

SILVERDEV

PROGRAMMING GUIDE

R P G I L E programming

Copyright information

The information contained in this document may be modified without notification and may not be considered binding in any manner upon **EXPERIA Europe**. Software packages are supplied under licence or confidentiality agreement. The software package may only be used, copied or reproduced onto any support in compliance with the terms of the said licence or confidentiality agreement. The buyer may only make copies for backup or archiving purposes.

No part of the manual or software package may be reproduced or transmitted via any means, electronic or otherwise, including photocopying, recording or any other storage, processing or information retrieval process for any purpose other than the buyer's personal use without express, written permission from **EXPERIA Europe**.

SilverDev is a registered trademark belonging to EXPERIA Europe.

IBM, AS/400, iSeries, System i, i5, Power i are registered trademarks belonging to International Business Machines Corporation.

Windows is a registered trademark belonging to Microsoft.

All other products are registered trademarks belonging to their respective companies.

www.silverdev.com

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

Contents

Chapter 1. Server menu 6

Connection.....	6
Disconnection	7
Configuration	7

Chapter 2. Context 9

Introduction	9
Context creation.....	9
Set up a context.....	10
Select a context	11
Upgrade a context.....	14

Chapter 3. Application creation..... 15

Chapter 4. Form design 19

Palette de composants	19
Manipulate components	20
Move/Resize a component	20
Move/Size a component to the nearest pixel	21
Return to the container component	21
Select several components using the mouse.....	21
Invisible component.....	22
Forms menu.....	22
New form	22
Open form.....	22
Recent forms.....	22
New using a model.....	23
Open forms	23
Save form	24
Save form as.....	24
Save all forms	24
Compile form.....	24
Close form	26
Close all forms	26
Property inspector.....	26
Directly modifiable properties.....	26
Dialogue boxes	26
Objects	27
Sets	28
References	28
Collections	28
Events	29
Edit menu.....	30
Cut.....	30
Copy	30
Paste	31
Undo.....	31
Tab order.....	32
Background.....	34
Foreground	34
Alignment.....	34
Sizes.....	35
Text view	35
Locking	36
Themes.....	36

Activate theme	36
Applying a theme to the selection	37
Save selection in a theme.....	37

Source locking.....	38
---------------------	----

Configuration menu	38
Preferences.....	38

Help.....	39
Access from Designer.....	39
Contents	39
Index.....	40
Search	41
Help topic.....	42

Tools menu	43
------------------	----

Disconnections	43
----------------------	----

Chapter 5. Source editor 45

Introduction	45
Using a context	45
Members menu	47
Associated window	48
Compilation	48
Command	48
Integrity check.....	48
Compilation results, spools and jobs submitted	49
Code completion	50
Functions	50
Window tree structure.....	51
List of functions and moments	52
Bookmarks	53
Shortcuts	54
Searching an expression in a source file	54

Chapter 6. Generator 58

RPG IV source	58
*/BLOCK	58
Event managers: */EVENT.....	60
OptionsGeneration	62
QRPGLESRC protection	63
PSVDCMP.....	64
Generated source.....	65
Basic program elements	65
sdStart	66

Chapter 7. RT mode or LR mode 68

Handling the components 70

sdSetString, sdGet	70
SdSetNum, sdGetNum	70
sdSetBool, sdgetBool	71

Chapter 8. Debug 72

Server side debug with designer	72	Formatting an occurrence	117
Launching a debug	72	Group printing	117
Choosing the program	73	Drawing a table	121
Choosing the module	74	Value reports:	122
Choosing the view	75	Page breaks	122
Functions in a view	76	Formatting in the middle of a string	122
Debug command field	78	PDF printing	123
Click and Double clicks	79	Selecting the printer	125
Debug log	79	Imposing a printer	125
List of functions	80	FromPage ToPage	126
Adding a module	80	mm/Pixel conversion	126
Ending the debug	81		
Debug attributes	81	Chapter 15. Tools menu.....	128
Entry point	82	Command	128
Re start a program	83	Job log	131
Client side debug	84	Job list	131
Trace tab sheet	86	Spool list	131
Script tab sheet	87	Library list	132
Microsoft Script Debugger	89	Assistant	132
Server side debug with 5250	91	Database diagrams	135
Entry point	93	Component list	138
Debug instructions	94	Server event list	138
ATTR	94	Local event list	138
Break	95	Tree structure	139
Sbreak	95	Decompilation	141
Clear	95	Explorer	142
EVAL	96	Versionning	142
QUAL	96	Compare sources	143
Step Statement	97	Compare contents	143
Watch	97		
Chapter 9. Errors	98	Chapter 16. Common mistakes	144
Server side errors	98	Chapter 17. Solutions replaced	145
Client side errors	99	CStringGrid component	145
Just-in-time debugging:	99	SdSelectColor, sdSelectFile, sdPrintDlg	145
Chapter 10. JOBD programs in MSGW ...	101	SdSetSet, sdAddSet, sdDelSet	145
Chapter 11. Screen sources	102	sdAddNode	145
Chapter 12. Events.....	104	Chapter 18. Disconnections	147
Event manager	104	Chapter 19. Advanced technicals.....	148
Event parameters	105	Functions of List category	149
Dynamic allocation of events	108	Optimisation	149
Chapter 13. Modal forms	110	SFL : Copying data locally	149
Introduction	110	Local events	149
ModalResult	110	Multi form applications	149
Example	110	Mdi applications	149
Execution only	112	Multi forms applications with tabs	149
Chapter 14. Printouts.....	114	Multi form occurrences	149
Simple printing	114	Multi form occurrences, sdsrvlst service program	149
Print preview	114	Dialog box functions	149
Orientation	115	PC file category functions	149
Printing a list	115	Collections category functions	149
CRepLabel	116	TStrings category functions	149
CRepMemo:	116	Set category functions	149
CRepChart:	116	Miscellaneous functions	149
		Tips	149
		Web services	149
		Screen translation	149
		Image category functions	149
		List of forms and events	149
		Colors	149
		Anchoring components	149
		Drag and drop operations	149
		Screen size adaptation	149

ADO.....	149	Sdsrvjson service program	150
Exit points	150	WebBrowser	150
Interactions with external programs.....	150		

Chapter 1. Server menu

Connection

The "Server/Connection" menu enables a connection to be made with the SilverDev server.

You can work on a form off-line but to save or open a form, you must be logged on.

If you try to complete an operation that requires connection while you are off-line, Designer will open a dialogue box to enable connection.

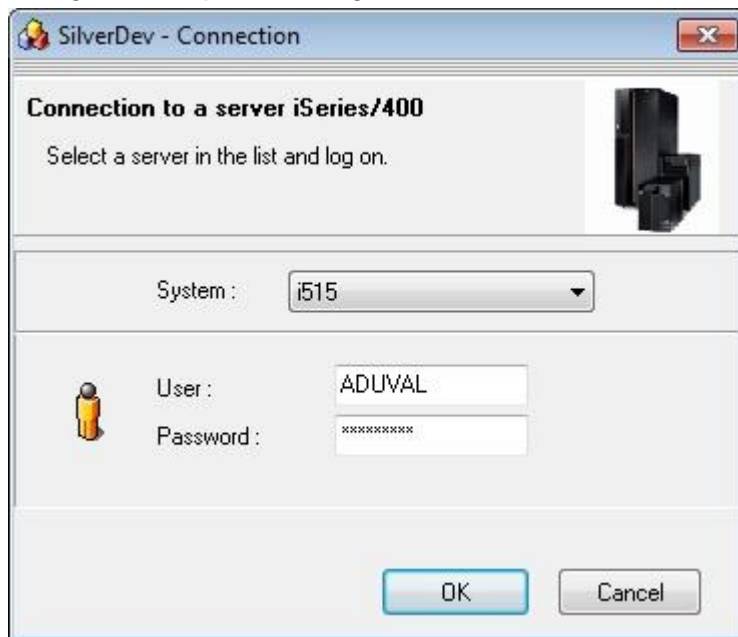


Figure 1

The list of servers displayed in the scroll list comes from an ini file stored on the client PC. This list can be modified via the "Server/Configuration" menu.

When you are connected, a job is allocated to you on the server.

The number of this job is shown at the bottom left of Designer.

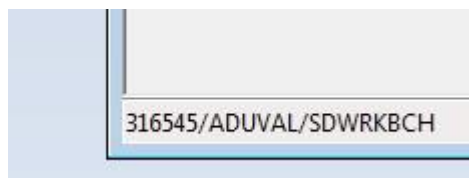


Figure 2

Disconnection

The "Server/Disconnection" menu enables disconnection. If you close Designer, it will disconnect automatically.

Configuration

To ensure faster connection, you can save the data enabling access to the SilverDev server. Several servers can be configured.

This configuration part is common to the MyDesk.exe configuration.

Use the Server/Configuration menu.

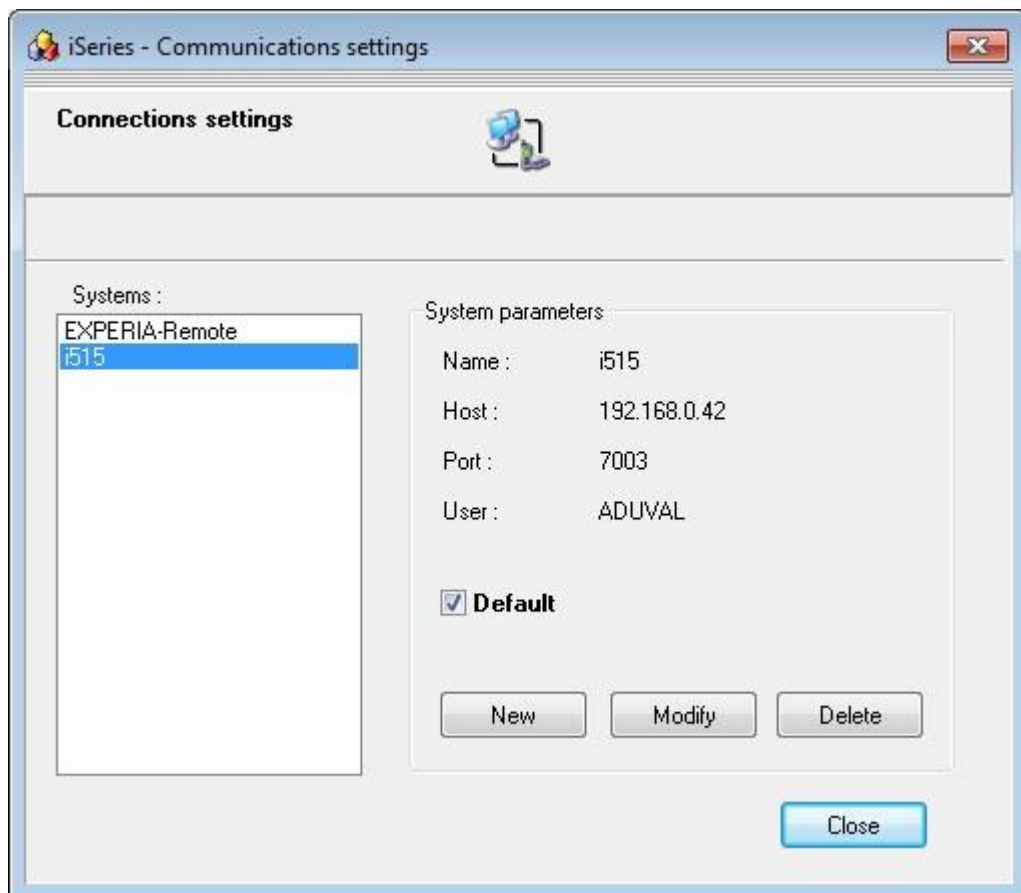
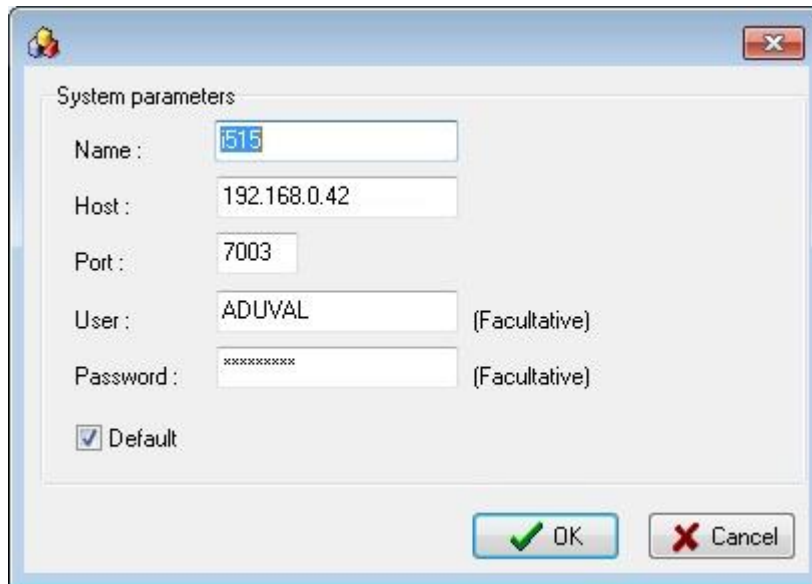


Figure 3

To add a system name, use the New button and to change the data related to a system, use the Modify button.

