp       sdAssignTo(Fiche:'Image1.picture':
C                      FicheImg:'Image1.picture')
C              callp       sdShowModal(FicheImg)
P              E

```

*Remarque :*

*Cette fonction peut s'appliquer aux objets de type suivant :*

*TChartSeries, TBitmap, TBrush, TFont, TIcon, TListColumn,  
TListColumns, TListItems, TNode, TNodes, TPicture, TString*



---

## sdAddImg

### Prototype :

d	sdAddImg	pr	10i	0
d	form			like(tWHandle) const
d	component		30	varying value
d	TypeImage		3	value
d	path		256a	varying value

### Description :

Permet d'ajouter une image dans un composant CReplImage  
Ou dans un composant CimageList.

### Exemple :

C	callp	sdAddImg(F2:'Img1':'jpg':
C		BaseImage+'Img'+%char(A_IDLIVRE)+' .jpg')

### Valeurs de retour :

0 : Tout s'est bien passé

-1: Le fichier spécifié n'existe pas ou vous n'avez pas les droits. L'image n'est pas chargée.

-2: Le fichier envoyé ne semble pas être du type indiqué, ou le composant ne peut pas chargé ce type d'image.

(Vous envoyez une image dont le format ne correspond pas au format indiqué par le paramètre TypeImage)

# Chapitre 14. Fonctions boites de dialogue

## SdDialog

### Prototype :

```
d sdSaveToFile      pr          N
d pform             5u 0 const
d Component         30      varying value
```

### Description :

La fonction sdDialog permet d'afficher une boite de dialogue.  
La fonction renvoie \*on si l'utilisateur a cliqué sur OK.

Cette fonction s'applique aux composants suivants :  
CfontDialog, CcolorDialog, COpenDialog, CsaveDialog,  
COpenPictureDialog, CSavePictureDialog.

### Exemple :

```
DName      s          200      varying
C          if          sdDialog(f1:'fontdialog1')
C          eval        name=sdget(f1:'fontdialog1':'font.name')
C          callp       sdshowmessage(name)
C          endif
```

```
DCouleur   s          10i 0
C          if          sdDialog(f1:'ColorDialog1')
C          eval        Couleur= sdGetInt(f1:'ColorDialog1':'Color')
C          callp       sdshowmessage(%char(Couleur))
C          endif
```

## sdShowMessage

Si vous souhaitez simplement afficher un message dans une fiche modale, il n'est pas nécessaire de créer une fiche pour cela.  
Utilisez plutôt la fonction sdShowMessage.

Cette fonction fait appel à une boîte de dialogue du système d'exploitation.  
La taille de la boîte de dialogue s'adapte au texte que vous souhaitez afficher.

Pour afficher un message sur plusieurs lignes, ajoutez le code x'0d' dans le texte pour chaque saut de ligne.

---

## sdMsgDlg

Si vous souhaitez créer une fiche de dialogue avec l'utilisateur deux solutions s'offrent à vous.

### Première solution :

Créez une fiche.

Ajoutez y un label et des boutons ayant différentes valeurs de modalResult.

Afficher cette fiche en modal et interrogez la propriété modalResult de la fiche pour savoir sur quel bouton l'utilisateur a appuyé.

### Deuxième solution :

Utilisez la fonction sdMsgDlg .

Cette fonction prend plusieurs paramètres.

Ces paramètres vous permettront de modifier le texte contenu dans la boîte de dialogue, d'indiquer les boutons que vous souhaitez y voir apparaître et éventuellement d'y ajouter une icône.

Cette fonction renvoie une valeur. il est ainsi possible de savoir sur quel bouton l'utilisateur a cliqué.

*Remarque : La fonction sdMsgDlg peut être utilisée à la place de la fonction sdShowMessage en ne mettant qu'un seul bouton.*

### Prototype :

d sdMsgDlg	pr	4	0
DIcône		2	0 value
DBoutons		4	0 value
dTexte		3000	varying value

### Paramètres :

**Valeur de retour :** Permet de connaître le bouton sur lequel l'utilisateur a appuyé.

**Boutons :** Boutons affichés dans la boîte de dialogue

Valeurs possibles de Boutons et de la valeur de retour :

Valeurs prédéfinies dans H,SILVERDEV :

DBtnNONE	C	0
DBtnOK	C	1
DBtnCnl	C	2
DBtnAbort	C	4
DBtnRetry	C	8
DBtnIgnore	C	16
DBtnYes	C	32
DBtnNo	C	64
DBtnAll	C	128
DBtnNoToAll	C	256
DBtnYesToAll	C	512

**Icone** : Image insérée dans la boite de dialogue.

### Valeurs possibles de Icone :

Valeurs prédéfinies dans H,SILVERDEV :

DBoxWarning	C	0
DBoxError	C	1
DBoxInfo	C	2
DBoxConfirm	C	3
DBoxNone	C	4

**Texte** : Texte affiché dans la boite de dialogue.

Pour afficher plusieurs boutons, il suffit d'additionner les valeurs désirées.

### Exemple :

C	if	sdMsgDlg(boxConfirm:btnOK + btnCnl:
C		'Confirmez vous la suppression ? ') = btnOK
C	endif	

---

## sdSelectFile

Remarque :

Préférez l'utilisation de la fonction `sdDialog` avec le composant `COpenDialog` ou `CSaveDialog`.

### Prototype :

<code>d sdSelectFile</code>	<code>pr</code>	<code>N</code>	
<code>DFilter</code>		<code>1000</code>	<code>varying value</code>
<code>DFileName</code>		<code>1000</code>	<code>varying</code>
<code>DTypeSelect</code>		<code>1</code>	<code>0 value</code>
<code>DInitialDir</code>		<code>1000</code>	<code>varying value options(*nopass)</code>

### Description :

Cette fonction ouvre une boîte de dialogue qui permet à l'utilisateur de sélectionner un fichier sur son disque dur.

### Valeur de retour :

Le booléen renvoyé est à \*ON si l'utilisateur a sélectionné un fichier, et à \*OFF si l'utilisateur a abandonné la sélection de fichier.

### Paramètres :

#### Filter :

Détermine les masques de fichiers (filtres) disponibles dans la boîte de dialogue. La boîte de dialogue de sélection de fichier inclut une liste déroulante de types de fichiers sous la boîte de saisie. Quand l'utilisateur sélectionne un type de fichier dans cette liste, seuls les fichiers de ce type sont présentés dans la boîte de dialogue.

Affectez à Filter une valeur composée d'une description et d'un masque, séparés par un caractère barre verticale. La barre verticale ne doit pas être encadrée d'espaces. Par exemple : `'Fichiers texte (*.txt)|*.TXT'`

Plusieurs filtres doivent être séparés par plusieurs barres verticales. Par exemple :

```
'Fichiers texte (*.txt)|*.TXT|Fichiers RTF(*.rtf)|*.RTF'
```

Pour inclure plusieurs masques dans un seul filtre, séparez les masques par des points-virgules.

Par exemple :

```
' Fichiers images|*.BMP;*.JPG;*.ICO'
```

Si aucune valeur n'est affectée à Filter, la boîte de dialogue affiche tous les types de fichiers.

### FileName :

Nom du fichier sélectionné par l'utilisateur. (Vérifiez que l'utilisateur a bien sélectionné un fichier en testant la valeur de retour)

### TypeSelect :

Détermine le type de boîte de dialogue.

### Valeurs possibles :

DSeOpen	C	0
DSeSave	C	1
DseOpenPicture	C	2
DseSavePicture	C	3

### InitialDir:

Ce paramètre est optionnel, il permet de préciser le répertoire initial affiché dans la boîte de dialogue.

### Exemple :

```
PSelectImg      B
D               PI
D path          s      1000      varying
C               if      sdSelectFile('*.jpg |*.jpg':
C                   Path:seOpenPicture)
C               callp    sdLoadFromFile(F1:'Image1.picture':
C                   path)
C               endif
P               E
```

---

## sdSelectDir

### Prototype :

D sdSelectDir	pr	N	
DFileName		1000	varying
DInitialDir		1000	varying value options(*nopass)

### Remarque :

Vous pouvez créer votre propre fonction sdSelectDir en utilisant le composant CShellTreeView de l'onglet système.

---

## sdPrintDlg

### Remarque :

Préférez l'utilisation de la fonction sdDialog avec le composant CPrintDialog.

Cette fonction permet d'ouvrir une boîte de dialogue de sélection de l'imprimante.

### Prototype :

d sdprintDlg	pr	N
--------------	----	---

### Exemple :

C	if	sdPrintDlg
C	callp	sdPrint(F2: 'Rep1')
C	endif	

---

## sdSelectColor

### Remarque :

Préférez l'utilisation de la fonction sdDialog avec le composant CColorDialog.

Cette fonction permet d'ouvrir une boîte de dialogue pour choisir une couleur.

Prototype :

d sdSelectColor	pr	N
DColor		10i 0

Paramètres :Valeur de retour :

Le booléen renvoyé est à \*ON si l'utilisateur a sélectionné une couleur, et à \*OFF si l'utilisateur a abandonné la sélection de couleur

Color :

Valeur de la couleur sélectionnée par l'utilisateur.  
Vaut 0 si l'utilisateur n'a pas choisi une couleur.

Exemple :

```
pButton1_OnClick...
p          B
d          PI
d PevtInf          *   const options(*nopass)
DCouleur          s          10i 0
C          if          sdSelectColor(Couleur)
C          callp          sdsetInt(F1:'edit1':'color':Couleur)
C          endif
p          E
```

Remarque :

*Les fonctions sdSelectColoret sdPrintDlg sont devenues obsolètes depuis la création de la fonction sdDialog.*



# Chapitre 15. Fonctions catégorie fichiers PC

Les fonctions suivantes permettent de manipuler les fichiers côté client.

## sdGetDir

### Prototype

```
d sdgetDir      pr      *
d Dir           256     varying value
```

### Description

La fonction sdGetDir permet de lister les fichiers correspondant à un masque dans un dossier sur le poste client.

L'utilisation de la fonction sdGetListLevel permet de connaître la taille du fichier.

L'utilisation de la fonction sdGetListState permet de connaître le type de fichier ou de dossier :

Valeurs possibles pour le type de fichier :

1	fichier en lecture seule
2	fichier caché
4	fichier système
8	fichier volume id
16	répertoire
32	archive
64	fichier

Les valeurs peuvent être combinées. Exemple : 65=64 + 1 signifie fichier en lecture seule

Exemple :

```
Dlist      s      *
D i         s      10i 0
D Size      s      10u 0
D Type      s      5u 0
```

```

D Name          s          50      varying
C              eval      list = sdgetdir('c:/*')

C              for        i=0 to sdgetlistcount(list)-1
C              eval      Name = sdgetlistLabel(list:i)
C              eval      Type = sdgetliststate(list:i)
C              eval      Size = sdgetlistLevel(list:i)
C              endfor
C              callp      sdFreelist(list)

```

---

## SdDelFile

### Prototype :

d sdDelFile	pr	10u 0
d Dir		1000 Varying Value

### Description :

La fonction sdDelFile permet de supprimer un fichier sur le poste du client.

### Valeur de retour :

- 0 : Le fichier a été supprimé.
- 1 : Le fichier n'existe pas
- 2 : Le fichier existe mais n'a pas pu être supprimé

---

## sdDownload

### Prototype :

d sdDownload	pr	10i 0
d pathAs		1000a varying value
d pathPC		1000a varying value
d Erase		N Options( *nopass) Value

### Description :

Permet de copier un fichier de l'ifs vers le pc.

### Paramètres :

Le paramètre Erase permet de forcer l'écrasement du fichier sur le pc si celui ci existe.

Si Erase est à \*OFF ou s'il est omis et que le fichier existe sur le pc, une boîte de dialogue demande à l'utilisateur une confirmation avant d'écraser le fichier existant.

### Valeur de Retour :

0 : Tout est ok.

-1 : Le fichier n'existe pas sur l'ifs.

-2 : Un fichier du même nom existe déjà et l'utilisateur a refusé de l'écraser

-3 : Un problème est survenu au moment de la sauvegarde

#### Remarque :

*Si un fichier du même nom existe déjà sur le pc du client, une boîte de dialogue lui demande confirmation pour écraser ce fichier. Si l'utilisateur renonce, le code de retour de la fonction est -2.*

### Exemple :

```
*/EVENT Button1_OnClick
D pathPc      s      1000    varying
D pathAs      s      1000    varying
D retour      s      10i 0

C              eval      pathAs=sdget(f1:'edit1':'text')
C              if        sdSelectFile('*. * |*. *':
C                          PathPc:seSave)
C              eval      Retour = sdDownload(pathAs:
C                          pathPc)
C              endif
```

## SdForceDir

### Prototype :

```
d sdForceDir    pr      10u 0
d Dir           1000    Varying Value
```

### Description :

La fonction sdForceDir permet de créer un répertoire sur le poste du client.

### Valeur de retour :

- 0 : Le répertoire a été créé.  
1 : Le répertoire n'a pas été créé.

---

## SdGetFileSize

### Prototype :

D	sdGetFileSize	pr	10u	0
D	FilePath		1000	Varying Value

### Description :

Permet d'interroger la taille d'un fichier sur le poste client.

### Exemple :

C	eval	Taille=sdGetFileSize (FileFrom)
---	------	---------------------------------

---

## SdSaveToFile

### Prototype :

d	sdSaveToFile	pr	10u	0
d	pform		5u	0 const
d	Component		30	varying value
D	File		1000	Varying Value

### Description :

Permet de sauvegarder une image ou le contenu d'un TStrings dans un fichier.

### Valeurs de retour :

- 0 : Tout s'est bien passé  
1 :Le composant passé en paramètre ne semble pas gérer cette fonction  
2 :La sauvegarde n'a pas réussi (le fichier est peut être occupé)  
3 :Le répertoire n'existe pas

### Exemple :

```
C          callp      sdSaveToFile(f1:'memo1.lines':
C          'c:/test.txt')
```

```
C          callp      sdSaveToFile(f1:'Image1.picture':
C          'c:/test.bmp')
```

```
C          callp      sdSaveToFile(f1:'Chart1':
C          'c:/test. bmp ')
```

---

## SdSelectFile

Voir chapitre sdSelectFile

---

## SdSetXFerBar

### Prototype :

```
d sdSetXFerBar      pr
d pform              5u 0 const options(*nopass)
d Component          30    varying value options(*nopass)
```

### Description :

Permet de désigner un composant de type CProgressBar qui matérialisera l'avancement des fonction sdUpload et sdDownload.

#### Remarque :

Pour supprimer l'affectation, appelez la fonction sans paramètres.

### Exemple :

```
C          callp      sdSetXFerBar(F1:'Progressbar1')
...
C          callp      sdSetXFerBar
```

---

## SdUpload

### Prototype :

d sdUpLoad	pr	10i 0	
d pathPC		1000a	varying value
d pathAs		1000a	varying value

**Description :**

Permet de copier un fichier du pc du client vers l'ifs.

**Paramètres :****Valeur de Retour :**

0 : Tout est ok.

-1 : Le fichier pc n'existe pas.

-2 : Le nom du fichier ifs est incorrect ou vous n'avez pas les droits suffisants.

-3 : Le fichier PC semble occupé.

**Exemple :**

```

P EvtUpload          B
D                    PI
D FichierPC          s      1000    varying
D FichierAs          s      1000    varying
D Retour             s      10i 0

C                    eval      FichierAs='/home/images/testUpload.jpg'
C                    if        sdSelectFile('*.*jpg |*.jpg':
C                               FichierPc:seOpenPicture)
C                    eval      retour =sdUpload(FichierPc:FichierAs)
...
C                    endif
P                    E

```

---

## SdSelectDir

Voir sdSelectDir

---

## SdFileExists

Prototype :

d	sdFileExists	pr	N	
d	File		1000	Varying Value

---

**SdDirExist**Prototype :

d	sdDirExists	pr	N	
d	Dir		1000	Varying Value

---

**sdCopyFile**Prototype :

d	sdRenameFile	pr	N	
D	OldFile		1000	Varying Value
D	NewFile		1000	Varying Value

Description :

La fonction sdCopyFile permet de copier un fichier du pc.

Valeur de retour :

\*On : La fonction a réussi.

\*OFF : La fonction a échoué (OldFile inexistant ou occupé, répertoire de destination inexistant, NewFile existe déjà etc..)

Exemple :

D	ret	S	N	
C		eval		ret=sdCopyFile('c:/test.txt':'c:/toto/'
C				'test.txt')

---

**SdGetRealFolder**Prototype :

d	sdGetRealFolder...		
d		pr	1000 Varying

d File

1000

Varying Value

**Description :**

La fonction sdGetRealFolder renvoie le nom réel d'un fichier à partir d'une expression conteant une variable d'environnement telle que %TEMP%

**Exemple :**

DPath	s	1000	varying
C	eval	Path	='%TEMP%\test.txt'
C	eval	Path	=sdGetRealFolder(Path )
C	callp	sdDownload('/home/Doc.txt':	
C		Path)	
C	callp	sdExecutePc('open':Path:	
C		'':W_NORMAL)	

Pour connaitre les variables systèmes, click droit sur le poste de travail :



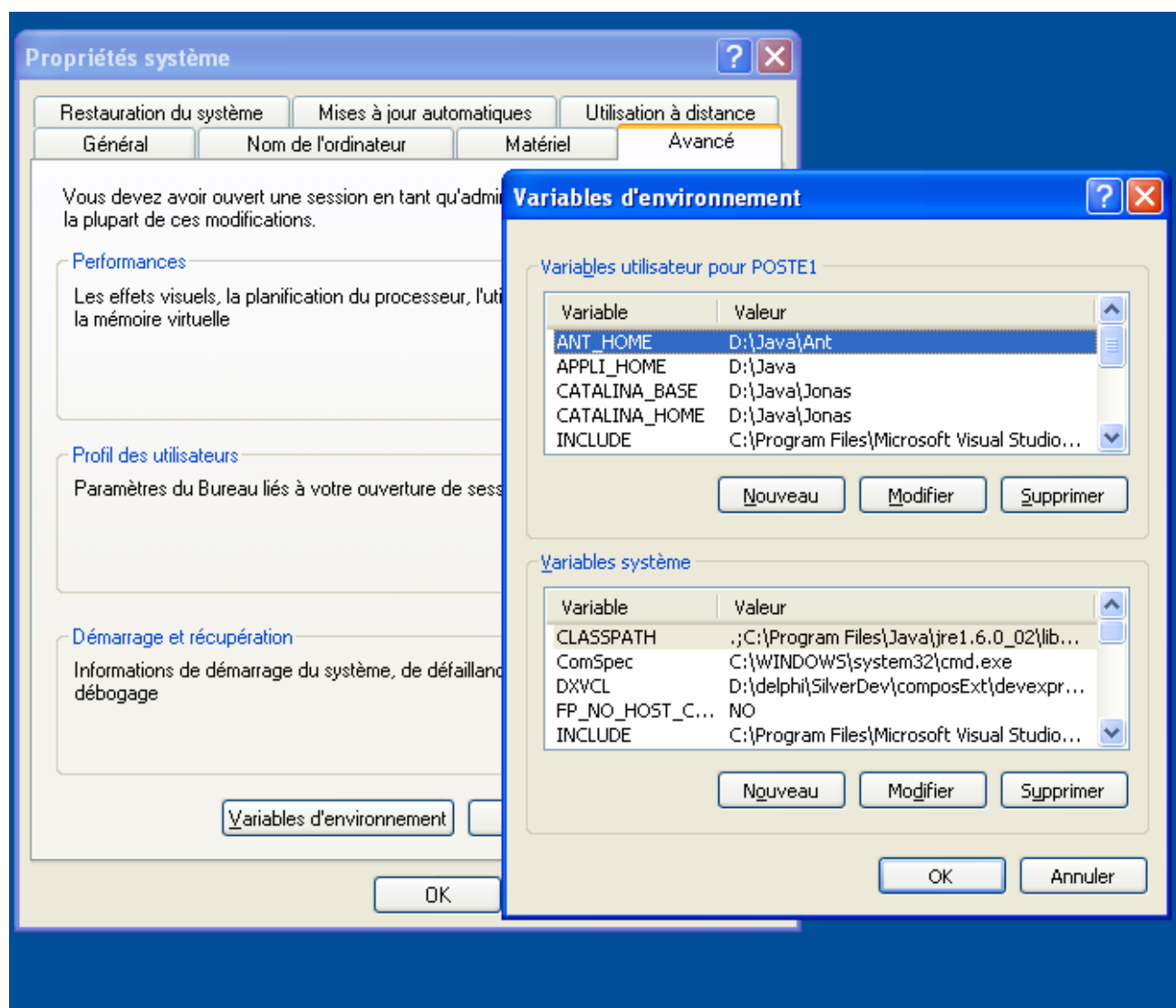


Figure 18

## SdRenameFile

### Prototype :

```

d sdRenameFile      pr                      N
D OldFile           1000      Varying Value
D NewFile           1000      Varying Value

```

### Description :

La fonction sdRenameFile permet de renommer un fichier sur le pc.

**Valeur de retour :**

\*On : La fonction a réussi.

\*OFF : La fonction a échoué (OldFile inexistant ou occupé, répertoire de destination inexistant, NewFile existe déjà etc..)

**Exemple :**

D	ret	S	N
C		eval	ret=sdRenameFile('c:/test.txt':'c:/toto/'
C			'test.txt')

---

# Chapitre 16. Fonctions catégorie collections

Les propriétés CDateEdit.ValidDates, CSFL.columns ou CSFL.ExAttributes sont des collections.

Les fonctions suivantes permettent de manipuler les collections par programme.

---

## sdCollecClear

### Prototype

d sdCollecClear	pr		
d pform		5u 0	
d component		30	varying value
d Collection		30	varying value

---

## sdCollecAdd

### Prototype

d sdCollecAdd	pr		
d pform		5u 0	
d component		30	varying value
d Collection		30	varying value

---

## sdCollecDelete

### Prototype

d sdCollecDelete	pr		
d pform		5u 0	
d component		30	varying value
d Collection		30	varying value
d Elt		10	0 value

---

## sdCollecIns

### Prototype

d sdCollecIns	pr		
---------------	----	--	--

```
d  pform          5u 0
d  component      30    varying value
d  Collection     30    varying value
d  Elt           10  0 val
```

### Exemple :

```
C          callp    sdCollecClear(F1:'listview1':'items')
C          callp    sdCollecAdd(F1:'listview1':'columns')
C          callp    sdSetString(F1:'listview1':
C          'columns(0).caption':'Colonne 1 ')

```

---

# Chapitre 17. Fonctions catégorie TStrings

---

## Introduction

Une propriété de type TStrings est une liste de chaîne de caractères.

Les propriétés suivantes sont des propriétés de type TStrings :

CMemo.lines, CListBox.items, CListBox.keys, CComboBox.items, CComboBox.Keys

*Remarque : Pour les propriétés de type TStrings, il est possible d'utiliser les fonctions de la catégorie List voir*

### **Optimisation**

*Dans toutes les applications client-serveur, les temps de réponse, c'est le nerf de la guerre.*

Ce chapitre vous présente les moyens utilisés par Silverdev pour optimiser les temps de réponse et vous donne quelques conseils pour améliorer vos applications.

---

## Fonctions avec réponse

L'appel aux fonctions demandant une réponse sont plus lentes car elles nécessitent un aller-retour sur le réseau.

Les fonctions ne renvoyant pas de valeur sont groupées dans un buffer.

C'est le cas par exemple des fonction sdSet, sdSetCell etc..

Ce buffer est envoyé au client à la fin du gestionnaire d'événement ou lors de l'appel d'une fonction qui renvoie une valeur (par exemple sdGet) ou encore à l'appel de la fonction sdApplySet.

A titre d'exemple :

Sur une machine de test, nous avons mesuré :

1000 appels à sdGet : Environ 3 secondes.

1000 appels à sdset Environ 15 millisecondes

## Cache des variables

Afin d'améliorer les performances des fonctions de type sdGet, à chaque événement, certaines valeurs de propriétés sont envoyées au serveur.

Ces valeurs sont mises en cache.

Les fonctions de type sdGet commencent par chercher dans ce cache.

Ce mécanisme est transparent pour le développeur, mais il permet d'améliorer nettement les performances.

Seules les propriétés qui ont la plus grande probabilité d'être interrogées sont mises en cache.

Par exemple, seule la propriété text du composant CEdit est mise en cache.

Dans le debugger :

Envoyer événement

Evenement : \_2.BtnOK.OnClick

-----

13/10/2007 15:01:59:218

-----

Envoi du cache

\_2.TITRE.text=Les robots

\_2.PRIX.value=4

\_2.STOCK.value=0

\_2.ACOMMANDER.value=Y

\_2.Themes.keySelected=2

\_2.Themes.text=Fantastique

\_2.IDAUT.value=17

\_2.NOMAUT.text=Dostoïevski

\_2.NOMEDI.text=j'ai Lu

\_2.IDEDI.value=4

\_2.DATEPARU.value=20040513

\_2.DATEPARU.DateIso=2004-05-13

-----

13/10/2007 15:01:59:234

-----

Execution Get

\_2.DATEPARU.ValidDate

Résultat : True

-----

13/10/2007 15:01:59:343

Execution Script

function Main() {

```
_2.ModalResult=1;
}
```

```
-----
```

```
C          Eval      TITRE= sdGet (F1: 'TITRE': 'Text')
C          if        sdGetBool (F1: 'DATEPARU': 'ValidDate')=*off
C          eval      Message =Message + 'La date est -
C                  invalide' + X'0D'
C          endif

C          Eval      PRIX=sdGetNum (F1: 'PRIX': 'Value')
C          Eval      STOCK=sdGetInt (F1: 'STOCK': 'Value')
C          Eval      ACOMMANDER=sdGet (F1: 'ACOMMANDER':
C                  'Value')
C          Eval      IDTHEME=sdGetNum (F1: 'Themes':
C                  'KEYSELECTED')
C          Eval      IDAUT=sdGetInt (F1: 'IDAUT': 'Value')
C          Eval      IDEDI=sdGetInt (F1: 'IDEDI': 'Value')
C          eval      DATEPARU=sdGetDate (F1: 'DATEPARU')
C          callp     sdSetInt (F1: '*FORM': 'ModalResult': Mrok)
```

## Vidage du cache

Lorsque vous appelez une fonction qui n'effectue pas un get, le cache est vidé. Il faut donc veiller autant que possible à appeler les fonctions de types get en premier dans le gestionnaire d'évènement.

Exemple :

Le code suivant