**Figure 4**

In the "Name" field, enter the name you want to use. This name will be proposed in the connection dialogue box.

Then enter the IP address of the System i, with the port number on which the SilverDev server was started.

Enter the profile under which you want to log on.

You do not have to enter a password at this stage.

If entered, the password will be stored (encoded) in the SilverDev.ini file.

This configuration enables entry of information for several SilverDev servers as well as for a single server with different profiles.

Chapter 2. Context

Introduction

A context is an environment of work.

Context creation

To create a context, use the SILVERDEV/CRTSDCTX command.
You can also use the "Tools/Create a context..." menu.



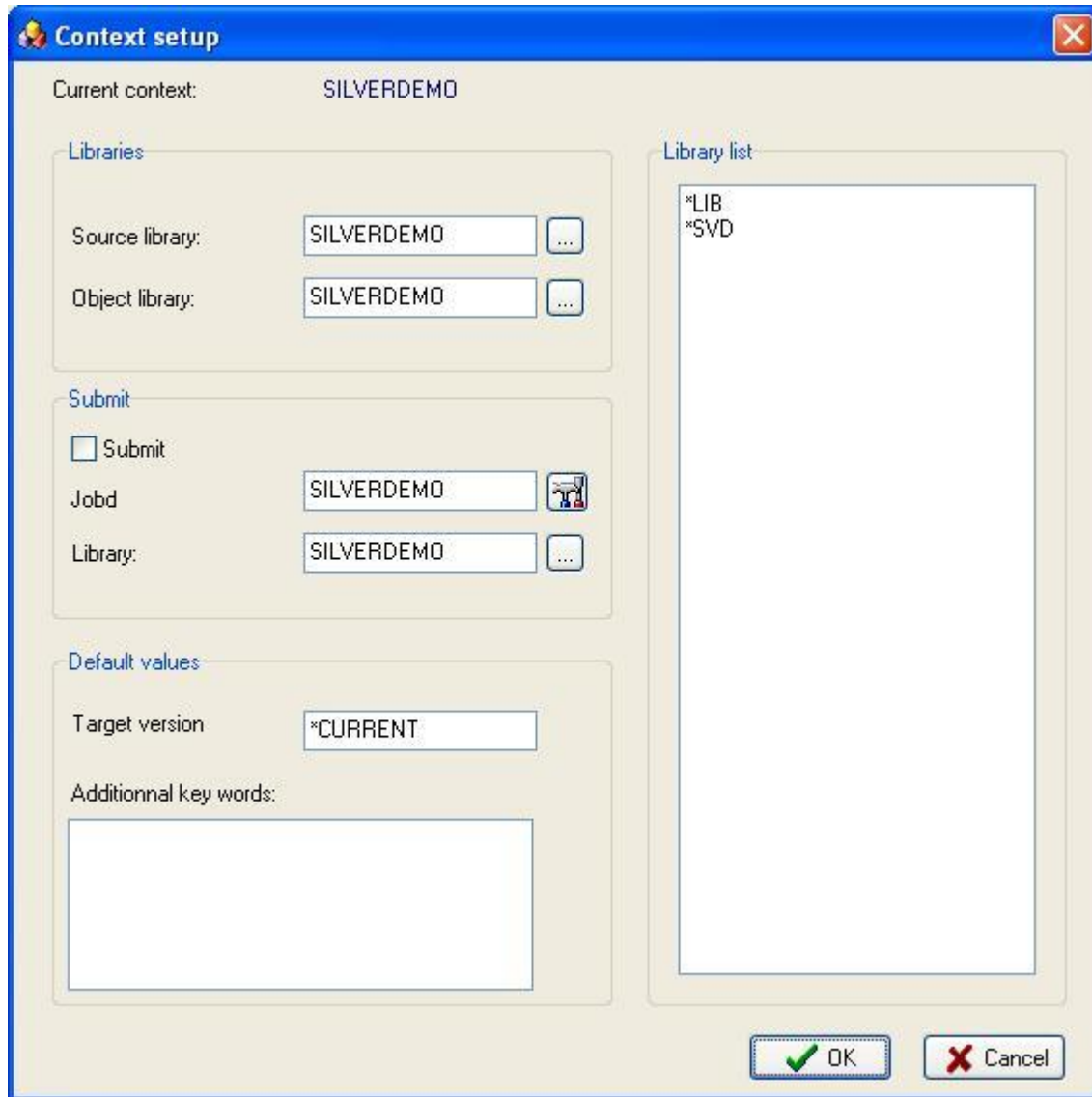
This command creates three files in the library received as a parameter.
These three files are :
PSVDSPA,PSVDSPG et PSVDSPL

The PSVDSPA file store general information on the context.
The PSVDSPG file store the context's program list.
The PSVDSPL file store the context's library list.

Set up a context

To set up a context, use the "Tools/Set up a context..." menu.

When you chose a context in the list, the following window is displayed :



The "Context setup" dialog box is shown with a blue title bar and a close button. It contains several sections:

- Current context:** A label followed by the text "SILVERDEMO".
- Libraries:** A section containing two labels, "Source library:" and "Object library:", each followed by a text box containing "SILVERDEMO" and a browse button (...).
- Submit:** A section containing a checkbox labeled "Submit", a label "Jobd" followed by a text box containing "SILVERDEMO" and a browse button (...), and a label "Library:" followed by a text box containing "SILVERDEMO" and a browse button (...).
- Default values:** A section containing a label "Target version" followed by a text box containing "*CURRENT", and a label "Additional key words:" followed by a large empty text area.
- Library list:** A large list box on the right side containing the text "*LIB" and "*SVD".
- Buttons:** At the bottom right, there are two buttons: "OK" with a green checkmark icon and "Cancel" with a red X icon.

Source library: indicates the library that contains the sources. When creating a program from option 1 of the silver menu, two sources are generated: one in QTXTLESRC for moments and one QRPGLSRC for the final program.

Object library: indicates the library where the objects will be compiled.

JOBID submission: indicates the name of the job queue in which the generation and compilation jobs are placed.

Default values: this section enables entry of compilation parameters in general. It is however possible to redefine the compilation parameters for each SilverDev program.

Library list : This section enables entry of a library list wich is assigned when this context is selected.

You can use the special values *LIB and *SVD.

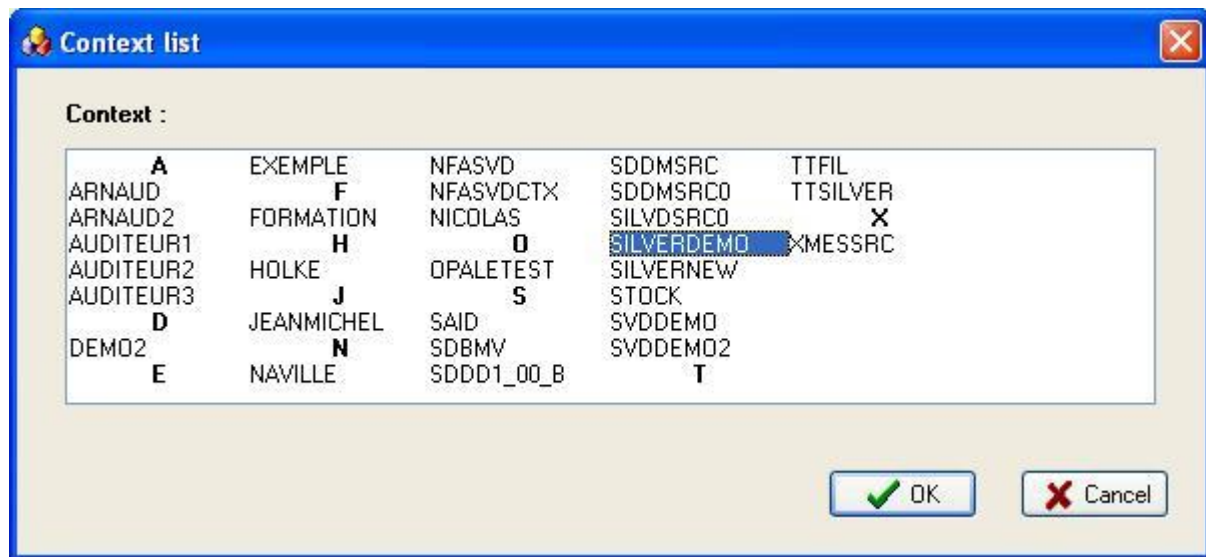
*LIB refers to the library of PSVDSPA,PSVDSPG and PSVDSPL files

*SVD refers to the library of Silverdev.