```
*/EVENT Button4_OnClick
/free
row = sdGetInt (f1: 'sfl1': 'rowSelected');
sdSetBool (f1: 'button1': 'enabled': *off);
```

est plus efficace que le code suivant :

```
*/EVENT Button4_OnClick
/free
sdSetBool (f1: 'button1': 'enabled': *off);
row = sdGetInt (f1: 'sfl1': 'rowSelected');
```

## Événements déclenchés par code

Lorsqu'un évènement est déclenché par code, le cache n'est pas envoyé par la partie cliente. (Dans le cas contraire, le cache pourrait ne pas être valide)

---

## Cache des fenêtres

Les fenêtres silverdev doivent être transmises du serveur vers le client.

Si celles ci contiennent des images, le transfert peut être assez long.

Les fenêtres créées sont donc enregistrées sur le disque du client.

Lors de la création d'une fenêtre, le client et le serveur commencent par discuter pour savoir si la fenêtre existe déjà sur le poste du client.

Les fenêtres sont stockées dans le répertoire /cache

Le nom du fichier cache est le code md5 de la fenêtre.

Lorsqu'une fenêtre est modifiée, et donc remise en cache, l'ancien cache de cette fenêtre est supprimé. La correspondance entre la provenance d'une fenêtre et son fichier cache est sauvegardée dans la base de registres.

---

## Fonctions de type List

Comme nous l'avons vu précédemment, les fonctions de type sdGet prennent un temps non négligeable.

La relecture de tous les éléments d'un composant de type treeview, memo, listbox, checklistbox...peut donc être assez long.

Les fonctions de la catégorie List permettent de gagner une temps considérable.

Le principe est de remonter toutes les informations intéressantes d'un composant, de les copier en mémoire afin de pouvoir les interroger en local. Pour plus de détail, voir (0)

---

## Fonction sdGetSfl

Afin de lire les valeurs dans un composant CSfl, utilisez la fonction sdGetSfl et les fonctions associées (sdGetSflStr,sdGetSflModified..)

Les données sont recopiées en mémoire sur le serveur.La lecture des données est ensuite très rapide.



---

## Code MD5 des icônes

Lors de l'affichage d'un répertoire dans MYDESK, différentes données sont envoyées à MyDesk.

Les informations concernant l'application et éventuellement une icône.

Les informations sont de type texte et ne représentent que quelques octets.

Les icônes sont beaucoup plus lourdes et représentent 90% des données à transférer.

Un mécanisme basé sur un code de hashage md5 est utilisé afin de n'envoyer les icônes qu'une seule fois.

Pour chaque application il existe sur le serveur un fichier à l'extension .app et éventuellement un fichier à l'extension .ico.

Le code MD5 de l'icône est stocké dans le fichier .app.

Ce fichier est envoyé en premier.

MyDesk teste l'existence de l'icône sur le disque.

Si l'icône existe déjà, elle est utilisée sinon, elle est téléchargée puis sauvée sur le disque avec comme nom, le code de hashage de l'icône.

Les icônes sont stockées dans le répertoire /icônes

Remarque :

Ces icônes sont aussi utilisées lors de la création de raccourcis sur le bureau.

---

## Réutilisation de fenêtres

Lorsque vous créez une fenêtre, des ressources sont utilisées côté client pour créer les composants de cette fenêtre.

Lorsque l'utilisateur ferme une fenêtre, celle ci est cachée, mais elle n'est pas détruite.

C'est à dire que les composants de la fenêtre et la fenêtre elle même utilisent de la mémoire côté PC.

La fenêtre sera détruite au moment de la fermeture du programme (côté pc)

Si vous souhaitez libérer la fenêtre au moment de sa fermeture, ajoutez le code suivant dans l'événement onclose de la fenêtre :

```
p ONCLOSE...
p          B
d          PI
D Parameters ds          based(pevtinf)
D Win          5u 0
D Evt          48a
```

```

,* Variable
D Action                                10i 0
,* 0 : ne rien faire
,* 1 : Cacher la fenêtre
,* 2 : Libérer la fenêtre
,* 3 : Minimiser la fenêtre
,*
c                                eval      action = 2
c                                callp     sdFreeForm(Win)

```

Que la fenêtre soit détruite ou non, n'oubliez pas de tester qu'une fenêtre n'existe pas déjà avant de la créer :

```

C                                If          F1 <> 0
C                                eval        F1 = sdCreateForm(F1REF)
C                                endif

```

Si elle existe déjà, vous allez créer plusieurs fenêtres identiques qui utiliseront des ressources inutilement.

---

## Compression

Lors des premiers échanges de données, le temps des échanges est mesuré. En fonction du temps calculé, le serveur détermine si le client est distant ou sur le réseau local. Si le client est distant, les données sont compressées avant d'être transmises.

---

## Evenements locaux

Il est possible d'effectuer tous les traitements dans des evenements serveurs, mais le traitement des evenenemts en local est plus rapide que le traitement sur le serveur. Lorsqu'un événement se produit très souvent et que le traitement ne necessite pas l'utilisation d'objets sur le serveur, préférez les événements locaux.

---

## sdAddNode

La fonction sdAddNode2 est à préférer à la fonction sdAddNode car elle est beaucoup plus rapide et plus simple d'utilisation.

## sdSIAdd

### Prototype

DsdSIAdd	pr	
d F1		5u 0 const
D component		30 varying value
D property		30 varying value
d value		999a varying value
d Trim		N value options(*nopass)

### Description :

La fonction sdSIadd ajoute une chaîne en fin de liste.

---

## sdSIClear

### Prototype

d sdSIClear	pr	
d F1		5u 0 const
D component		30 varying value
D property		30 varying value

### Description :

Efface toutes les chaines de l'objet TString.

---

## sdSIDelete

### Prototype

D sdSIDelete	pr	
d F1		5u 0 const
D component		30 varying value
D property		30 varying value
d Index		10i 0 value

### Description :

Supprime une chaîne dans une liste.

---

## sdSlGet

### Prototype

D sdSlGet	pr	999a	varying
d F1		5u 0	const
D component		30	varying value
D property		30	varying value
d Index		10 0	value

### Description :

Renvoie la chaîne d'indice Index

---

## sdSlInsert

### Prototype

D sdSlInsert	pr		
d F1		5u 0	const
D component		30	varying value
D property		30	varying value
d Index		10i 0	value
d value		999a	varying value
d Trim		N	value options(*nopass)

### Description :

Insère une chaîne dans un objet TStringList à l'emplacement indiqué par l'indice.

## Chapitre 18. Fonctions catégorie ensembles

Certaines propriétés sont de type ensemble, c'est à dire qu'elle peuvent contenir zéro,une ou plusieurs valeurs.

Exemple : La propriété font.style est un ensemble qui peut contenir les valeurs : FsBold,fsItalic,fsUnderLine,fsStrikeOut

L'affectation de ces valeurs peut se faire dans le designer, ou à l'exécution.

L'ensemble des fonctions qui manipulent ces valeurs peuvent être rangées en deux catégories :

La première catégorie fonctionne avec les valeurs(1,2,4,8 etc)

Il s'agit des fonctions :

SdSetSet

SdAddSet

SdDelSet

La seconde catégorie fonctionne avec les rangs((1,2,3,4 etc)

Dans cette catégorie, chaine de 32 caractères représente l'ensemble.

Exemple :La chaine '101100...' signifie bold, pas italic, underline, strikeout.

Il s'agit des fonctions :

SdGetset

SdUpdSet

sdIsInSet

### sdSetSet

#### Prototype :

```
d sdSetSet          pr
d  pform              5u 0
d  component          30   varying value
d  Property           30   varying value
d  P4                 30   varying value options(*nopass)
d  P5                 30   varying value options(*nopass)
d  P6                 30   varying value options(*nopass)
d  P7                 30   varying value options(*nopass)
d  P8                 30   varying value options(*nopass)
d  P9                 30   varying value options(*nopass)
d  P10                30   varying value options(*nopass)
```

d	P11	30	varying value options(*nopass)
d	P12	30	varying value options(*nopass)
d	P13	30	varying value options(*nopass)
d	P14	30	varying value options(*nopass)
d	P15	30	varying value options(*nopass)

**Description :**

La fonction sdSetSet permet de modifier une propriété de type "Set" comme par exemple Font.Style

**Exemple 1 :**

C	callp	sdSetSet(form:'button1':'font.style':
C		'fsBold':'fsUnderline')

**Exemple 2:**

Pour supprimer toutes les valeurs de l'ensemble :

C	callp	sdSetSet(form:'Button1':'font.Style')
---	-------	---------------------------------------

---

**sdAddSet****Prototype :**

d	sdAddSet	pr	
d	pform	5u 0	const
d	component	30	varying value
d	Property	30	varying value
d	Value	30	varying value

**Description :**

La fonction sdAddSet permet d'ajouter des éléments dans les propriétés de type "Set" comme par exemple Font.Style

**Exemple 1**

C	callp	sdAddSet(F1:'Label1':'Font.style':
C		'fsBold')
C	callp	sdAddSet(F1:'Label1':'Font.style':
C		'fsUnderLine')

## sdDelSet

### Prototype :

```
d sdDelSet      pr
d pform          5u 0 const
d component      30    varying value
d Property       30    varying value
d Value          30    varying value
```

### Description :

La fonction sdDelSet permet d'enlever des éléments dans les propriétés de type "Set" comme par exemple Font.Style

### Exemple :

```
C      callp      sdDelSet(F1:'Label1':'Font.style':
C                               'fsBold')
C      callp      sdDelSet(F1:'Label1':'Font.style':
C                               'fsUnderLine')
```

## SdGetSet

### Prototype :

```
d sdGetSet      pr          32
d pform          5u 0 const
d component      30a    varying value
d Property       50a    varying value
```

### Description :

Utilisez cette fonction pour interroger une propriété de type ensemble.

Le résultat est une chaîne de 32 caractères maximum.

Chaque caractère correspond à une valeur de l'ensemble.

Pour la propriété font.style par exemple, la chaîne '1011' correspond à :

<b>FsBold</b>	<b>True</b>
<b>FsItalic</b>	<b>False</b>
<b>FsUnderline</b>	<b>True</b>
<b>FsStrikeOut</b>	<b>false</b>

Exemple :

```
D Style          s          32
C                eval      style=sdgetset(f1:'label1':'font.style')
C                if        sdIsInSet(Style:2)
C                callp      sdShowmessage('2')
C                endif
```

---

## SdUpdSet

Prototype :

```
d sdUpdSet      pr          32
d pform          5u 0 const
d component      30a    varying value
d Property       50a    varying value
d Value          32a    varying value
```

Description :

Cette fonction permet de mettre à jour un ensemble.

Le paramètre Value est une chaîne de 32 caractères maximum.

Chaque caractère correspond à une valeur de l'ensemble.

Pour la propriété font.style par exemple, la chaîne '1011' correspond à :

<b>FsBold</b>	<b>True</b>
<b>FsItalic</b>	<b>False</b>
<b>FsUnderline</b>	<b>True</b>
<b>FsStrikeOut</b>	<b>false</b>

Exemple :

```
C                callp      sdUpdSet(f1:'label1':'font.style':'1111')
```

---

## SdIsInSet

Prototype :



d sdIsInSet	pr	N
d Chaine		32
d position		10i 0 value

**Description :**

Utilisez cette fonction pour savoir si une valeur est dans un ensemble.

Récupérez la valeur d'un ensemble à l'aide de la fonction sdGetSet, ou interroger directement un paramètre de type ensemble d'un événement.

**Exemple :**

```
pButton1_OnMouseDown...
p          B
d          PI
d PevtInf          *   const options(*nopass)
DParams          ds          based(PevtInf)
D Win          5u 0
D Event          48
D Button          10i 0
D Shift          32
C          if          sdIsInSet(Shift:3)
C          callp          sdShowMessage('3')
C          endif
```

Remarque :

La propriété font.style est composée des valeurs fsBold, fsItalic, fsUnderline et fsStrikeOut.

C	callp	sdUpdSet(f1:'label1':'font.style':'1010')
---	-------	---

est équivalent à :

C	callp	sdSetSet(form:'button1':'font.style':
C		'fsBold':'fsUnderlIne')

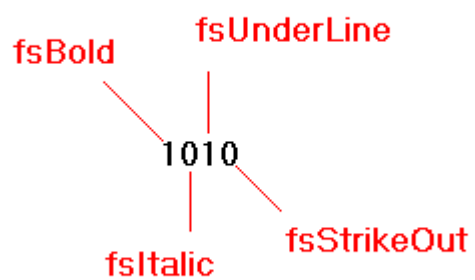


Figure 19

---

# Chapitre 19. Fonctions diverses

---

## SdHideMainForm

### Prototype :

```
d sdHideMainForm pr
```

### Description :

Par défaut, Silverdev rend visible la première fenêtre reçue même si la propriété visible de cette fenêtre est à false.

Ce comportement convient dans la plupart des cas, mais peut être gênant dans certaines applications. Si vous souhaitez que la première fenêtre ne soit pas visible, appelez la fonction sdHideMainForm.

Bien entendu, cette fonction doit être appelée avant le premier appel à la fonction sdcreateForm.

### Exemple :

```
C      callp      sdHideMainForm
c      eval      F1 = sdcreateForm(F1REF)
c      callp      sdSetCallBack(F1
c                  : 'MenuItem1.OnClick'
c                  : %paddr (
c                  'MENUITEM1_ONCLICK' )
...
```

---

## SdCursor

### Prototype :

```
d sdCursor      pr
dModif          N      value
```

### Description :

Lorsqu'un programme SILVERDEV signale un évènement au serveur, le curseur est modifié en sablier le temps que le serveur réponde.

Pour activer/désactiver cette fonctionnalité, utilisez la fonction sdCursor.

### Exemple :

```
C                                callp      sdCursor (*OFF)
```

## SdApplySet

Les fonctions Silverdev génèrent du code compréhensible par la partie cliente, c'est à dire le launcher.

Lorsque les fonctions ne retournent pas de valeur, le code généré est mis dans un buffer. Ce buffer est envoyé soit à l'appel d'une fonction qui renvoi une valeur (sdGet, sdshowmodal, sdSelectFile etc...) soit à la fin du gestionnaire d'évènement (c'est à dire à la sortie de la procédure associée à l'évènement) ou en encore à l'appel de la fonction sdApplySet.

Normalement, cette fonction n'a pas à être appelée, Silverdev s'occupe de l'appeler dès que nécessaire dans les deux cas précédemment cités par exemple.

Cependant, il est possible de rencontrer des cas que l'équipe de développement de silverdev n'aurait pas prévu.

## sdEnd

```
d sdEnd                                pr
```

### Description :

Indique à la partie cliente de se fermer.

Attention : Cette fonction indique à la partie cliente de se fermer, mais ne sort pas du programme rpg. Si des instructions suivent l'appel à la fonction sdEnd, ces instructions sont exécutées.

## sdInnerEnabled

```
d sdInnerEnabled pr
d form                                like(tWHandle) const
d component                          30    varying value
d value                              n      value
```

### Description :

La fonction `sdInnerEnabled` modifie la propriété `Enabled` des composants contenus dans celui auquel s'applique la fonction.

#### Remarque 1 :

La fonction s'applique de façon récursive.

---

## sdInnerReadOnly

```
d sdInnerReadOnly pr
d form                                like(tWHandle) const
d component                          30    varying value
d value                               n    value
```

#### Description :

La fonction `sdInnerReadOnly` modifie la propriété `readOnly` des composants contenus dans celui auquel s'applique la fonction.

#### Remarque 1 :

La fonction s'applique de façon récursive.

---

## sdGetClientIp

#### Prototype :

```
D sdgetClientIp...
D                                pr                20    varying
```

#### Description :

Cette fonction permet de récupérer l'adresse IP du pc connecté.

```
g value
```

---

## SdGetEnvVar

#### Prototype :

```
d sdEnvVar                                pr                *
```

d Variables	256	options(*string) value
-------------	-----	------------------------

**Description :**

La fonction sdGetEnVar permet d'interroger des variables d'environnement sur la machine cliente. Les variables à interroger doivent être séparées par des virgules. Il est ainsi possible de récupérer plusieurs valeurs en même temps.

**Exemple :**

```
Dlist      s      *
D i        s      10i 0

D Size     s      10u 0
D Value    s      500    varying
D Name     s      500    varying
C          eval    list = sdgetEnvVar('path,clientName')
C          for     i=0 to sdgetListcount(list)-1
C          eval    Name = sdgetListLabel(list:i)
C          eval    Value = sdgetListKey(list:i)

C          endfor
C          callp   sdFreelist(list)
```

---

**sdMsgDebug****Prototype :**

d sdMsgDebug	pr	
dTexte	3000	varying value

**Description :**

La fonction sdMsgDebug permet d'envoyer un message dans le programme de debug.

Exemple :

C	callp	sdmsgdebug('la variable i vaut : ' + %char(i))
---	-------	--

---

**sdSetKeepAlive**

### Prototype :

```
D sdSetKeepAlive  pr
D Interval                10u 0 value
```

### Description :

La fonction sdSetKeepAlive permet d'activer un timer côté client qui envoie une trame régulièrement au serveur. Le paramètre interval permet de déterminer le nombre de secondes entre les émissions de trame.

Cette fonctionnalité permet de détecter les jobs pour lesquels la connexion a été coupée et permet aussi de prévenir des déconnexions dues à un proxy ou un firewall.

Lorsque interval est supérieur à zéros, si le serveur ne reçoit plus de données pendant une durée égale au paramètre interval + 60 secondes, le job passe en MSGW (message SVD0208)

Lorsque le paramètre interval vaut zéros, les trames ne sont plus envoyées, et le serveur attend indéfiniment une action du client.

---

## SdBringApplicationToFront

### Prototype :

```
D sdBringApplicationToFront...
D                                pr
```

### Description :

La fonction sdBringApplicationToFront met l'application au premier plan.

---

## SdFlashApplication

### Prototype :

```
D sdFlashApplication...
D                                pr
```

### Description :

La fonction sdFlashApplication permet d'informer l'utilisateur que l'application requiert son attention en modifiant la couleur du bouton dans la barre des tâches.





---

## Chapitre 20. Liste des fenêtres et des évènements

---

---

### sdGetFormCount

#### Prototype :

d sdGetFormCount	pr	5u 0
------------------	----	------

#### Description :

Cette fonction permet de connaître le nombre de fenêtres en cours dans l'application.

---

### sdGetFormHandle

#### Prototype :

d sdGetFormHandle...		
d	pr	5u 0
D ind		5u 0

#### Description :

Cette fonction permet de connaître la valeur renvoyée par la fonction sdcreateForm lors de la création de la fenêtre.

---

### sdGetFormName

**Prototype :**

d sdGetFormName	pr	11a	Varying
d form		5u	0

**Description :**

Cette fonction permet de connaître le nom de la fenêtre côté client.

---

**sdGetFormLib****Prototype :**

d sdGetFormLib	pr	10
d form		5u 0

**Description :**

Cette fonction permet de connaître la bibliothèque de provenance de la fenêtre.

---

**sdGetFormObj****Prototype :**

d sdGetFormObj	pr	10
d form		5u 0

**Description :**

Cette fonction permet de connaître le nom du usrspc d'où provient la fenêtre.

---

## sdGetEventCount

### Prototype :

d	sdGetEventCount...		
d		pr	5u 0
D	PForm		5u 0

### Description :

**Cette fonction permet de connaître le nombre d'évènements d'une fenêtre.**

---

## sdGetEventName

### Prototype :

d sdGetEventName	pr	48
D PForm		5u 0
D EventNumber		5u 0

### Description :

Cette fonction permet de connaître le nom d'un évènement dans une fenêtre.

---

## sdGetEventProc

### Prototype :

d sdGetEventProc	pr	*	procptr
D PForm		5u 0	
D EventNumber		5u 0	

### Description :

Cette fonction permet de connaître l'adresse de la fonction associée à un évènement.

### Exemple :

D i	s	5u 0	
D j	s	5u 0	
D handle	s	5u 0	
D Lib	s	10	
D Obj	s	10	
D name	s	8	
D eventcount	s	5u 0	
D eventname	s	48	varying
D eventProc	s	*	procptr
C	for	i = 1 to sdGetFormCount	
C	eval	Handle=sdWinHandle(i)	
C	eval	Lib=sdGetFormLib(Handle)	
C	eval	Obj=sdGetFormObj(Handle)	
C	eval	Name=sdGetFormName(Handle)	
C	for	j = 1 to sdEventCount(handle)	

```
C      eval      eventname = sdEventName(handle:j)
C      eval      eventProc  = sdEventProc(handle:j)
C      endfor
C      endfor
```

| *Remarque : Ces fonctions n'interrogent pas la partie cliente.*

---

# Chapitre 21. Couleurs

---

## Designer

La modification d'une propriété de type couleur dans le designer se fait à l'aide d'une boîte de dialogue comme indiqué sur la figure suivante.



Figure 20

Le résultat est soit une couleur pré déterminée (clBlue, clRed, etc...) soit une valeur hexadécimale de 4 octets.

Les trois premiers octets représentent respectivement l'intensité RGB des couleurs bleu, vert et rouge. La valeur 00FF0000 représente un bleu pur de pleine intensité, 0000FF00, un vert pur de pleine intensité et 000000FF, un rouge pur de pleine intensité. 00000000 représente le noir et 00FFFFFF, le blanc.

Si l'octet de poids fort vaut zéro (00), la couleur obtenue est celle la plus approchante dans la palette système. Si l'octet de poids fort vaut un (01), la couleur obtenue est celle la plus approchante dans la palette actuellement réalisée. Si l'octet de poids fort vaut deux (02), la couleur obtenue est celle la plus approchante dans la palette logique du contexte de périphérique en cours.

---

## Modification à l'exécution

Pour modifier une propriété de type couleur à l'exécution, il faut donner les valeurs RGB .

Les trois exemples ci-dessous assignent la couleur bleu :

C	<code>callp</code>	<code>sdset(f1:'Tree1':'color':'0xFF0000')</code>
C	<code>callp</code>	<code>sdSetInt(F1:'tree1':'color':X'FF0000')</code>
C	<code>callp</code>	<code>sdSetInt(F1:'tree1':'color':16711680)</code>

Les deux premières solutions sont recommandées car plus lisibles.

---

## Choix de couleur

Pour laisser la possibilité à l'utilisateur de choisir une couleur, utilisez la fonction `sdSelectColor`.

Cette fonction affiche une boîte de dialogue identique à celle du designer.

La fonction `sdSelectColor` renvoie un booléen indiquant si l'utilisateur a choisi une couleur. La couleur choisie par l'utilisateur est alors rapportée dans le paramètre de la fonction. Voir (**`sdSelectColor`**)

---

## Couleurs prédéfinies

Certaines couleurs peuvent être indiquées par leur nom.

Dans ce cas, à l'exécution, utilisez le code suivant :

C	<code>callp</code>	<code>sdset(f1:'editnum1':'color':'clblue')</code>
---	--------------------	--

Remarque :

*Si vous interrogez la valeur d'une propriété couleur, vous obtiendrez la valeur numérique et pas le texte associé. Soit 16711680 pour `clBlue`.*

Attention : Les couleurs telles que `clInfoBkg`, `clBtnFace`... sont dépendantes de la configuration du pc. (click droit sur le bureau, propriétés/apparence/avancé)

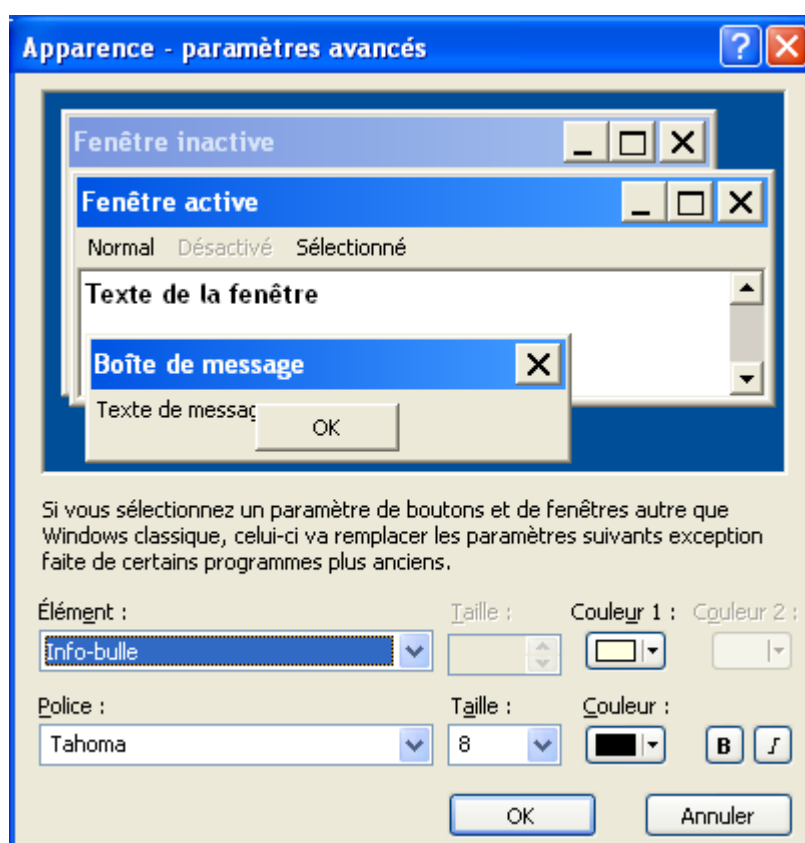


Figure 21

**Tableau récapitulatif :**

Valeur	Signification
CIAqua	Eau
CIBlack	Noir
CIBlue	Bleu
CICream	Cream
CIDkGray	Gris foncé
CIFuchsia	Fuchsia
clGray	Gris
clGreen	Vert
clLime	Vert citron
clLtGray	Gris clair
clMaroon	Marron
clMedGray	Gris moyen
clMoneyGreen	Vert menthe
clNavy	Bleu marine
clOlive	Vert olive
clPurple	Violet
clRed	Rouge



<b>clSilver</b>	<b>Argent</b>
<b>clSkyBlue</b>	<b>Bleu ciel</b>
<b>clTeal</b>	<b>Canard</b>
<b>clWhite</b>	<b>Blanc</b>
<b>clYellow</b>	<b>Jaune</b>
<b>clInfoText</b>	<b>Windows 95 ou NT 4.0 seulement : couleur du texte des fenêtres de conseils</b>
<b>clInfoBk</b>	<b>Windows 95 ou NT 4.0 seulement : couleur du fond des fenêtres de conseils</b>
<b>clGradientActiveCaption</b>	<b>Windows 98 ou 2000 seulement : couleur du côté droit dans le dégradé de couleur de la barre de titre d'une fenêtre active. clActiveCaption spécifie la couleur du côté gauche</b>
<b>clGradientInactiveCaption</b>	<b>Windows 98 ou 2000 seulement : couleur du côté droit dans le dégradé de couleur de la barre de titre d'une fenêtre inactive. clInactiveCaption spécifie la couleur du côté gauche</b>
<b>clDefault</b>	<b>Couleur par défaut pour le contrôle auquel la couleur est affectée</b>
<b>clBackground</b>	<b>Couleur du fond du bureau Windows</b>
<b>clActiveCaption</b>	<b>Couleur de la barre de titre de la fenêtre active</b>
<b>clInactiveCaption</b>	<b>Couleur de la barre de titre des fenêtres inactives</b>
<b>clMenu</b>	<b>Couleur du fond des menus</b>
<b>clWindow</b>	<b>Couleur du fond des fenêtres</b>
<b>clWindowFrame</b>	<b>Couleur des cadres de fenêtres</b>
<b>clMenuText</b>	<b>Couleur du texte des menus</b>
<b>clWindowText</b>	<b>Couleur du texte dans les fenêtres</b>
<b>clCaptionText</b>	<b>Couleur du texte de la barre de titres de la fenêtre active</b>
<b>clActiveBorder</b>	<b>Couleur de la bordure de la fenêtre active</b>
<b>clHighlight</b>	<b>Couleur du fond du texte sélectionné</b>
<b>clAppWorkSpace</b>	<b>Couleur de l'espace de travail de l'application</b>
<b>clHighlightText</b>	<b>Couleur du texte sélectionné</b>
<b>clBtnFace</b>	<b>Couleur de la face d'un bouton</b>
<b>clBtnShadow</b>	<b>Couleur de l'ombre projetée par un</b>

	<b>bouton</b>
<b>clGrayText</b>	<b>Couleur d'un texte estompé</b>
<b>clScrollBar</b>	<b>Couleur de la barre de défilement</b>
<b>clBtnText</b>	<b>Couleur du texte sur un bouton</b>
<b>clInactiveCaptionText</b>	<b>Couleur du texte de la barre de titre des fenêtres inactives</b>
<b>clBtnHighlight</b>	<b>Couleur d'un bouton sélectionné</b>
<b>cl3DDkShadow</b>	<b>Windows 95 ou NT 4.0 seulement : ombre foncée des éléments s'affichant en 3D</b>
<b>cl3DLight</b>	<b>Windows 95 ou NT 4.0 seulement : couleur claire des éléments s'affichant en 3D (pour les côtés faces à la source de lumière)</b>

---

# Chapitre 22. Ancrer des composants

---

## Introduction

Ce chapitre montre comment ancrer une fenêtre dans un panel à l'exécution.

---

## Exemple de base

Créez une fenêtre nommée SDDMDOCK1.  
Ajoutez un panel nommé panelLeft.  
Affectez la propriété align du composant panelLeft à alLeft.  
Affectez la propriété DockSite du composant panelLeft à true.

Créez une seconde fenêtre nommée SDDMDOCK2.

Modifiez la propriété dragMode de SDDMDOCK2 à dmAutomatic.  
Modifiez la propriété dragKind de SDDMDOCK2 à dkDock.

Appelez SDDMDOCK2 depuis SDDMDOCK1.  
La fenêtre SDDMDOCK2 peut être ancrée ou désancrée dans le composant panelLeft.

Ajoutez un composant de type CPanel nommé panelRight aligné à droite dans SDDMDOCK1.  
Modifiez la propriété DockSite de panelRight à true.

Il est alors possible de déplacer SDDMDOCK2 d'un panel à l'autre

*Remarque 1:*

*De la même façon, vous pouvez ancrer tous les composants de type TControl.*

*Remarque 2:*

*Tous les composants de type TWinControl peuvent servir de site d'ancrage (CPanel, CPageControl, CScrollBar....)*

---

## Fenêtre flottante

Dans l'exemple précédent, la fenêtre SDDMDOCK2 peut être déplacée d'un panel à l'autre, mais la fenêtre peut aussi être désancrée.

Si le composant désancré n'est pas une fenêtre, mais un panel par exemple (dockablePanel), celui-ci se retrouve alors dans une fenêtre autonome. Si cette fenêtre est fermée, il suffit de modifier la propriété visible de dockablePanel à true pour la faire réapparaître.

Vous pouvez empêcher le composant ancrable de devenir flottant à l'aide de l'événement local onUnDock.

L'exemple suivant empêche l'utilisateur de désancrer un composant à partir de panelLeft dans le vide.