Select a context

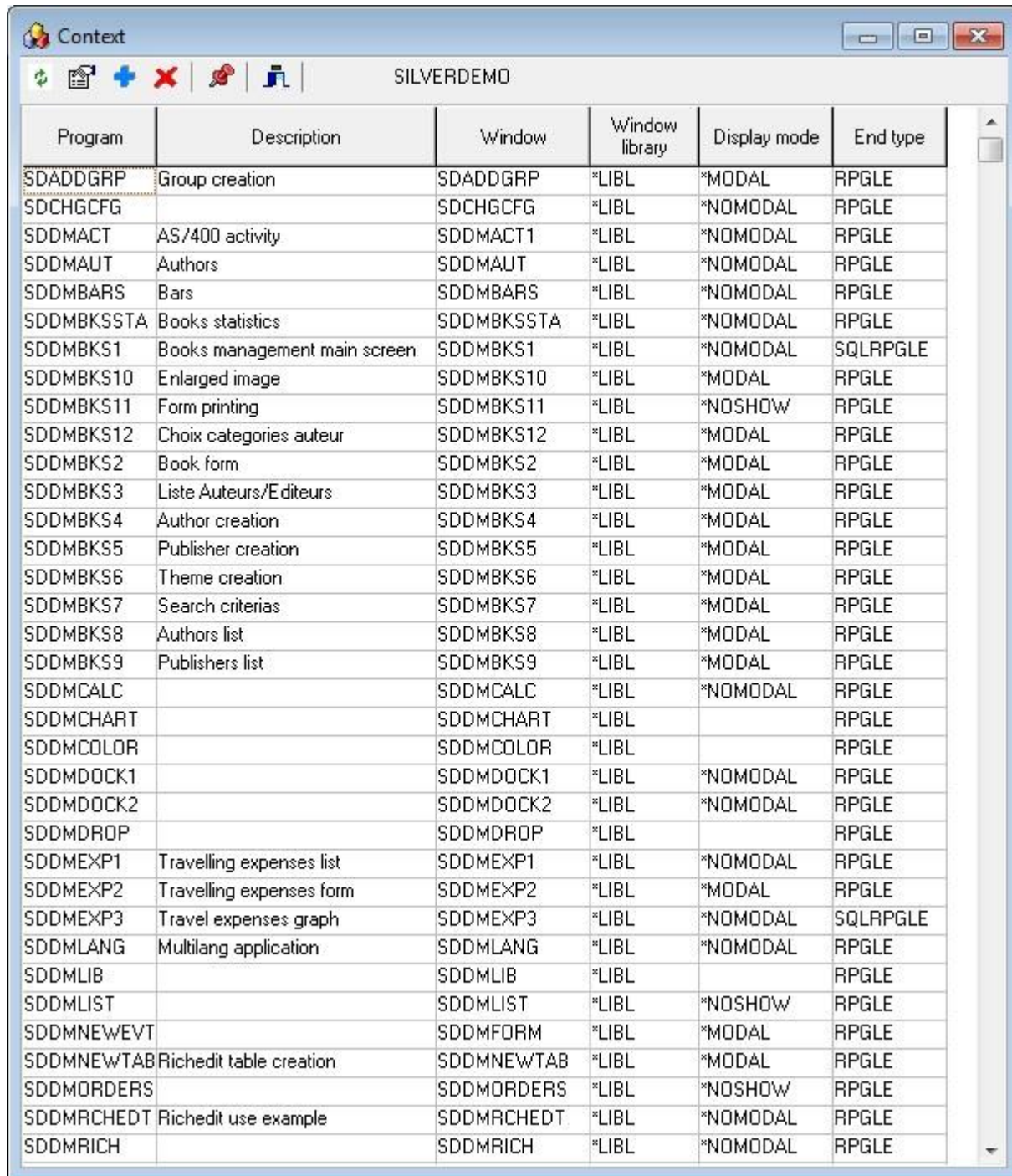
Use the "Tools/context..." menu to display the context list.

This list is created by finding PSVDSPA files.



Click on OK to select a context.

The context's program list is displayed in a grid.



The screenshot shows the 'Context' application window with the title bar 'SILVERDEMO'. The window contains a table with 6 columns: Program, Description, Window, Window library, Display mode, and End type. The table lists various programs and their properties.

Program	Description	Window	Window library	Display mode	End type
SDADDGRP	Group creation	SDADDGRP	*LIBL	*MODAL	RPGLE
SDCHGCFG		SDCHGCFG	*LIBL	*NOMODAL	RPGLE
SDDMACT	AS/400 activity	SDDMACT1	*LIBL	*NOMODAL	RPGLE
SDDMAUT	Authors	SDDMAUT	*LIBL	*NOMODAL	RPGLE
SDDMBARS	Bars	SDDMBARS	*LIBL	*NOMODAL	RPGLE
SDDMBKSSTA	Books statistics	SDDMBKSSTA	*LIBL	*NOMODAL	RPGLE
SDDMBKS1	Books management main screen	SDDMBKS1	*LIBL	*NOMODAL	SQLRPGLE
SDDMBKS10	Enlarged image	SDDMBKS10	*LIBL	*MODAL	RPGLE
SDDMBKS11	Form printing	SDDMBKS11	*LIBL	*NOSHOW	RPGLE
SDDMBKS12	Choix categories auteur	SDDMBKS12	*LIBL	*MODAL	RPGLE
SDDMBKS2	Book form	SDDMBKS2	*LIBL	*MODAL	RPGLE
SDDMBKS3	Liste Auteurs/Editeurs	SDDMBKS3	*LIBL	*MODAL	RPGLE
SDDMBKS4	Author creation	SDDMBKS4	*LIBL	*MODAL	RPGLE
SDDMBKS5	Publisher creation	SDDMBKS5	*LIBL	*MODAL	RPGLE
SDDMBKS6	Theme creation	SDDMBKS6	*LIBL	*MODAL	RPGLE
SDDMBKS7	Search criterias	SDDMBKS7	*LIBL	*MODAL	RPGLE
SDDMBKS8	Authors list	SDDMBKS8	*LIBL	*MODAL	RPGLE
SDDMBKS9	Publishers list	SDDMBKS9	*LIBL	*MODAL	RPGLE
SDDMCALC		SDDMCALC	*LIBL	*NOMODAL	RPGLE
SDDMCHART		SDDMCHART	*LIBL		RPGLE
SDDMCOLOR		SDDMCOLOR	*LIBL		RPGLE
SDDMDOCK1		SDDMDOCK1	*LIBL	*NOMODAL	RPGLE
SDDMDOCK2		SDDMDOCK2	*LIBL	*NOMODAL	RPGLE
SDDMDROP		SDDMDROP	*LIBL		RPGLE
SDDMEXP1	Travelling expenses list	SDDMEXP1	*LIBL	*NOMODAL	RPGLE
SDDMEXP2	Travelling expenses form	SDDMEXP2	*LIBL	*MODAL	RPGLE
SDDMEXP3	Travel expenses graph	SDDMEXP3	*LIBL	*NOMODAL	SQLRPGLE
SDDMLANG	Multilang application	SDDMLANG	*LIBL	*NOMODAL	RPGLE
SDDMLIB		SDDMLIB	*LIBL		RPGLE
SDDMLIST		SDDMLIST	*LIBL	*NOSHOW	RPGLE
SDDMNEWEVT		SDDMFORM	*LIBL	*MODAL	RPGLE
SDDMNEWTAB	Richedit table creation	SDDMNEWTAB	*LIBL	*MODAL	RPGLE
SDDMORDERS		SDDMORDERS	*LIBL	*NOSHOW	RPGLE
SDDMRCHEDT	Richedit use example	SDDMRCHEDT	*LIBL	*NOMODAL	RPGLE
SDDMRICH		SDDMRICH	*LIBL	*NOMODAL	RPGLE

Figure 5

To change a program properties , use the  button, et to add a program to the list, use the  button.

In modification mode, you have the following window :

Silverdev program

General

Program name: End type: ☒ RT ☐ LR

Type:

Description:

Created by: the:

Main window

Library: ... Display mode: ☒ Normal ☐ Modal ☐ No display

Window: ...

source:

Compiling

Target version:

Additional key words:

Configuration value:

☐ Ignore config key words

Program name: Name of the program generated.

Type: Type of source.

Description: Description linked to the object generated (*PGM).

Main window: Name of the window used to generated the *EVENT blocks.

Window Display Mode: 3 possible values:

*NOMODAL by default: the window is displayed normally.

*MODAL: the window is displayed in modal mode.

*NOSHOW: the window is not displayed, used for print models.

Type of end: LR or RT.

The default value is RT.

The main code of a silverdev program is:

C	callp	sdStart(%paddr('INIT'))
C	eval	*inrt = *on

Target version: value of the TGTRLS parameter for compilation.

Additional key words: enter any parameter for the compilation command.

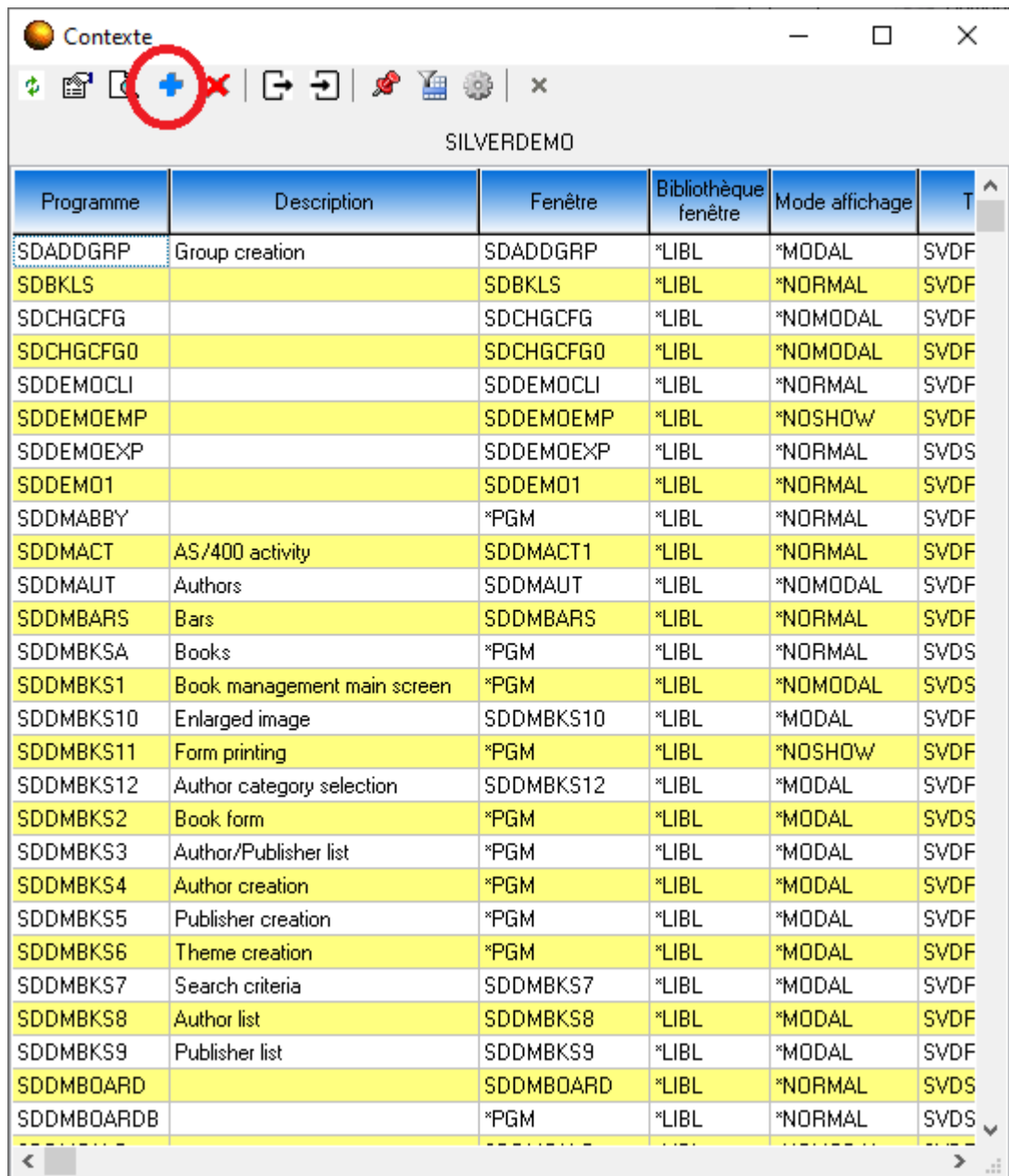
Ignore config. key words: determines whether or not the compilation parameters in the general configuration for the context are used for this program.

Upgrade a context

See document Administration chapter compatibility between releases.

Chapter 3. Application creation

From the context window, click on the following button :



The screenshot shows the 'Programme Silverdev' dialog box with the 'Général' tab selected. The dialog has three tabs: 'Général', 'Compilation', and 'Commandes'. The 'Général' tab contains two sections: 'Général' and 'Fenêtre principale'. In the 'Général' section, 'Nom programme' is 'TEST', 'Type' is 'SVDRPG', 'Description' is empty, and 'Créé par' is 'ADUVAL'. The 'Type de fin' section has 'RT' selected. In the 'Fenêtre principale' section, 'Bibliothèque' is '*LIBL', 'Fenêtre' is '*PGM', and 'Source' is '*DEFAULT'. The 'Mode d'affichage' section has 'Normal' selected. At the bottom right are 'OK' and 'Annuler' buttons.