```
procedure PanelLeft_OnUnDock (Sender: TObject; Client: TControl;
NewTarget: TWinControl; var Allow: Boolean);
begin
    allow := newTarget <> nil;
end;
```

---

## Amélioration

Nous souhaitons à présent rendre possible l'ancrage d'un composant à gauche de la fenêtre SDDMDOCK1, mais sans que le composant panelLeft n'apparaissent lorsqu'aucun composant n'y est ancré.

Affectez la propriété width de panelLeft à zéro.

Modifiez l'événement local onDockOver de panelLeft ainsi :

```
procedure PanelLeft_OnDockOver (Sender: TObject; Source:
TDragDockObject; X: Integer; Y: Integer; State: TDragState; var Accept:
Boolean);
var
  ARect: TRect;
begin
  Accept := true;
  if (Source.DockRect.Right = Source.DockRect.left) then
  begin
    ARect := Source.DockRect;
    ARect.Right := ARect.left + 100;
    source.dockRect := ARect;
  end;
end;
```

Modifiez l'événement local onDockDrop de panelLeft ainsi :

```
procedure PanelLeft_OnDockDrop (Sender: TObject; Source:
TDragDockObject; X: Integer; Y: Integer);begin
  if(This.panelLeft.width = 0) then
  begin
    This.panelLeft.width := source.dockRect.right - source.dockRect.left;
  end;
end;
```

Modifiez l'événement local onDockDrop de panelLeft ainsi :

```
procedure PanelLeft_OnUnDock (Sender: TObject; Client: TControl;
NewTarget: TWinControl; var Allow: Boolean);
begin
  allow := newTarget <> nil;
  if(allow) and (This.panelLeft.DockClientCount =1) then
  begin
    This.panelLeft.width := 0;
  end;
end;
```

# Chapitre 23. Opérations de drag and drop.

## Introduction

Lors d'une opération de drag and drop, il y a deux composants qui interagissent. Le composant source qui démarre l'opération de drag and drop, et le composant cible qui reçoit l'opération de drag and drop.

Le même composant peut être à la fois la source et la cible.

La source et la cible peuvent être dans deux fenêtres différentes.

La source et la cible doivent être dans le même processus.

Un drag and drop peut être effectué depuis et vers n'importe quel composant, cependant il est plus courant d'effectuer des drag and drop avec des composants ayant une liste de valeurs. (ClistBox, CtreeView, ClistView, CSFL etc..)

## DragMode

La propriété DragMode concerne le composant source.

La propriété DragMode peut prendre deux valeurs : dmManual et dmAutomatic.

Si la propriété DragMode est dmAutomatic, cela signifie que dès que l'utilisateur clique sur le composant, une opération de drag and drop est démarrée.

Le cursor change automatiquement comme suit : 

L'opération de drag and drop s'arrête lorsque l'utilisateur relâche le bouton gauche de la souris.

Si la propriété DragMode est dmManual, il faut dans l'évènement onMouseDown appeler la fonction sdBeginDrag.

```
d sdBeginDrag      pr
d  pform            5u 0 const
d  component        30   varying value
d  Immediate        N   value
d  TreshOld         10i 0 value
```

Le paramètre `immediate` permet d'indiquer si l'opération de drag and drop est lancée immédiatement ou si elle sera lancée lorsque l'utilisateur commencera à déplacer la souris.

Si le paramètre `immediate` est `*on`, le paramètre `Threshold` permet d'indiquer de combien de pixels la souris doit se déplacer avant que l'opération de drag and drop ne soit démarrée. La valeur `-1` permet d'utiliser la valeur par défaut configurée sur le pc.

---


## OnDragOver

Une fois que l'opération de drag and drop est démarrée l'événement `onDragOver` est déclenché pour tous les composants que la souris survole.

Les paramètres de l'événement sont :

```
d Win          5u 0
d evt          48a
d drag         n
d name         100a  varying
d PosX         10i 0
d PosY         10i 0
d WinSrc       5u 0
```

Drag	Paramètre modifiable. Mettre à <code>*ON</code> pour que l'évènement <code>OnDragDrop</code> puisse être déclenché. (Et que le curseur soit celui d'un drag and drop)
Name	Nom du composant qui a initié le drag and drop
PosX,PosY	Position de la souris.
WinSrc	Numéro de la fenêtre sur lequel se trouve le composant qui a initié le drag and drop.

Le fait de mettre le paramètre `drag` à `false` modifie le curseur  pour indiquer à l'utilisateur qu'il ne peut pas faire de drop sur ce composant.

Les autres paramètres permettent d'avoir des informations sur le composant qui a initié le drag and drop et sur l'endroit où a lieu le dragover.

---

## OnDragDrop

L'évènement `onDragDrop` est déclenché lorsqu'une opération de drag and drop a été démarrée, que l'utilisateur relâche le bouton gauche de la souris et que le

paramètre drag a été mis à \*on dans le précédent événement onDragOver de ce composant.

```

d PevtInf          *      const options(*nopass)
d parm            ds              based(pevtinf)
d Win              5u 0
d evt              48a
d name             100a    varying
d PosX              10i 0
d PosY              10i 0
d WinSrc            5u 0

```

Name	Nom du composant qui a initié le drag and drop
PosX, PosY	Position de la souris.
WinSrc	Numéro de la fenêtre sur lequel se trouve le composant qui a initié le drag and drop.

Les paramètres de l'évènement permettent d'avoir des informations sur le composant qui a démarré le drag and drop et sur l'endroit où a lieu le drop.

## Examples

### Drag and drop d'un ClistBox vers un ClistBox

```

~/EVENT ListBox2_OnDragOver
,* -----*
,* Description :*
,* -----*
D Parameters      ds              based(pevtinf)
D Win              5u 0
D Evt              48a
,* Modifiable
D Drag              n
,* Non modifiable
D Name              100a    varying
D PosX              10i 0
D PosY              10i 0
,*

c                  eval      drag = *on

~/EVENT ListBox2_OnDragDrop
,* -----*
,* Description :*
,* -----*
D Parameters      ds              based(pevtinf)

```



```

D Win                                5u 0
D Evt                                48a
D Name                              100a   varying
D PosX                               10i 0
D PosY                               10i 0
,*
d sel                                s      10i 0
d Dest                               s      10i 0
d chaine                             s      30a
d cle                                s      30a
d List                               s      *
d i                                  s      10i 0
C                                  eval      Dest = sdItemAtPos (Win:'ListBox2':
C                                  PosX:PosY:*ON)
C                                  eval      List=sdGetList (Win:Name)
C                                  for        i =0 to sdGetListCount (List) -1
C                                  if        sdGetListSelected (List:i)
C                                  eval      chaine = sdgetListLabel (List:i)
C                                  eval      cle = sdGetListKey (List:i)
C                                  if        Dest =-1
C                                  callp     sdSlAdd (F1:'ListBox2': 'Keys':cle)
C                                  callp     sdSlAdd (F1:'ListBox2': 'Items':chaine)
C                                  else
C                                  callp     sdSlInsert (F1:'ListBox2': 'Keys':Dest:Cle)
C                                  callp
sdSlInsert (F1:'ListBox2': 'Items':Dest:Chaine)
C                                  eval      Dest=Dest+1
C                                  endif
C                                  endif
C                                  endfor
C                                  callp     sddelselected (Win:name)
C                                  callp     sdFreeList (List)

```

## Drag and drop d'un ClistView vers un ClistView

```

~/EVENT TreeView1_OnDragOver
,* -----*
,* Description :                                     *
,* -----*
D Parameters      ds                                based(pevtinf)
D Win              5u 0
D Evt              48a
,* Modifiable
D Drag            n
,* Non modifiable
D Name            100a   varying
D PosX            10i 0

```

```

D PosY                                10i 0
,*
C                                eval    drag = (Name ='TreeView1')

~*/EVENT TreeView1_OnDragDrop
,* -----*
,* Description :                                *
,* -----*
D Parameters      ds                      based(pevtf)
D Win                                5u 0
D Evt                                48a
D Name                                100a   varying
D PosX                                10i 0
D PosY                                10i 0
,*
DNode1            s                      10i 0
DNode2            s                      10i 0
C                                eval    node1=sdgetInt(F1:'TreeView1':
C                                'FirstSelected')
C                                eval    node2=sdNodeAt(F1:'treeView1':PosX:PosY)
C                                if      Node1 = -1  or Node1=Node2
C                                return
C                                endif
C                                if      Node2=-1
C                                callp   sdMoveNode(F1:'TreeView1':Node1:0:
C                                'naAdd')
C                                else
C                                callp   sdMoveNode(F1:'TreeView1':Node1:Node2:
C                                'naAddChild')
C                                endif

```

## DragCursor et CCustomCursor

Il est possible de modifier le curseur associé à l'opération de drag and drop.

Dans l'exemple suivant, nous changeons le curseur lorsque l'utilisateur maintient la touche Control appuyée au moment du onMouseDown :

```

D Parameters      ds                      based(pevtf)
D Win                                5u 0
D Evt                                48a
D Button                                10i 0
D Shift                                32
D X                                10i 0
D Y                                10i 0
,*
C                                if      sdIsInSet(shift:3)

```

```

c      callp      sdSetInt(F1: 'TreeView1': 'dragCursor': 1)
c      else
c      callp      sdSetInt(F1: 'TreeView1': 'dragCursor': -12)
c      endif
c      callp      sdBeginDrag(F1: 'TreeView1': *off: -1)

```

La valeur affectée à la propriété dragCursor doit être comprise entre -1 et -22 (curseurs prédéfinis) ou à une valeur définie pour un curseur personnalisé. Un curseur personnalisé est défini à l'aide du composant CCustomCursor.

En interrogeant la valeur de dragCursor au moment du drop, il est possible de déterminer comment a été initié le drag and drop.

Il est ainsi possible par exemple de distinguer un déplacement ou une copie selon que la touche contrôle soit activée au moment du onMouseDown.

## OnDragOverEx, OnDragDropEx

Pour simplifier les opérations de drag and drop vers un composant CSFL, le composant CSFL a deux événements supplémentaires.

Ces deux événements fonctionnent comme les événements OnDragOver et OnDragDrop, mais diffèrent par leurs paramètres.

Les paramètres de OnDragOverEx et OnDragDropEx permettent de connaître rapidement la cellule survolée.

### Paramètres de OnDragOverEx

```

d PevtInf      *      const options(*nopass)
d parm         ds      based(pevtinf)
d Win          5u 0
d evt          48a
d drag         n
d WinSrc       5u 0
d name         100a    varying
d Row          10i 0
d Col          100a    varying

```

Drag	Paramètre modifiable. Mettre à *ON pour que l'évènement OnDragDrop puisse être déclenché. (Et que le curseur soit celui d'un drag and drop)
WinSrc	Numéro de la fenêtre sur lequel se trouve le composant qui a initié le drag and drop.

Name	Nom du composant qui a initié le drag and drop.
Row	Ligne ou le drag est effectué.
Col	Colonne ou le drag est effectué.

### Paramètres de OnDragDropEx

```

d PevtInf          *    const options(*nopass)
d parm            ds          based(pevtinf)
d Win              5u 0
d evt             48a
d WinSrc           5u 0
d name            100a    varying
d Row             10i 0
d Col             100a    varying

```

WinSrc	Numéro de la fenêtre sur lequel se trouve le composant qui a initié le drag and drop.
Name	Nom du composant qui a initié le drag and drop.
Row	Ligne ou le drop est effectué.
Col	Colonne ou le drop est effectué.

## Afficher un popupmenu

L'exemple suivant permet d'afficher un popupmenu au moment du drop :

```

*/EVENT Edit2_OnDragDrop
* -----*
* Description :
* -----*

D Parameters      ds          based(pevtinf)
D Win             5u 0
D Evt            48
D Name           100a    varying
D X              10i 0
D Y              10i 0
D WinSrc         5u 0
D pt             ds          likeds(DsPoint)
/free

```

```
pt.x = x;  
pt.y = y;  
pt = sdGetClientToScreen(F1:'Edit2':pt);  
sdPopup(F1:'PopupMenu1': pt.x: pt.y);  
/end-free
```

Remarque : L'appel à `sdGetClientToScreen` est nécessaire car les paramètres `x` et `y` de l'évènement sont les coordonnées dans le composant recevant le drop et la fonction `sdPopup` attend des coordonnées écran.

---

# Chapitre 24. Adaptation taille écran

---

## Introduction

L'adaptation d'une fiche à la taille de l'écran est un sujet important et complexe. Ce sujet mérite donc un chapitre à lui seul afin de voir les différentes possibilités.

---

## Taille modifiable par l'utilisateur

Pour que l'utilisateur puisse modifier la taille d'une fiche, il faut que la propriété `borderStyle` de la fiche soit `bsSizeable` (valeur par défaut)

La propriété `borderIcons`, si elle contient la valeur `biMaximize` permet aussi à l'utilisateur de passer en mode plein écran.

---

## Align, anchors

Le plus simple pour s'adapter à la modification de la taille d'un écran est d'utiliser les propriétés `align` et `anchors`.

Lorsque la propriété `align` est à `alBottom`, le composant occupe tout l'espace bas de son conteneur.

Lorsque la propriété `align` est à `alTop`, le composant occupe tout l'espace haut de son conteneur.

Lorsque la propriété `align` est à `alRight`, le composant occupe tout l'espace droit de son conteneur.

Lorsque la propriété `align` est à `alLeft`, le composant occupe tout l'espace gauche de son conteneur.

Lorsque la propriété `align` est à `alClient`, le composant occupe tout l'espace restant de son conteneur.

Vous pouvez donc par exemple mettre un panel avec `align = alBottom` et une grille avec `align = alClient`.

Le panel occupera toujours l'espace bas de la fiche , et la grille tout le reste de l'espace

---

## Anchors

Dans l'exemple précédent, on veut ajouter deux boutons dans le panel du bas.

Par défaut ces deux boutons on leur propriété anchors avec la valeur [akLeft,akTop]

Cela signifie que le composant restera toujours à la même distance du bord gauche et du bord top de son conteneur.

Vous pouvez modifier en [akRight,akBottom] pour que le composant reste à la même distance des bords bas et droit

Si vous modifiez anchors pour qu'il contienne par exemple akRight et akLeft, afin de rester à la même distance des bords gauche et droit, il s'agrandira.

Cela peut être une solution pour avoir un composant qui s'agrandit sans toucher les bords de son conteneur.

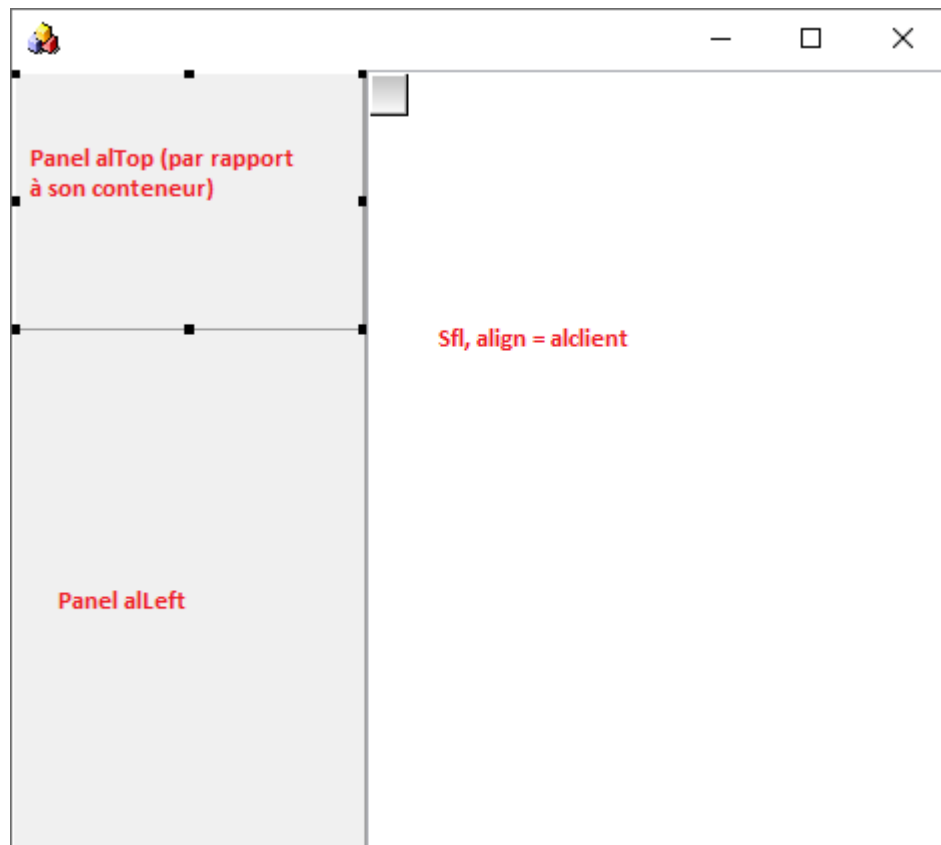
Pour un composant donné, choisissez d'utiliser align ou anchors, mais pas les deux en même temps.

En revanche, vous pouvez utiliser align pour certains composants et anchors pour d'autres (comme c'est le cas dans notre exemple)

---

## Conteneurs imbriqués

Afin d'obtenir une disposition complexe , il es possible par exemple d'avoir un panel avec align = alLeft, et à l'intérieur de ce panel, remettre un panel avec align = alTop par exemple



---

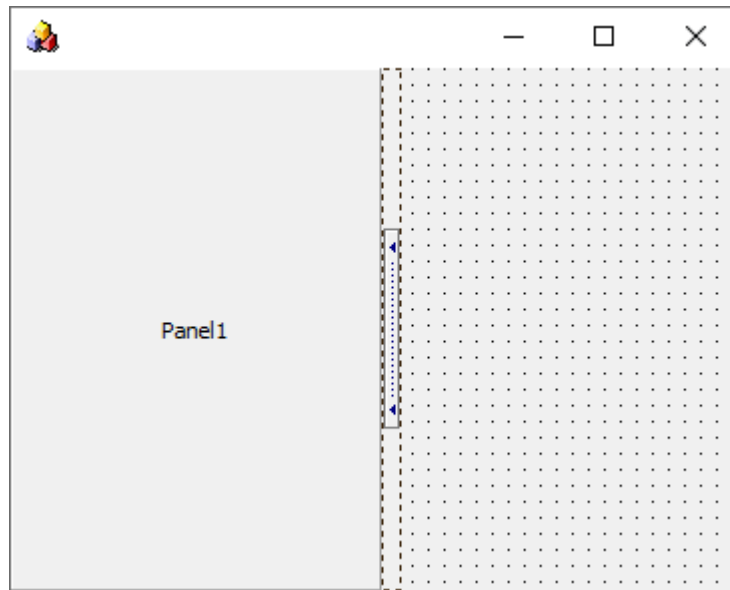
## Composant Splitter

Lorsque vous utilisez la propriété `align` pour splitter un écran, par exemple avec une zone en haut, une zone en bas et une zone à gauche, c'est le développeur qui décide de la taille des zone `top`, `bottom` et `left`.

Afin que l'utilisateur puisse modifier la taille de ces zones (et donc de la zone restante en `alClient`), ajoutez un composant `CSplitter`.

Si la propriété `align` du `CSplitter` est `alLeft`, il viendra se coller au composant en `alLeft`. L'utilisateur pourra alors modifier la largeur de la partie gauche.





---

## Gridpanel, flowPanel, relativePanel

Une autre approche possible est l'utilisation des composants GridPanel, FlowPanel et RelativePanel

L'idée est que ce sont les panels qui gèrent la disposition des composants. Les valeurs left et top des composants ne sont plus respectées.

Le panel redéfinit les positions des composants enfants à chaque fois qu'ils changent de taille.

Ces trois composants peuvent eux aussi être imbriqués.

---

## Composant CScreen

Le composant CScreen vous permet d'interroger la taille de l'écran de l'utilisateur.

Vous pouvez alors en fonction de la taille de l'écran de l'utilisateur, déplacer/redimensionner les composants par programme.

---

## Evènement onResize.

L'évènement onResize vous permet de détecter que l'utilisateur a changé la taille d'une fiche. Là encore, vous pouvez déplacer/redimensionner les composants par programme.

---

# Chapitre 25. ADO

---

## Introduction

ActiveX Data Objects est une technologie Microsoft fournissant une interface d'accès aux données dans l'environnement Windows.

Cette technologie permet d'accéder à une base de donnée distante.

L'accès aux données se fait depuis le client et non depuis le serveur de silverdev.

Les drivers ole dbv doivent donc être installés sur la machine cliente ou sera exécutée l'application silverdev.

L'ensemble des composants est dans la palette ADO.

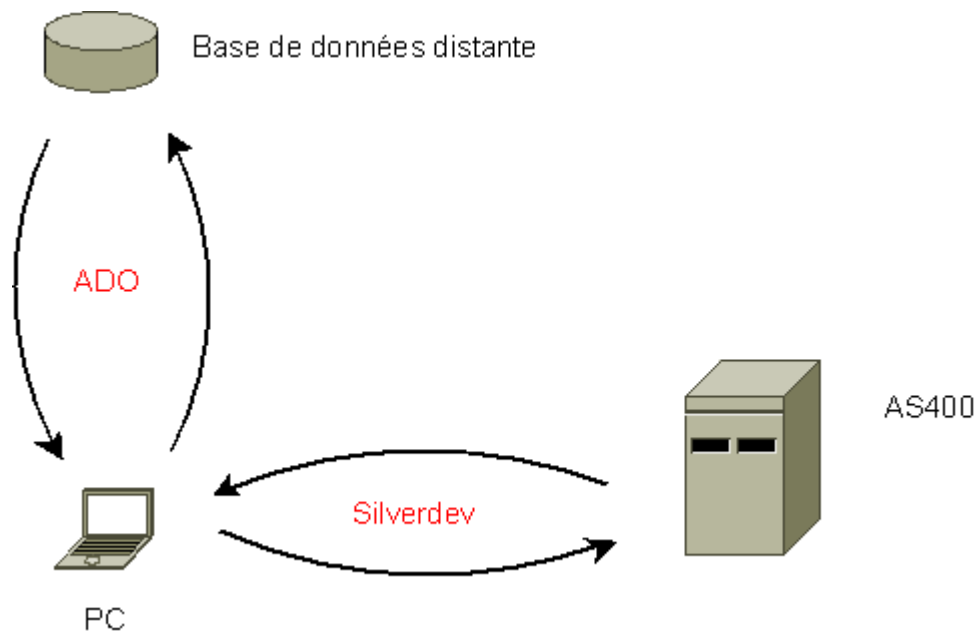


Figure 22

---

## CADOConnection

Le composant CADOConnection permet de définir une connexion à un driver ole. Ce composant doit ensuite être référencé par les autres composants.

La propriété clef de ce composant est la propriété ConnectionString .

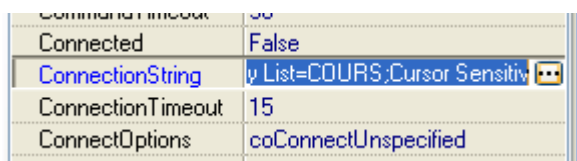


Figure 23

Un éditeur spécifique vous est proposé pour cette propriété.

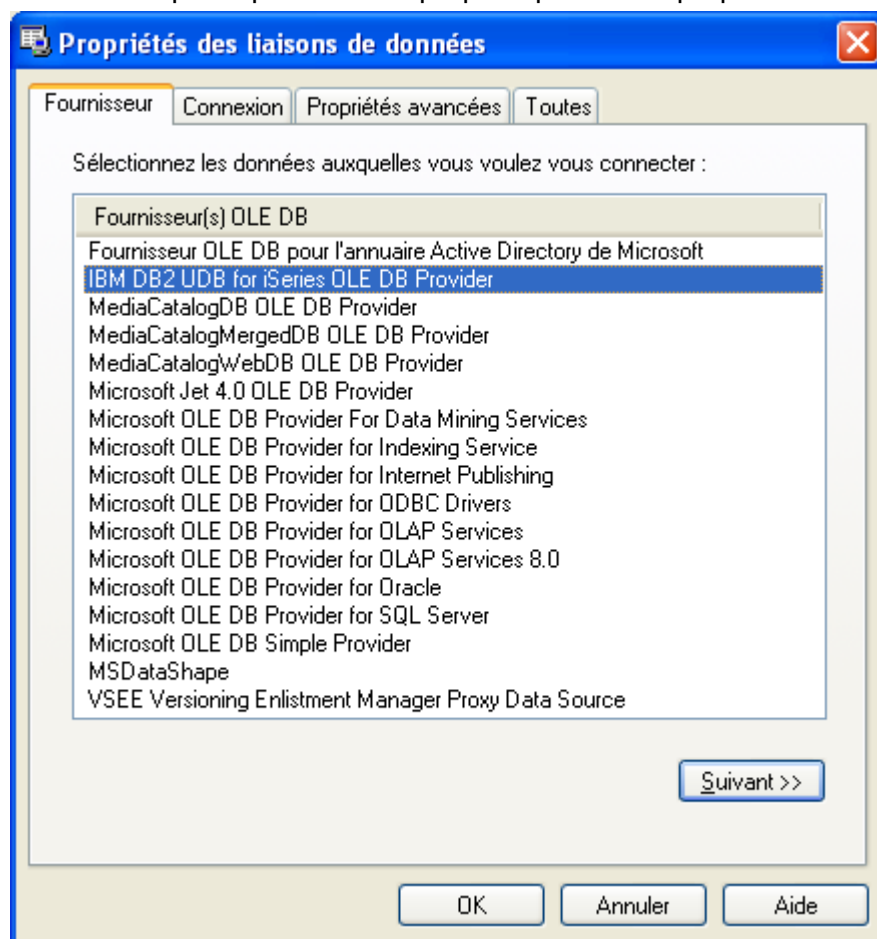


Figure 24

## CAdoQuery

Les autres composants de la palette ADO possèdent une propriété `ConnectionString`, mais il est plus simple de les faire pointer vers un composant `CAdoConnection` via la propriété `Connection`.

Le composante `CAdoQuery` permet d'exécuter une requête sur la base distante.

Le résultat de la requête peut être obtenu à l'aide de la fonction `sdGetSFL`.

Cette fonction créée pour le composant `CSFL` fonctionne aussi pour les composants de type ADO. Le paramètre colonne sera alors le nom d'un champ.

**Exemple :**