Programme Silverdev

Général Compilation Commandes

Général

Nom programme : TEST

Type : SVDRPG

Description :

Créé par : ADUVAL le :

Type de fin

☒ RT

☐ LR

Fenêtre principale

Bibliothèque : *LIBL

Fenêtre : *PGM

Source : *DEFAULT

Mode d'affichage

☒ Normal

☐ Modal

☐ Ne pas afficher

OK Annuler

The created program is added to the list. Double click on it to open rpg source and associated form.

SILVERDEMO						
Programme	Description	Fenêtre	Bibliothèque fenêtre	Mode affichage	Type	Chemini
SDDMXFER		SDDMXFER	*LIBL	*NORMAL	SVDRPG	
SDDMXFER2		SDDMXFER2	*LIBL	*NOMODAL	SVDRPG	
SDDM5250	5250 pgm call example	SDDM5250	*LIBL	*NOMODAL	SVDRPG	
SDDYNMNU	Dynamic menu creation	SDDYNMNU	*LIBL	*NOMODAL	SVDRPG	
SDFILEREF	Programmes utilisant un fichier	*PGM	*LIBL	*NORMAL	SVDSQLRPG	*DEFA
SDLIBLIST		*PGM	*LIBL	*NORMAL	SVDRPG	*DEFA
SDLSTMSGF	Display message file	*PGM	*LIBL	*NORMAL	SVDRPG	*DEFA
SDTSTCMD		SDTSTCMD	*LIBL	*NOMODAL	SVDRPG	
SDUPDGRP	Group modification	SDADDGRP	*LIBL	*MODAL	SVDRPG	
SDWRKADM	Administrator management	SDWRKADM	*LIBL	*NOMODAL	SVDRPG	
SDWRKGRP	Group management	SDWRKGRP	*LIBL	*NOMODAL	SVDRPG	
SDWRKMBR	Group member management	SDWRKMBR	*LIBL	*MODAL	SVDRPG	
SDZOOMIMG		SDZOOMIMG	*LIBL	*NORMAL	SVDRPG	
TEST		*PGM	*LIBL	*NORMAL	SVDRPG	*DEFA

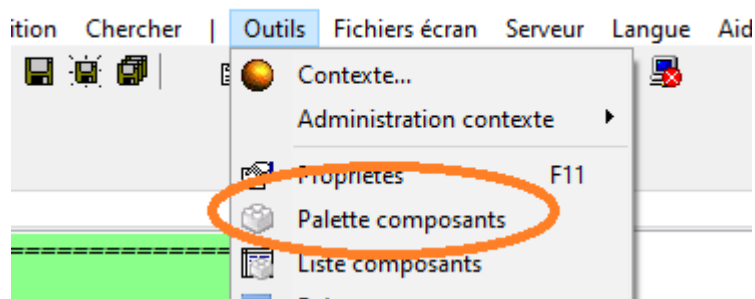
Form and rpg source are open :

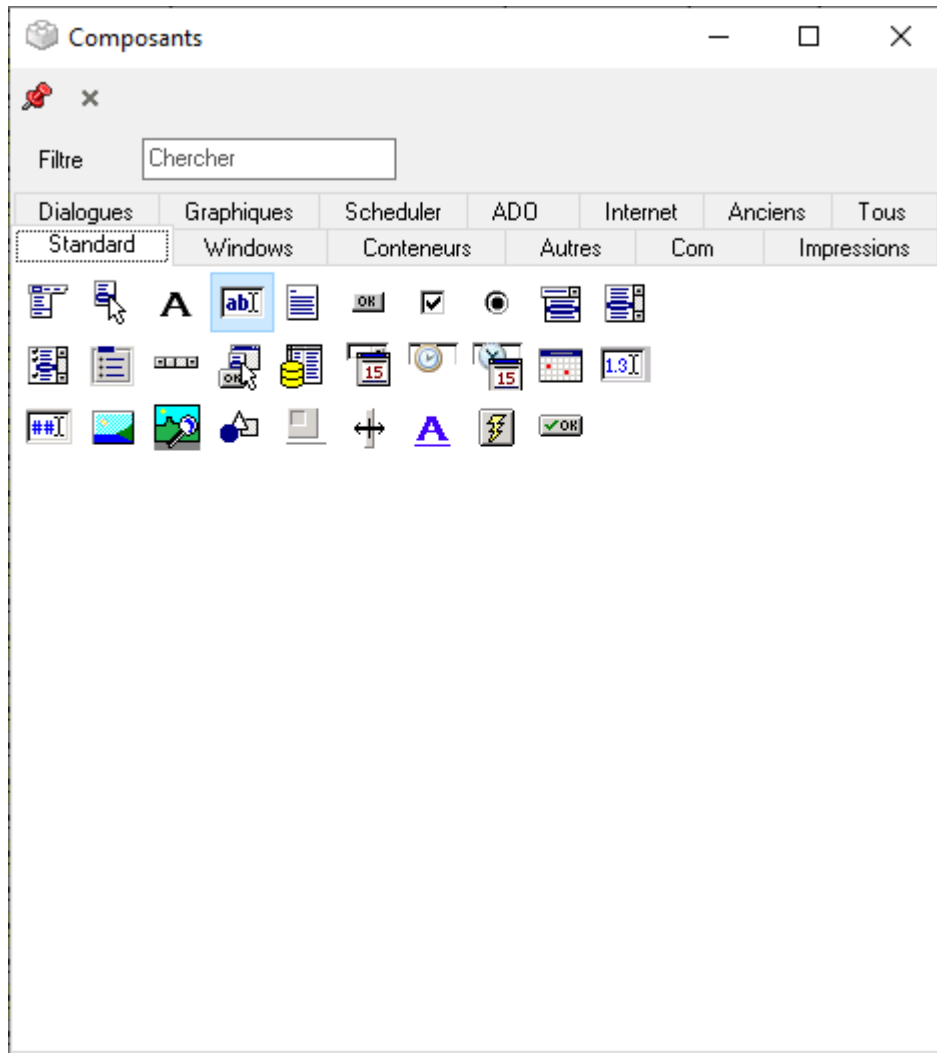
The screenshot shows the SilverDev IDE interface. The main editor displays the RPG source code for the TEST program, which includes comments and code blocks for various sections like Title, Author, Date, Project, Description, Technical data, Main SDF, and various RpgSPC blocks (RpgSPCIN, RpgSPCIF, RpgSPCID, RpgSPCII, RpgPARM, RpgPGMSTART, RpgBEFORECREATE, RpgAFTERCREATE, RpgBEFORESHOW). The Contexte window on the right shows a list of programs and their associated libraries, including SDDMPRTT, SDDMRCHDT, SDDMRICH, SDDMRP, SDDMSAVF, SDDMSAVF2, SDDMSAVF3, SDDMSKINS, SDDMSPLA1, SDDMSPLA2, SDDMSPLF, SDDMSPLF2, SDDMSPLF3, SDDMSPLF4, SDDMSPLF5, SDDMSPLF6, SDDMSPLF7, SDDMSPLF8, SDDMSPLF9, SDDMSPLF10, SDDMSPLF11, SDDMSPLF12, SDDMSPLF13, SDDMSPLF14, SDDMSPLF15, SDDMSPLF16, SDDMSPLF17, SDDMSPLF18, SDDMSPLF19, SDDMSPLF20, SDDMSPLF21, SDDMSPLF22, SDDMSPLF23, SDDMSPLF24, SDDMSPLF25, SDDMSPLF26, SDDMSPLF27, SDDMSPLF28, SDDMSPLF29, SDDMSPLF30, SDDMSPLF31, SDDMSPLF32, SDDMSPLF33, SDDMSPLF34, SDDMSPLF35, SDDMSPLF36, SDDMSPLF37, SDDMSPLF38, SDDMSPLF39, SDDMSPLF40, SDDMSPLF41, SDDMSPLF42, SDDMSPLF43, SDDMSPLF44, SDDMSPLF45, SDDMSPLF46, SDDMSPLF47, SDDMSPLF48, SDDMSPLF49, SDDMSPLF50, SDDMSPLF51, SDDMSPLF52, SDDMSPLF53, SDDMSPLF54, SDDMSPLF55, SDDMSPLF56, SDDMSPLF57, SDDMSPLF58, SDDMSPLF59, SDDMSPLF60, SDDMSPLF61, SDDMSPLF62, SDDMSPLF63, SDDMSPLF64, SDDMSPLF65, SDDMSPLF66, SDDMSPLF67, SDDMSPLF68, SDDMSPLF69, SDDMSPLF70, SDDMSPLF71, SDDMSPLF72, SDDMSPLF73, SDDMSPLF74, SDDMSPLF75, SDDMSPLF76, SDDMSPLF77, SDDMSPLF78, SDDMSPLF79, SDDMSPLF80, SDDMSPLF81, SDDMSPLF82, SDDMSPLF83, SDDMSPLF84, SDDMSPLF85, SDDMSPLF86, SDDMSPLF87, SDDMSPLF88, SDDMSPLF89, SDDMSPLF90, SDDMSPLF91, SDDMSPLF92, SDDMSPLF93, SDDMSPLF94, SDDMSPLF95, SDDMSPLF96, SDDMSPLF97, SDDMSPLF98, SDDMSPLF99, SDDMSPLF100.

Chapter 4. Form design

Palette de composants

Si la palette de composants n'est pas affichée, utilisez le menu outils/palette de composants :



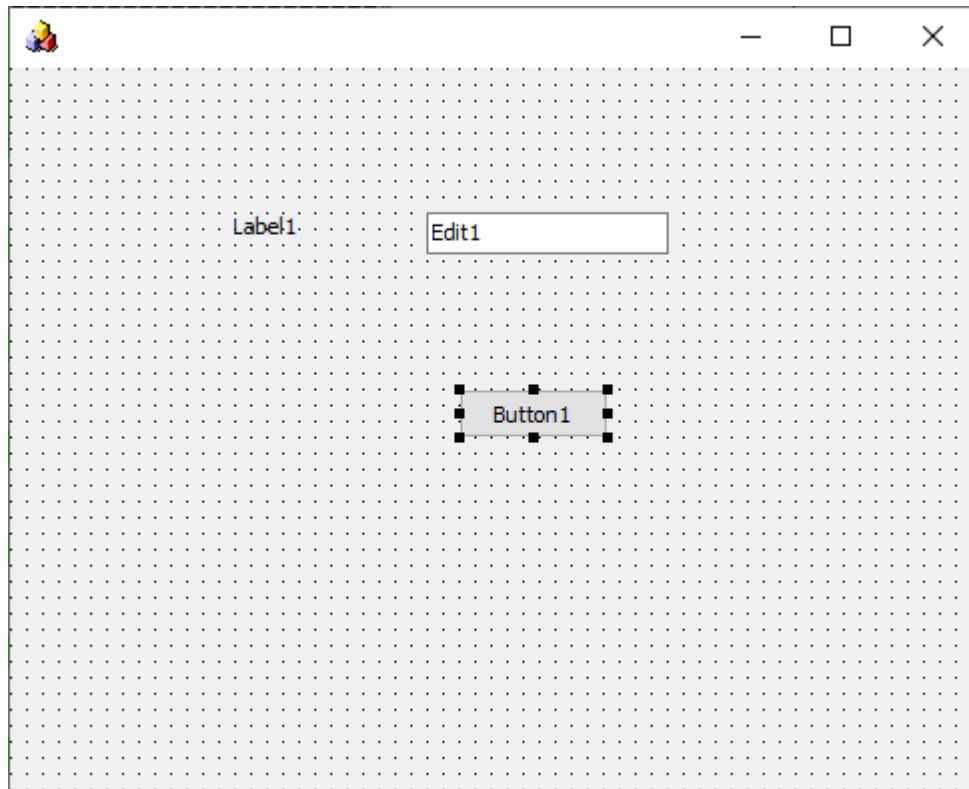


Sélectionnez un type de composant dans la palette et cliquez sur la fiche pour créer ce composant.