```

Dsfl          s          *
Dnomcli       s          40
D I           s          10i 0
  *Ouverture de la connexion
C              callp      sdSetBool(F1:'adoconnection1':'connected'
C                      *on)
  *Renseignement de la requête
C              callp      sdSlClear(F1:'adoquery1':'sql')
C              callp      sdSlAdd(F1:'adoquery1':'sql':
C                      'select * from cours.clients')
  *Application de la requête
C              callp      sdSetBool(F1:'adoquery1':'active'
C                      *on)
  *récupération du résultat
C              eval       sfl=sdGetSfl(F1:'adoquery1')
C              for        i= 1 to sdGetSflLines(sfl)
C              eval       nomcli=sdGetsflStr(sfl:'NOMCLI':i)
C              endfor
  *Libération des ressources allouées par sdGetSfl
C              callp      sdFreeSfl(sfl)

```

**Exemple 2 :**

```

DMsg          s          65000    varying
Dret          s          10i 0
C              callp      sdSetBool(F1:'adoconnection1':'connected'
C                      *on)
C              callp      sdSlClear(F1:'adoquery1':'sql')
C              callp      sdSlAdd(F1:'adoquery1':'sql':
C                      'insert into  cours.clients' +
C                      'values (50,1, ''test'',NULL,NULL, '+
C                      'NULL,NULL,NULL,NULL) ')
C              eval       ret=sdExecSql(F1:'AdoQuery1':msg)
C              callp      sdshowmessage(%char(ret))
C              callp      sdshowmessage(msg)

```

---

## Chapitre 26. Exit points

Pour associer un programme avec un exit point, utilisez la commande WRKREGINF, puis utilisez l'option 8.

Silverdev fournit les exits points suivants :

---

### SVDCTRLCPL

Permet de contrôler la compilation d'un écran

Exit point format :

CTRCPL01

Paramètres :

PGM	PARM(&USER &LIB &USRSPC &RET &TEXTRET)
DCL	VAR(&USER) TYPE(*CHAR) LEN(10)
DCL	VAR(&LIB) TYPE(*CHAR) LEN(10)
DCL	VAR(&USRSPC) TYPE(*CHAR) LEN(10)
DCL	VAR(&RET) TYPE(*CHAR) LEN(1)
DCL	VAR(&TEXTRET) TYPE(*CHAR) LEN(256)

Le paramètre User est le profil connecté.

Le paramètre Lib est la bibliothèque où l'écran va être compilé.

Le paramètre Usrspc est le nom de l'écran qui va être compilé

Le paramètre Ret doit renvoyer Y ou N (N, la compilation sera bloquée)

Le paramètre TextRet est un texte qui sera affiché dans le designer dans le cas où Ret est à N

---

### SVDEXCEPT

Permet d'intercepter une exception/erreur.

Exit point Format :

EXCEPT01

Paramètres :

D	pgm		extpgm('PGM')
D	jobName	10	const
D	jobUser	10	const
D	jobNumber	6	const
D	stopJob	1	

Affectez la valeur 'Y' au paramètre stopJob pour arrêter le job.

Si vous ne modifiez pas le paramètre stopJob, le job restera probablement dans l'état MSGW.

---

## SVDSTRAPP

Exit point Format :

SVDSTRAP

Paramètres :

D	pgm		extpgm('PGM')
D	user	10	const
D	ipadress	20	const
D	commande	32000	varying const
D	application	250	varying const

---

## SVDSTRPGM

Exit point Format :

SVDSTRPG

Paramètres :

D	pgm		extpgm('PGM')
D	user	10	const
D	jobName	10	const
D	jobUser	10	const

D jobNumber	6	const
-------------	---	-------



---

## Chapitre 27. Interaction avec des programmes externes.

Plusieurs solutions existent pour lancer silverdev depuis un autre programme ou pour lancer un autre programme depuis silverdev. Ce chapitre fait la synthèse des ces solutions.

Quelque soit la solution, Silverdev et le programme externe s'exécutent de manière indépendantes l'un de l'autre.

---

### Lancement d'une application Silverdev depuis un programme sur pc.

Pour exécuter une application silverdev, vous devez appeler le programme `launcher.exe`.

Lors de l'installation de silverdev, la variable d'environnement `SVDHOME` est créée correspondant au répertoire où se trouve le programme `launcher.exe`

Vous pouvez (et vous devez) passer un paramètre au programme `launcher.exe`. Ce paramètre doit être sous la forme d'une url contenant des couples clef/Valeur.

Exemple (les valeurs sont en vert) :

```
silverdev:host=192.168.0.80&port=7003&application=/examples/books/sddmbks  
&user=TOTO&pwd=1234
```

Les clefs possibles sont :

Clef	Obligatoire	Description
Host	Oui	Adresse IP du serveur
Port	Oui	Adresse tcp du serveur
Application	Oui	Application silverdev à lancer
System		Nom du système. Si cette valeur est indiquée, les valeurs host, port, user et pwd sont inutiles et seront écrasées par celles enregistrées sur le poste
User	Non	Utilisateur à connecter
Pwd	Non	Mot de passe
Caption	Non	Titre affiché dans la barre des tâches

Index	Non	Index de l'image interne
Md5	Non	Code md5 de l'icône
UserData	Non	Valeur qui peut être relue via sdGetUserData

Remarque :

Si une valeur possède des caractères tels que espace ou égal, il faut coder la valeur comme une url , exemple :

la valeur tic & tac doit être encodée : tic%20%26%20tac

## Lancement d'une application silverdev depuis un programme 5250

Pour lancer une application silverdev depuis un programme 5250, vous pouvez utiliser la commande strpcmd.

Si vous avez besoin de lancer une application silverdev avec des paramètres, vous pouvez procéder comme suit :

Le principe est de créer sur l'ifs un fichier contenant les informations du programme à lancer puis d'utiliser la commande strpcmd pour lancer le programme.

Ci dessous l'exemple complet :

```
Hbnddir('QC2LE')
Hdftactgrp(*no)
d O_RDONLY          S          10i 0 inz(1)
d O_WRONLY          S          10i 0 inz(2)
d O_RDWR            S          10i 0 inz(4)
d O_CREAT            S          10i 0 inz(8)
d O_EXCL             S          10i 0 inz(16)
d O_TRUNC            S          10i 0 inz(64)
d O_TEXTDATA         S          10i 0 inz(16777216)
d O_CODEPAGE         S          10i 0 inz(8388608)
d O_INHERITMODE      S          10i 0 inz(134217728)

d S_IRUSR            S          10i 0 inz(256)
d S_IWUSR            S          10i 0 inz(128)
d S_IXUSR            S          10i 0 inz(64)
d S_IRWXU            S          10i 0 inz(448)
d S_IRGRP            S          10i 0 inz(32)
d S_IWGRP            S          10i 0 inz(16)
d S_IXGRP            S          10i 0 inz(8)
d S_IRWXG            S          10i 0 inz(56)
d S_IROTH            S          10i 0 inz(4)
d S_IWOTH            S          10i 0 inz(2)
```

```

d S_IXOTH          S          10i 0 inz(1)
d S_IRWXO          S          10i 0 inz(7)

D access           pr          10i 0 extproc('access')
D filename         *          value options(*string)
d
D amode            10i 0 value

d open             pr          10i 0 extproc('open')
d filename         *          value options(*string)
d openflags        10i 0 value
d mode             10u 0 value options(*nopass)
d codepage         10u 0 value options(*nopass)

d close            pr          10i 0 extproc('close')
d filehandle       10i 0 value

d write            pr          10i 0 extproc('write')
d filehandle       10i 0 value
d datatowrite      *          value options(*string)
d nbytes           10u 0 value

d Qcmdexc          pr          extpgm('QCMDEXC')
d command          1024A      const options(*varsize)
d length           15P 5      const

Dfd                s          10i 0
Dpath              s          250   varying
DBuffer            s          2000   varying
DlnBuffer          s          10i 0
DpBuffer           s          *      inz(%addr(buffer))
DplnBuffer         s          *      inz(%addr(lnbuffer))
Dcmd               s          500   varying
d flags            s          10i 0
d mode             s          10i 0

d tIconv           ds
d tIconvRc         10i 0
d tIconvCd         10i 0          dim(12)

d fromCode         ds
d fcCCSID          10i 0
d fcConvert         10i 0          inz(0)
d fcShiftState     10i 0          inz(0)
d fcInputLen       10i 0          inz(0)
d fcError          10i 0          inz(0)
d                  8      inz(*allx'00')
```

```

d toCode          ds
d  tcCCSID          10i 0
d  tcConvert         10i 0          inz(0)
d  tcShiftState      10i 0          inz(0)
d  tcInputLen        10i 0          inz(0)
d  tcError           10i 0          inz(0)
d                   8      inz(*allx'00')

d iconvOpen        pr              like(tIconv) extproc('QtqIconvOpen')
d  fCode            *              options(*string) value
d  tCode            *              options(*string) value

d iconvClose       pr              extproc('iconv_close')
d  iconv_t          like(tIconv) value

d iconv            pr              10U 0 extproc('iconv')
d  iconv_t          like(tIconv) value
d  from             *
d  fromLen          *      value
d  to               *
d  toLen            *      value
D cpt              s              10u 0 inz(0)
D fileName          s              250  varying
/free
pbuffer=%addr(buffer) + 2;
fileName = %char(cpt)+'.app';
path = '/silverdev/applications/auto/'+fileName;
dow access(path:0) = 0;
  cpt = cpt + 1;
  fileName = %char(cpt)+'.app';
  path = '/silverdev/applications/auto/'+fileName;
enddo;
flags = O_WRONLY + O_CREAT + O_TRUNC+ O_CODEPAGE;
mode = S_IRUSR + S_IWUSR + S_IXUSR;
fd = open(path:flags:mode:819);
Buffer = '[001]'+X'0d25' + 'Description = MADESCRIPTION'+X'0d25' +
'Launch = call MALIB/MONPGM parm(''MONPARAMETRE'')' +X'0d25'+
'Caption = MONCAPTION' +X'0d25' + 'IconIdx = 1' +X'0d25' +
'MD5 = 26C0066C15293712903ECC1E20D4CEA1';
fcCCSID = 0;
tcCCSID = 1252;
tIconv = iconvOpen(%addr(toCode):%addr(fromCode));
lnBuffer = %len(buffer);
iconv(tIconv:pbuffer:plnBuffer:pbuffer:plnbuffer);
iconvClose(tIconv);
callp write(fd:buffer:%len(buffer));
callp close(fd);
path = '/auto/001';
cmd= 'STRPCCMD pccmd("SvdLauncher.exe '+

```

```
'host=192.168.0.42&port=7003&application=/auto/'+fileName + '"')';
Qcmdexc( cmd: %len(cmd) );
*inlr = *on;
/end-free
```

Remarque : Pour que le fichier .app soit supprimé après avoir été utilisé, ajoutez le code deleteApp = 'Y' dans le fichier .app

---

## Lancement d'une application Silverdev depuis une page html

Pour lancer une application silverdev depuis une page html, ajoutez un lien sous la forme suivante (le texte en vert est à modifier selon votre cas):

Exemple :

```
<a
href=silverdev:host=192.168.0.80&port=7003&application=/examples/books/sddmb
ks&user=TOTO&pwd=1234>Texte</a>
```

---

## Lancement d'une application Silverdev depuis une application Silverdev

Pour lancer une application silverdev dans le même processus, utilisez l'instruction call.

Pour lancer une application silverdev dans un processus différent, utilisez la fonction sdLaunch.

### Prototype :

d sdLaunch	pr		
d Commande		255	Varying Value
d Caption		255	Varying Value
d MD5		255	Varying Value
d Ind		255	Varying Value

### Description :

Permet de lancer une autre application Silverdev.

Paramètres :

Commande	Commande as400 à lancer
Caption	Titre de l'application
MD5	Code md5 de l'icône (si l'utilisateur a cette icône dans son cache, elle sera utilisée)
Ind	Index de l'icône interne à utiliser si icône correspondant à MD5 non trouvée.

## Lancement d'un programme depuis silverdev

Pour lancer un programme externe, utilisez la fonction sdExecutePc

Prototype :

```
d sdExecutePc      pr
d  action          256a  varying value
d  file            256a  varying value
d  parameters      256a  varying value
d  Window          2    0  value
```

Description :

La fonction sdExecutePc permet de lancer l'exécution sur le pc d'un fichier.

Il est possible de lancer un exécutable ou bien un fichier dont l'extension est associée à un exécutable. (ex : .xls pour excel)

Paramètres :**action :**

Action à réaliser sur le fichier.

Les actions courantes sont :

<b>open</b>	Ouvre le fichier spécifié par le second paramètre. Le fichier peut être de type exécutable, document ou dossier.
<b>edit</b>	Ouvre le document avec un éditeur associé
<b>print</b>	Imprime le document
<b>find</b>	Lance une recherche dans le répertoire spécifié par le second paramètre.
<b>explore</b>	Ouvre un explorateur dans le répertoire spécifié par le second paramètre.

Il peut y avoir d'autres valeurs possibles pour certains fichiers.

Les valeurs possibles pour un fichier sont les éléments que vous trouvez dans la base de registre sous :

HKEY\_CLASSES\_ROOT\object\_name\shell\verb

OU *object\_name* est le nom de l'objet.

La sous clef commande contient les données indiquant l'action exécutée.

### File :

Fichier sur lequel s'applique l'instruction.

### Parameters :

Paramètres pour l'exécutable lancé.

### Window :

Type d'ouverture de la fenêtre.

### Valeurs possibles :

constante	valeur	Signification
W_HIDE	0	La fenêtre est invisible (Utile pour l'action print)
W_NORMAL	1	La fenêtre est normale.
W_MIN	2	La fenêtre est minimisée
W_MAX	3	La fenêtre est maximisée.
W_NOA	4	La fenêtre a une taille normale, mais n'est pas activée.
W_MIN_NOA	7	La fenêtre est minimisée et non activée.

### Erreurs :

L'exécution du programme peut ne pas bien se passer. Dans ce cas, consultez le debugger de Silverdev. Dans la partie Trace, un texte est affiché indiquant si l'exécution s'est bien passé ou le message d'erreur éventuel.

### Exemples:

```

C          callp      sdExecutePc('open':'c:/test.txt':
C          '':W_NORMAL)

C          callp      sdExecutePc('print':'c:/test.txt':
C          '':W_HIDE)

C          callp      sdExecutePc('explore':'c:/winnt ':
C          '':W_NORMAL)

C          callp      sdExecutePc('find':'c:/winnt ':

```

```

C                                '':W_ NORMAL)

C                                callp    sdExecutePc('open':'c:/test.txt':
C                                '':W_ NORMAL)

```

---

## Lancement d'une page web depuis une application Silverdev

Pour lancer une page web dans un browser depuis une application silverdev, vous pouvez utiliser la fonction sdExecutePc en mettant l'url dans le paramètre file.

Exemple :

```
sdExecutePc('open':'http://www.silverdev.com':'':w_normal);
```

---

## Insertion d'une application 5250 dans un programme silverdev

Vous pouvez insérer un programme 5250 dans une fenêtre silverdev en incluant le composant CGreenScreen.

Le programme qui s'exécute dans le composant tourne dans un job différent du programme silverdev.

Ci-dessous un exemple pour lancer un programme 5250 sans avoir le prompt en 5250 :

```

c                                callp    sdSetString(F1:myGS:
c                                'host':sdGetServerIp())

c                                callp    sdSetNum(F1:myGS:
c                                'port':23)

c                                callp    sdSetString(F1:myGS:
c                                'user':'ADUVAL')

c                                callp    sdSetPwd(F1:myGS)

c                                callp    sdSetString(F1:myGS:

```



```
c                                'CurLib':pgmCurUser)

c                                callp    sdSetString(F1:myGS:
c                                'InitialPgm':'SDDMGS0')

c                                callp    sdConnect(F1:myGs:0
```

---

## Insertion d'une page web dans un programme silverdev

Pour insérer une page web dans une fenêtre Silverdev, vous pouvez utiliser le composant CWebBroser. Utilisez la fonction sdGotoUrl pour modifier l'url affichée dans le composant.

exemple :

```
C                                callp    sdGotoUrl(F1:'webBrowser1':
c                                'www.silverdev.com')
```

Remarque : Il est possible d'exécuter du javascript dans le navigateur à partir du programme silverdev, ainsi que de récupérer la valeur d'une variable dans le navigateur.

---

# Chapitre 28. Programme de service sdsrvjson

---

## Introduction :

Le programme de service sdsrvjson est livré avec le produit silverdev.  
Pour l'utiliser , vous pouvez faire référence au répertoire de liage sdsrvjson.  
Les prototypes des fonctions sont dans le membre silverdev/h/sdsrvjson

---

## Fonctions

La fonction jsonparse permet de parser du contenu au format json et de stocker le résultat dans une structure hiérarchique.

---

## Chapitre 29. WebBrowser

## Chapitre 30. Web services

Exemple d'appel d'un web service :

```

Sc */BLOCK RPGSPCIH
, *----- RPGSPCIH : H specifications (Heading)
h bnddir('SDSRVDSTR': 'SDSRVLST': 'SDSRVNODES') 1

Sc */BLOCK RPGSPCIF
, *-----, RPGSPCIF : Files declarations (F Spec.)

Sc */BLOCK RPGSPCID
, *----- RPGSPCID : Data descriptions (D Spec.)
, *-----
/copy h, sdsrvjson
/copy h, sdsrvDStr
/copy h, sdsrvnodes 2
/copy h, res_parse

```

La partie 1 permet de lier le programme à des répertoires de liage.  
Chaque répertoire de liage contient un programme de service du même nom.

SDSRVLST est un programme de service qui permet de gérer des listes de structures allouées dynamiquement. Ces listes sont comme des tableaux qui s'agrandissent automatiquement

SDSRVSTR est un programme de service qui permet de gérer des chaînes de caractères allouées dynamiquement. Cela permet de contourner la limite de la taille des variables de type alphanumérique.

SDSRVNODES est un programme de service qui permet de charger un nœud **xml** à partir d'un texte et qui permet de lire les sous nœuds et attributs facilement.

La partie 2 permet d'inclure les définitions des procédures et des types de données déclarés dans les programmes de service précédemment nommés.

La fenêtre contient un composant CWebService :



On déclare les variables suivantes :

```
D xml          ds          likeDs(DsDynStr)
D rootNode     ds          likeDs(DsNode) inz
D subNode      ds          likeDs(DsNode)
D              based(ptrNode)

D nodeName     s           500    varying
D nodeText     s           500    varying
```

Le type DsDynStr est déclaré dans sdsrvdstr

Le type DsNode est déclaré dans sdsrvnodes

```

* -----*
D Parameters      ds      based(pevtnf)
D Win              5u 0
D Evt              48
/free

// Passage de parametres au composant webservice
sdSlClear(Fl : 'webService1' : 'Params') ;
sdSlAdd(Fl : 'webService1' : 'Params' : %trim(para1)) ;
sdSlAdd(Fl : 'webService1' : 'Params' : %trim(para2)) ;
sdSlAdd(Fl : 'webService1' : 'Params' : %trim(para3)) ;
sdSetString(Fl : 'webService1' : 'url' : url);
sdSetBool(Fl : 'webService1' : 'asynchronous' : *off);

// Exécution du webservice
sdExecute(Fl : 'webService1');
xml = sdGetDynStr(Fl : 'webService1' : 'httpResponse.data');
code = sdGetInt(Fl : 'webService1' : 'httpResponse.code');
if code = 200;
  loadNodesFromText(rootNode:resultParse:xml);
  freeDynStr(xml);
  nodeName = getDynStr(rootNode.name);
  if nodeName = 'response' ;
    for i= 1 to rootNode.nodes.count;
      ptrNode = getItem(rootNode.nodes:i);
      nodeName = getDynStr(subNode.name);
      if nodeName = 'tokenDec';
        nodeText = getDynStr(subNode.text);
        jeton=nodeText;
      endif;
    endfor;
  endif;
  clearNode(rootNode);
else;
  sdShowMessage('web service failed' + X'0d25' +
    ' code : ' + %char(code) + X'0d25' +
    ' data : ' + getDynStr(xml));
  anomalie='O';
endif;
/end-free

```

Diagramme de numérotation des parties du code :

- 1 : Passage de paramètres au composant webservice (sdSlClear, sdSlAdd, sdSetString, sdSetBool)
- 2 : Exécution du webservice (sdExecute)
- 3 : Récupération des données retournées (sdGetDynStr, sdGetInt)
- 4 : Traitement des données (loadNodesFromText, freeDynStr)
- 5 : Traitement des données (nodeName = getDynStr)
- 6 : Traitement des données (if nodeName = 'response', for i= 1 to rootNode.nodes.count)
- 7 : Traitement des données (clearNode)

La partie 1 prépare le composant webService en ajoutant les paramètres de la requête.

La partie 2 appelle le web service.

La partie 3 récupère les données retournées par le web service dans une variable de type DsDynStr

La partie 4 charge un nœud (structure de type DsNode) à partir de la variable de type DsDynStr

La partie 5 libère la mémoire allouée pour la variable xml (de type DsDynStr)

La partie 6 lit le nœud rootnode de type DsNode

La partie 7 libère la mémoire allouée pour la variable rootNode