Manipulate components

Move/Resize a component

Click on a component to select it.



Click on the component to select it, hold the left button pressed and move the component.

To resize the component, click on one of the black squares around the component.

A screen is created using the Designer.exe tool.

Each screen corresponds to a source.

Move/Size a component to the nearest pixel

To move a component, select it, hold down CTRL then use the arrow keys to move it. You can do this if several components are selected.

To size a component, select it, hold down SHIFT then use the arrow keys to modify it. You can do this if several components are selected.

Return to the container component

If a component fills all the space in its container component, the container cannot be selected by clicking; just use the ESC key to return to the container component.

Select several components using the mouse

To select several components at the same time, hold down the CTRL key and click the left mouse button at the same time, then move the mouse to draw a rectangle on the form. When you release, all the components within the rectangle will be selected.

Another option is to hold down the SHIFT key and click on each of the desired components one at a time.

The properties common to all these components can thus be defined in one go.

Invisible component

If a component is not visible in Designer because its left property is too high or because it is underneath another component, you can use the tree structure or the property inspector to select it.

Forms menu

New form

The "Forms/New form" menu enables creation of a new form.

A new form can be created even if other forms are being used. Designer allows users to work on several forms at the same time.

Open form

The "Forms/Open form" menu is used to open existing forms. The screen sources are stored on the ifs but you do not need access to the ifs via the network. However, the server must be running and Designer must be connected to the server.

If you try to open a form without being connected, Designer will open a dialogue box asking you to log on.

Several source screens can be open at the same time.

If you ask for a source screen that is already open, it will be reactivated. Source screens cannot be opened twice.

When a source file is opened in Designer the file is locked.

A source cannot be opened by two users at the same time in Designer.

Recent forms

To help you find the screens on which you have been working recently, the file paths of the last 20 screens are saved. To open one of these screens, double-click on its name.

Use the Delete menu to remove the screens from your list of recent screens. (This does not delete the screens from the ifs).

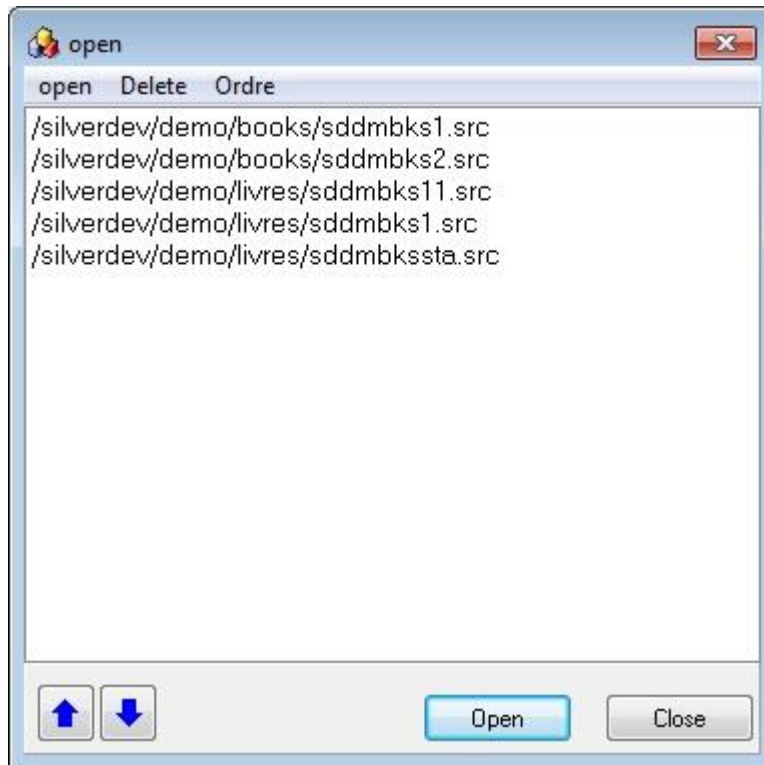


Figure 6

New using a model

The "Forms/New form based on" menu enables creation of a new form on the basis of an existing form.

Open forms

The "Forms/Open forms" menu lists all the open forms.
The active one is in bold.

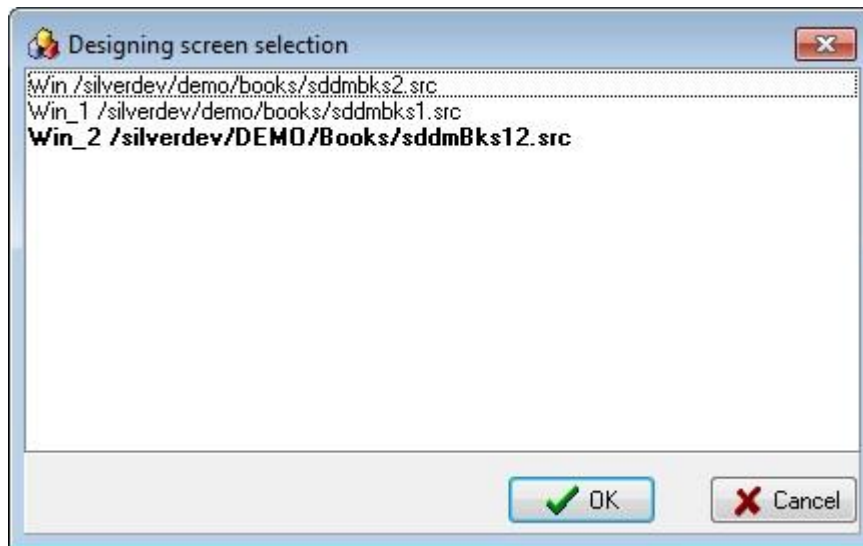


Figure 7

Save form

The "Forms/Save form" menu enables the form to be saved on the ifs. You do not need to have access to the ifs via the PC network neighbourhood. However, the server must be running and Designer must be connected to the server. The file path for saving the form is displayed at the top of Designer. If the current form has never been saved, a dialogue box will open as if you had used the "File/Save as..." menu.

Save form as...

The "Forms/Save form as..." menu enables you to select the location of the current form to be saved. If you open a form in one location and use "Save as...", any subsequent saves will be made to this new location.

Save all forms

The "Forms/Save all forms" menu enables all the open forms to be saved. If a window has no associated source name, Designer will ask you where to save the form.

Compile form

The "Forms/Compile form" menu enables generation of the screen object on the System i. This operation requires connection to the SilverDev server. Designer will always ask you where you want to compile the screen. This menu is only accessible if the current form has a source name and has not been modified.

A dialogue box is opened. By default, the File field displays the source file name. The library has no default value but the text entered will be memorised from one compilation to another, even for different screens (unless Designer is restarted in between).

The right button of the field enables selection of a library.

If you start typing in the name of a library, only those beginning with the same letters will be displayed when you press "...".



Figure 8

Note:

By default, the "Write over the existing file" tick box is clear.

If you want this box to be ticked by default, go to the "preferences/config" menu

Compilation control

If you want to control compilation, you can add an exit program to the exit point SVDCTRLCPLSDF by using WRKREGIN command.

C	*entry	plist		
C		parm	JobUser	10
C		parm	Lib	10
C		parm	UsrSpc	10
C		parm	Ret	1
C		parm	Texte	256

The first three parameters are on input, the next two on output.

If the Ret parameter is 'N' the compilation will be abandoned and the text in the Text field will be displayed in Designer.

Close form

Close the current form.

If the current form has been modified, a dialogue box will ask if you want to save it.

Press "YES" to save the form before closing. Press "NO" to close the form without saving. Press "CANCEL" to cancel the closing operation.

Close all forms

Close all open forms.

For each modified form, Designer will ask if you want to save it.

If you choose "YES", Designer will save the form and continue to close the other forms.

If you choose "NO", Designer will close the form without saving and continue to close the other forms.

If you choose "CANCEL", Designer will stop closing the forms.

Property inspector

The property inspector enables modification of the properties of the components selected on the current form.

Directly modifiable properties

Some properties can be modified directly in the inspector.

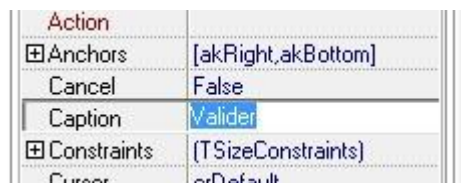


Figure 9

Dialogue boxes

Others require use of a dialogue box.

To display this dialogue box, click on the ellipsis button in the inspector. These properties are shown in blue.

Name	Image1
ParentShowHi	False
Picture	TPicture
PopupMenu	

Figure 10



Figure 11

Objects

Some properties can be developed. In general, these are object or set properties.

Enabled	True
Font	TFont
Charset	DEFAULT_CHARSET
Color	clWindowText
Height	-13
Name	MS Sans Serif
Pitch	fpDefault
Size	10
Style	[]
Glvph	TBitmap

Figure 12

Sets

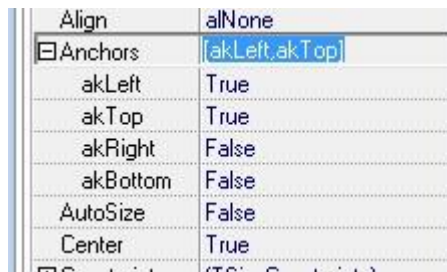


Figure 13

References

The properties in red are the properties that refer to another component in the form. A scroll list is displayed and the component can be selected.

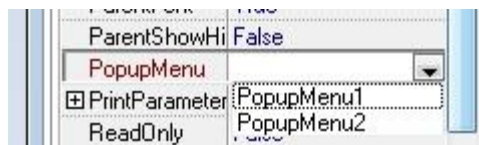


Figure 14

Collections

Some properties are collections, i.e. lists of objects.

For example, component CSFL has a collection that corresponds to the component columns.

To modify a collection, click on the button opposite the name of the property.

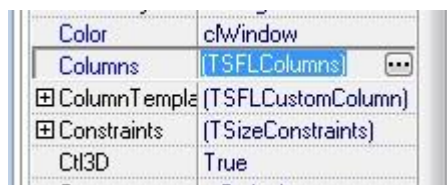


Figure 15

The items of the collection are then displayed.

Select one of the items from the collection and modify the properties of this item in the inspector.

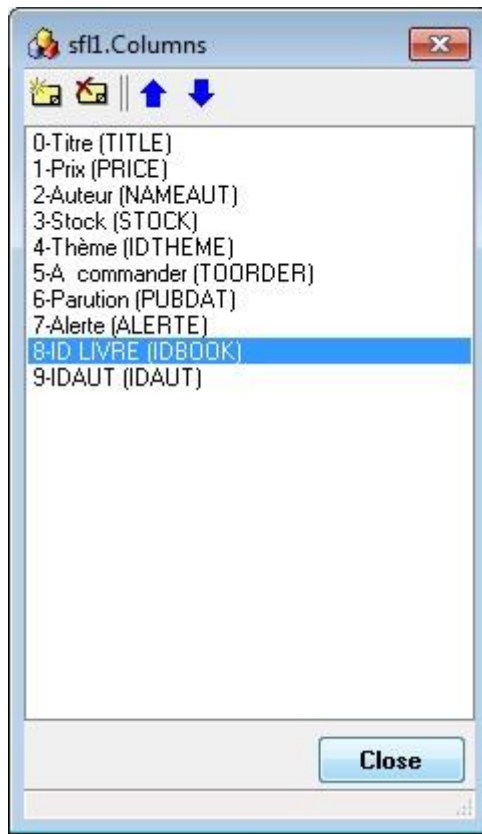


Figure 16

Events

The property inspector also enables selection of the events to be notified to the RPG program.

Use the second tab and tick the events to be notified.

In the program, you will use the `sdSetCallBack` function to link these events to RPG procedures. If you use the generator, calls to the `sdSetCallBack` functions will be generated automatically.

```
c          callp      sdSetCallBack(F1
c                               : 'Button1.OnClick'
c                               : %paddr (
c                               'BUTTON1_ONCLICK' ) )
```

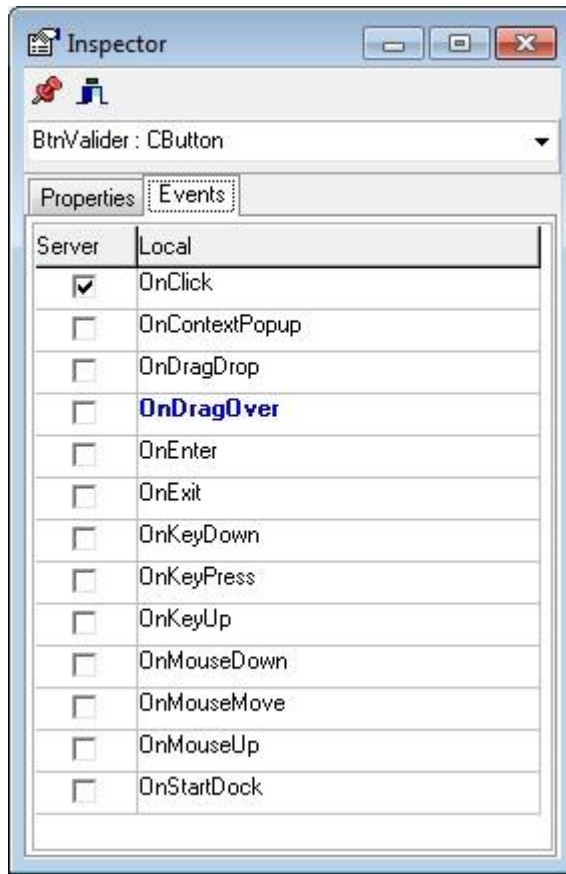


Figure 17

Opposite the tick boxes, you can enter code to be run locally. This function will be presented in **Erreur ! Source du renvoi introuvable.**

Edit menu

Cut

The "Edit/Cut" menu enables deletion of the current components on the active form and places them on the clipboard.

Current components are identified by black squares around the component.

Copy

The "Edit/Copy" menu enables the current components to be copied to the clipboard.

Current components are identified by black squares around the component.

Only the component properties are copied, not the events.

Paste

The "Edit/Paste" menu enables the components copied to the clipboard to be pasted on the current form. If a component able to contain other components is selected for pasting (CPanel, CGroupBox, etc.), the components will be copied into it.

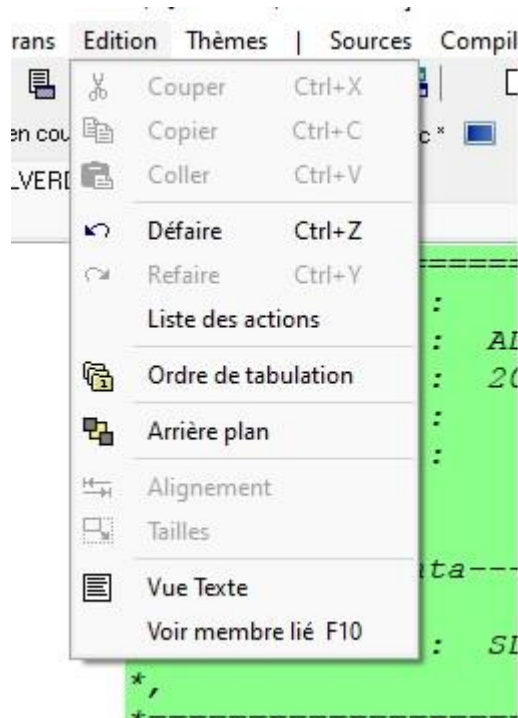
All the components on the clipboard whose names are the same as an existing component will be renamed automatically.

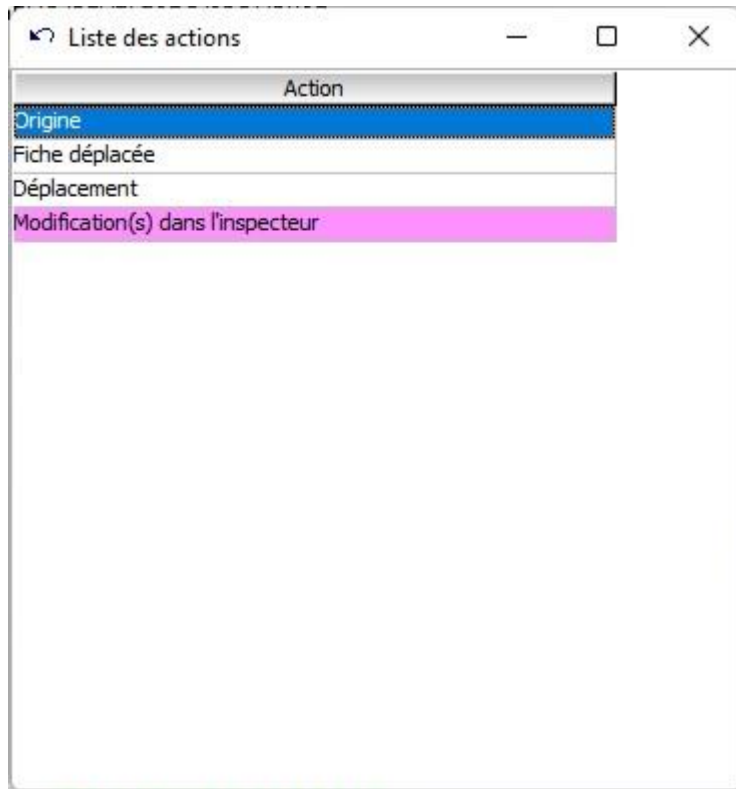
Undo

Use the undo menu to come back to a previous state. Shortcut is ctrl + z

Note : You can also redo with the redo menu, shortcut is ctrl + Y

You can display all actions that you can undo :





Double click on a line to go back to the state after this action.

Note : When you go back to a state n , states $n+1$ etc are still available, but when you change the form, state $n+1$ is replaced.

Tab order

The "Edit/Tab order" menu opens the following dialogue box:

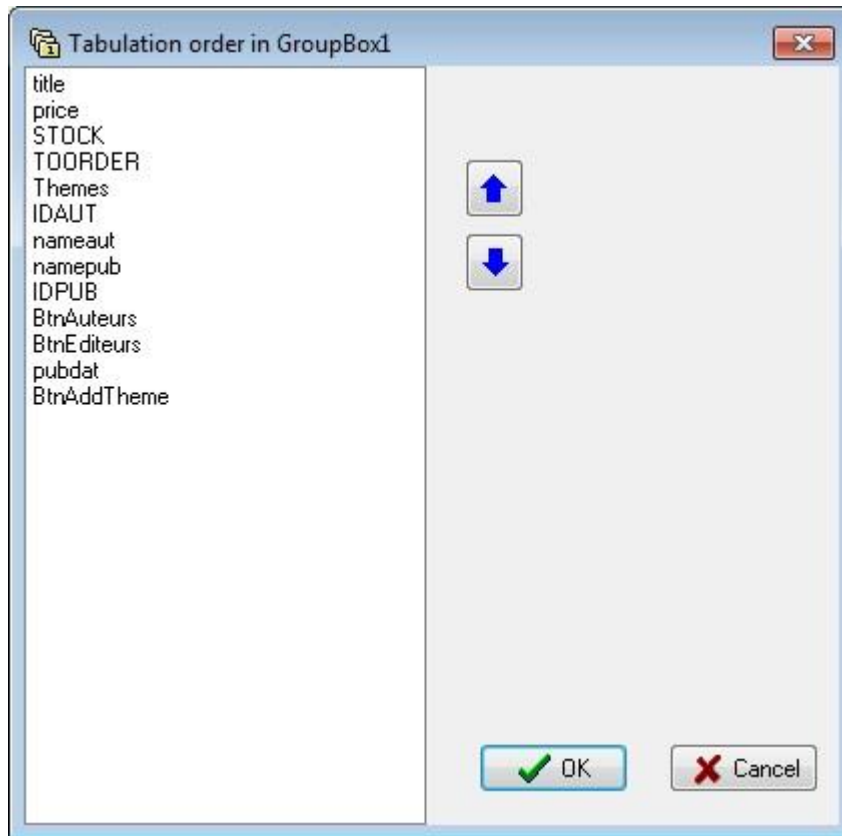


Figure 18

This dialogue box enables modification of the order in which the controls are focussed when the user hits the tab key.

This menu is deactivated if several components are selected or if the selected component cannot contain any other controls.

This editor modifies the TabOrder properties of the components.

It is possible to ensure that the focus does not pass via a component by setting the TabStop property of this component to false.

In this case, the item is grayed in the list.

If a component contains other components, for example, a CPanel, when the focus arrives on this component, the tab order within this component is used, then once the last component in the CPanel is reached, the focus returns to the tab order of the component containing the CPanel.

The first component with the focus is therefore the component with the TabOrder value 0. However, if a form is hidden, then visible again, the focus will not pass again to the component with the TabOrder at 0. Focus will remain on the component that had the focus before the window was hidden.

To restore the focus on a component by programming, use the sdSetFocus function or the ActiveControl property.

To modify the `ActiveControl` property on execution, use the `sdSetCtrl` function.

Caution, you cannot use the `sdSetFocus` function on a component if the window of the component is not visible. Call the `sdSetFocus` function after a `sdShow`. If you use the `sdShowModal` function, call the `sdSetFocus` function in the `OnShow` event of the window or use the `ActiveControl` property.

Background

The "Edit/Background" menu places either a windowed control behind all the other windowed controls of its parent or a non-windowed control behind all the other non-windowed controls of its parent.

Foreground

The "Edit/Foreground" menu places the control in front of all the other controls of the parent control.

Alignment

The "Edit/Alignment" menu gives access to the following dialogue box:

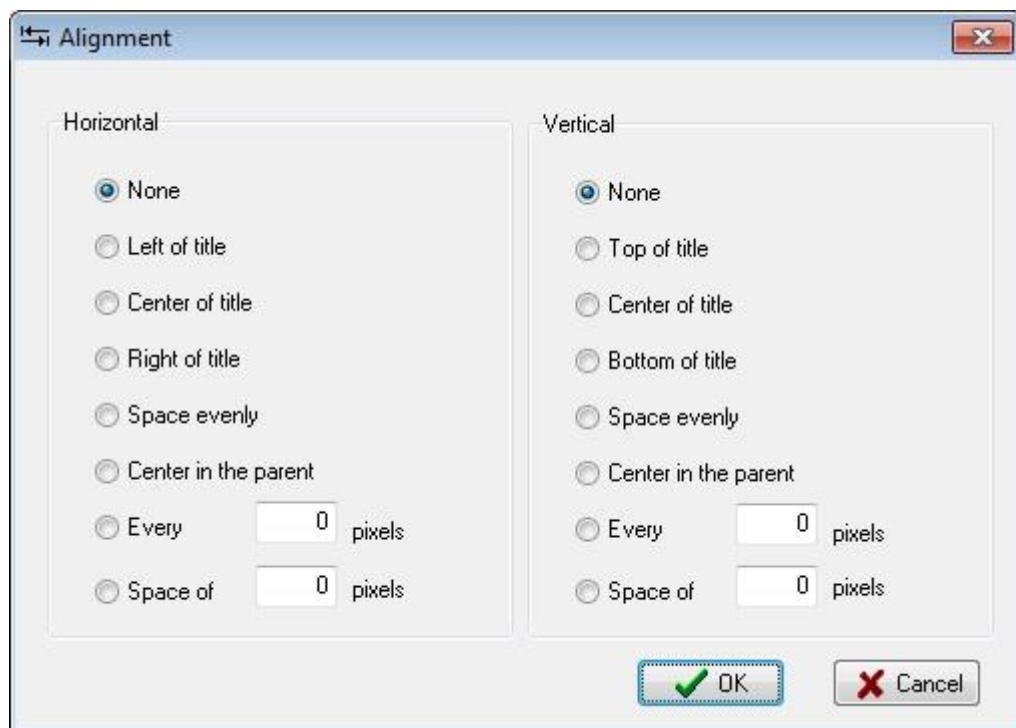


Figure 19

This dialogue box enables automatic repositioning of the selected controls.

It is only proposed if there are several components selected and if they are all contained in the same parent.

Modifications are made in relation to one of the components. If several components are selected, one of them is outlined in black and the others in grey. The component outlined in black is the reference.

Sizes

The "Edit/Sizes" menu gives access to the following dialogue box:

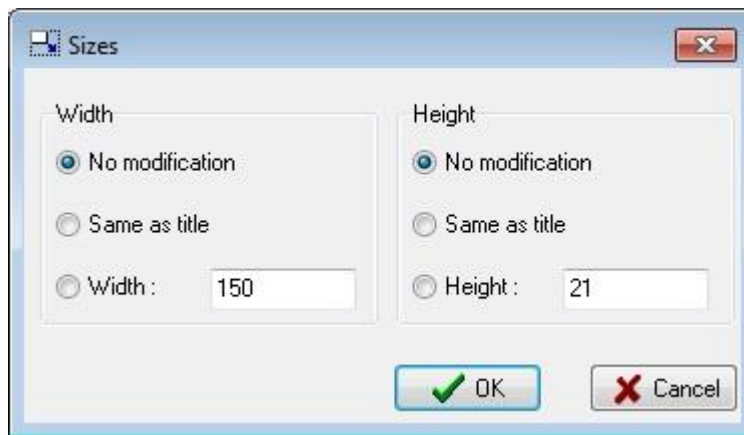


Figure 20

This dialogue box enables the selected controls to be re-sized automatically. It is only proposed if there are several components selected and if they are all contained in the same parent.

Modifications are made in relation to one of the components. If several components are selected, one of them is outlined in black and the others in grey. The component outlined in black is the reference.

Text view

The text view enables the form to be displayed like a text, which may be useful when changing several properties.

If you change the text, the form will be modified, provided there are no syntax errors.

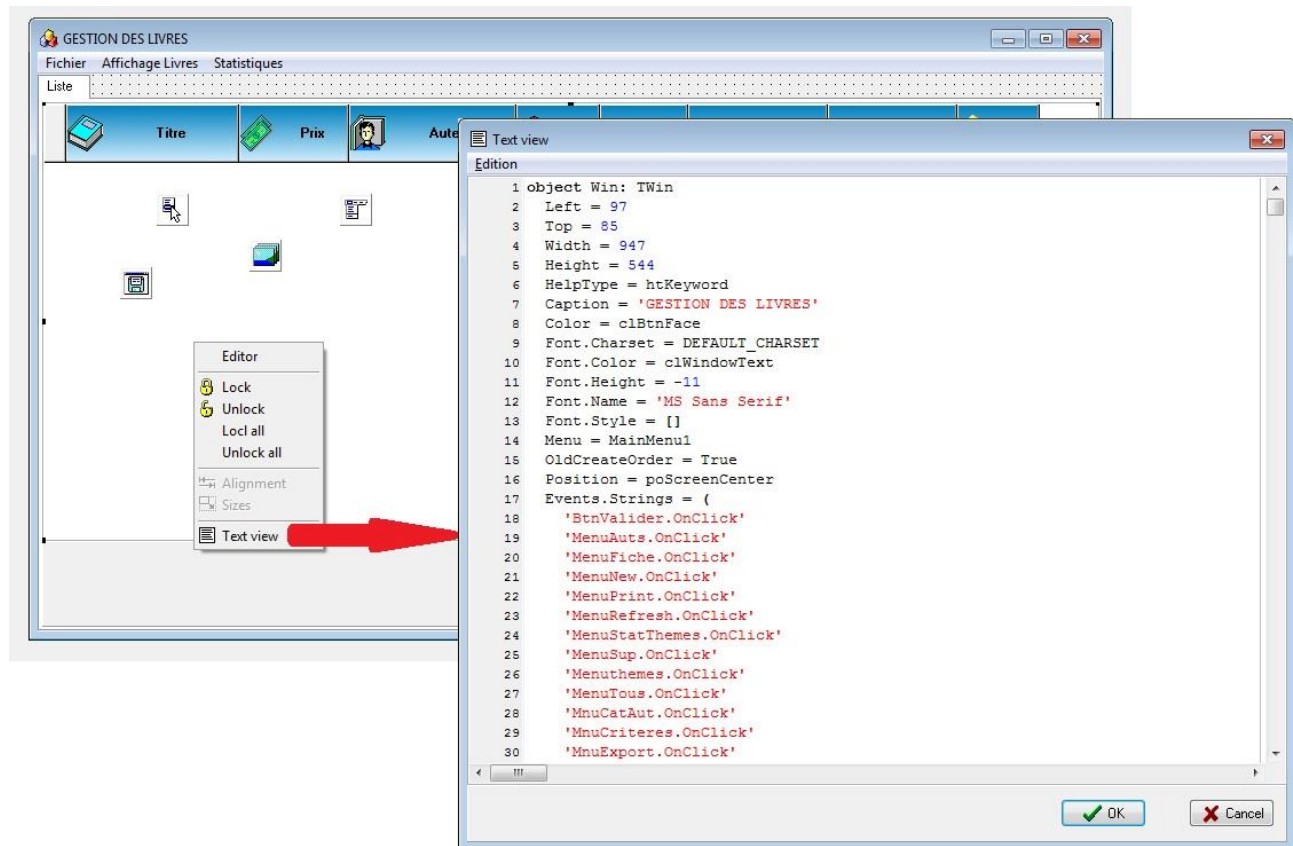


Figure 21

Locking

Right click to access the Lock/Unlock menu.

Locking the components means that they cannot be moved with the mouse although they can still be modified via the property inspector. This prevents accidental displacement of a component after having spent time aligning all the components. Locked components are outlined in red instead of black.

Themes

When you deposit components on a form, you often have to modify certain properties of the component using the property inspector.

You often have to allocate the same values to a certain number of properties. This repetitive task can be avoided using themes.

A theme enables saving of the property values of each component for re-application to other components.

Different themes can be saved.

Activate theme

To activate a theme, use the "Theme/Activate theme" menu.

A dialogue box displays the list of the existing themes on your disk.
Select one of the themes to activate it by double-clicking on its name.
The name of the current theme appears in the Designer status bar.
The current theme will be applied to all the components that you place.

Note 1: Each theme is actually a sub-directory of the directory C:/program files/experia/designer/thèmes

Note 2: You can delete or create a theme from this dialogue box.

*Note 3: You can use the themes without activating a theme.
(see paragraphs below)*

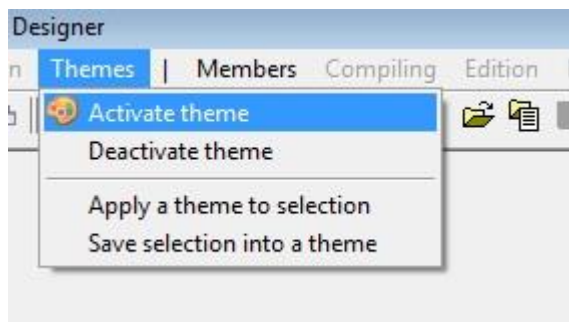


Figure 22

Applying a theme to the selection

Apply a theme other than the current theme using the “Themes/Apply a theme to the selection” menu.

A dialogue box opens and you can choose the theme to be applied to the selected components.

This does not change the current theme.

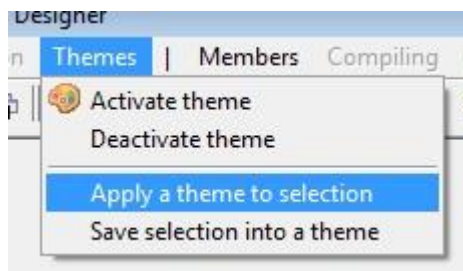


Figure 23

Save selection in a theme

Save the properties of the selected components in a theme.

A dialogue box opens and you can choose the theme into which the properties will be saved. This does not change the current theme.

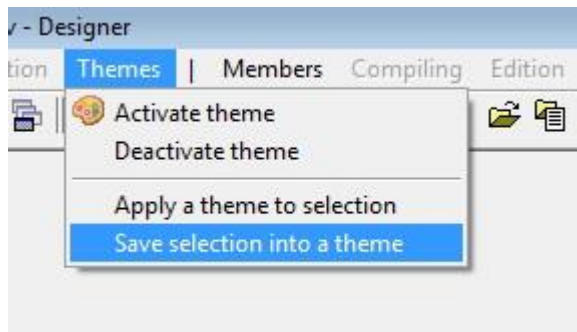


Figure 24

Source locking

When you open a form in the designer, the source is locked. It is therefore impossible for two developers to open the same screen source at the same time.

Configuration menu

Preferences

The "Configuration/Preferences" menu enables display of the following dialogue box:

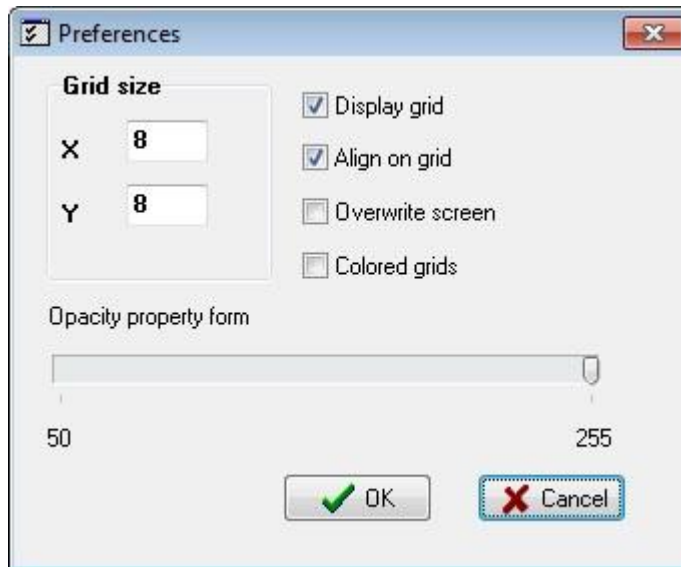


Figure 25

The X and Y fields enable modification of grid on the forms.

The "Display the grid" option determines whether or not the grid is visible on the forms.

The "Align on the grid" option enables modification of Designer behaviour.

If the box is ticked, the components created are aligned on the lines closest to the position where the component is deposited.

The "Anchor properties" field enables you to choose whether or not you want the property inspector always to be above the other windows.

The "Opacity" field makes the property inspector transparent. (This function only works under Windows 2000 or Windows XP).

The "Write over screen" field enables you to choose whether or not the "Write over screen" tick box is ticked by default when you compile a screen.

Help

Access from Designer

Help on the SilverDev components is accessible from Designer via the "Help" menu or by pressing F1. If you are on a form, Help about the selected component will be displayed. If you are in the property inspector, Help about the selected property will be displayed.

Contents

The contents table shows all the Help chapters classed into categories.

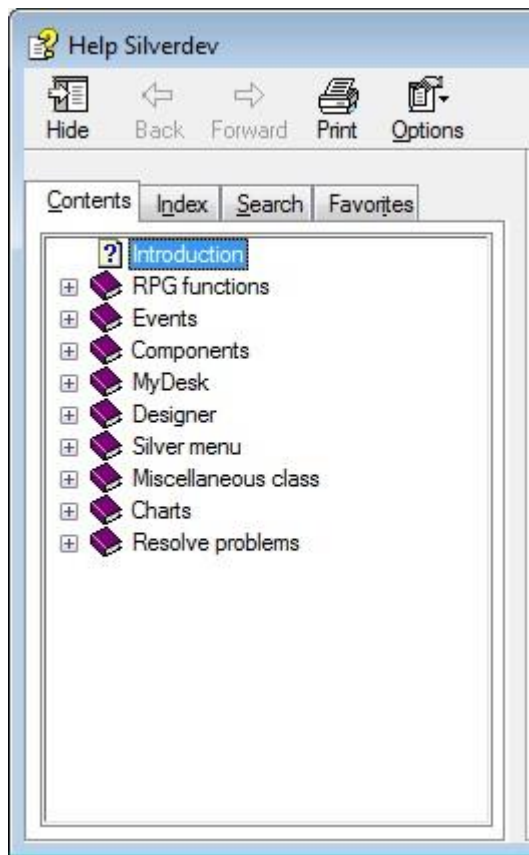
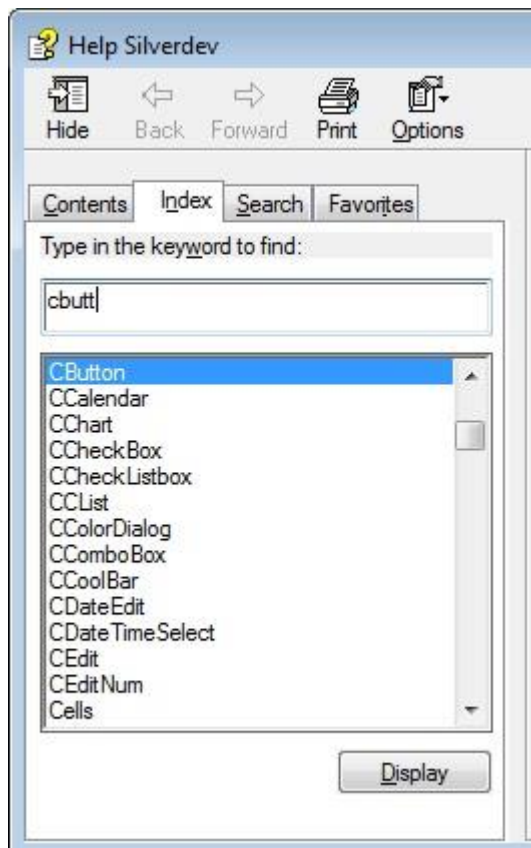


Figure 26

Index

The index enables Help searches by key word.

**Figure 27**

Search

The Search tab searches through all the Help chapters for the word entered.

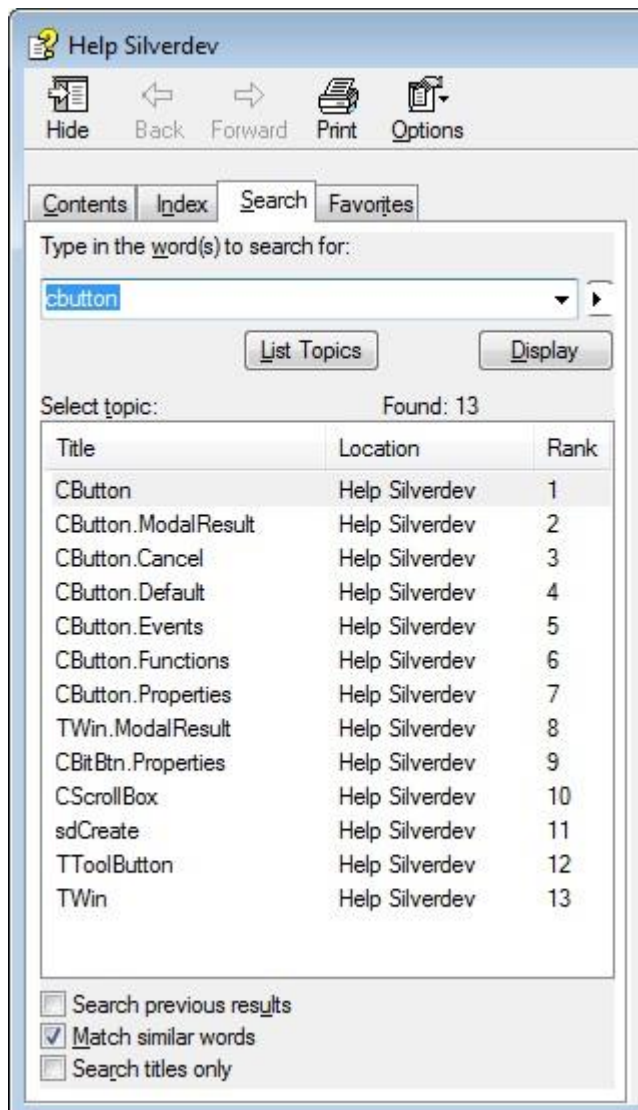


Figure 28

Help topic

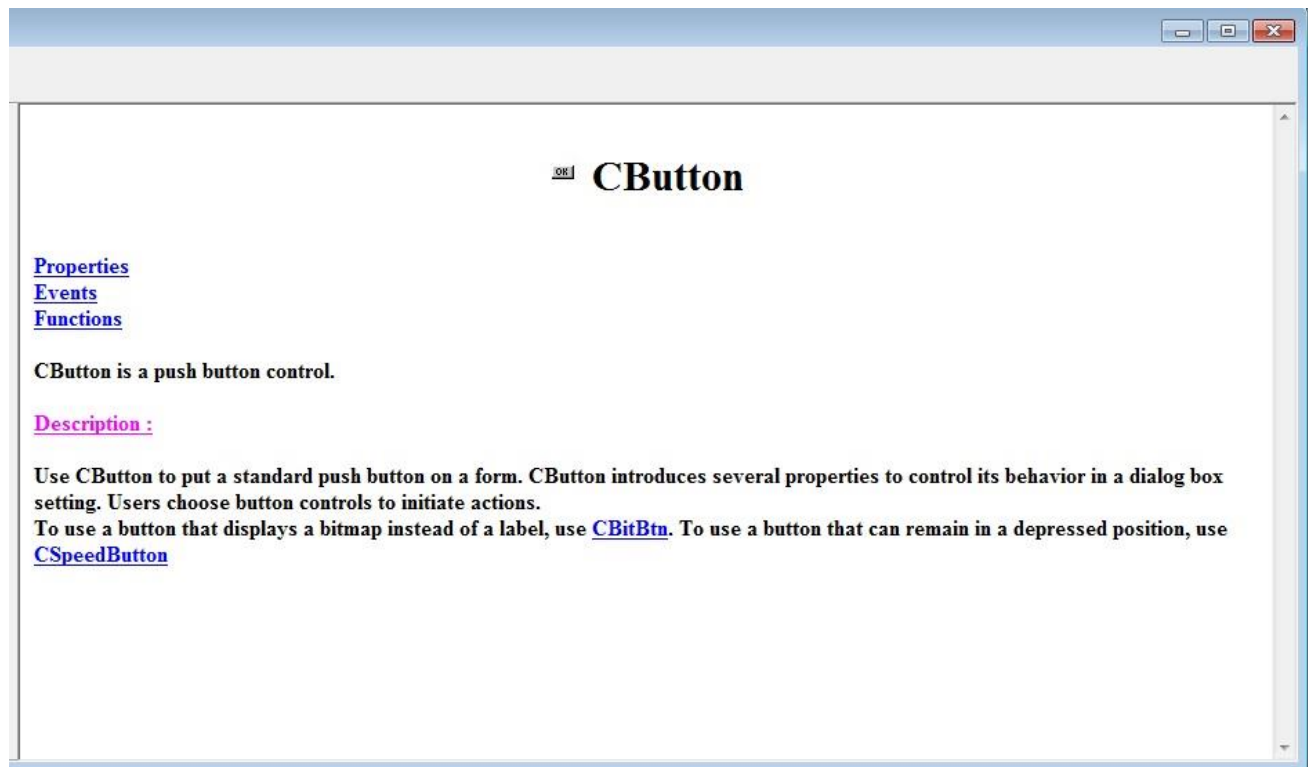


Figure 29

Tools menu

Disconnections

If there is a disconnection on the network while you are working with the designer, you will not be informed automatically.

However, you will be informed when you try to save the source.

The following message will be displayed:



Figure 30

In this case, use the disconnection menu, then log on again.

Caution: the job from the previous connection will still be active, locking your source.

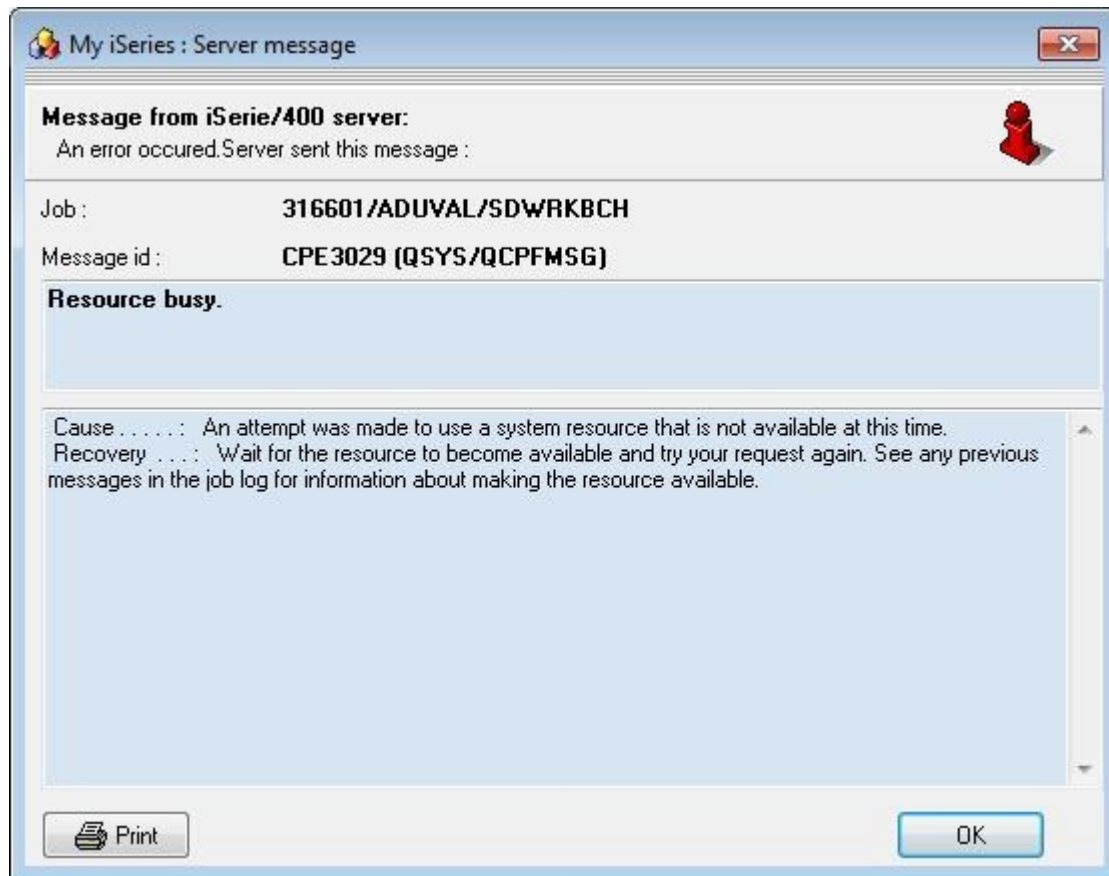


Figure 31

You will have to stop this job to unlock the source.

Chapter 5. Source editor

Introduction

As of version 2.0.0.0, the designer includes a source editor. This editor enables edition of all types of sources, including non-SilverDev sources.

However, the principal advantages of the editor are related to the development of SilverDev applications. In tandem with the designer, it enables access to the associated event manager from the property inspector.

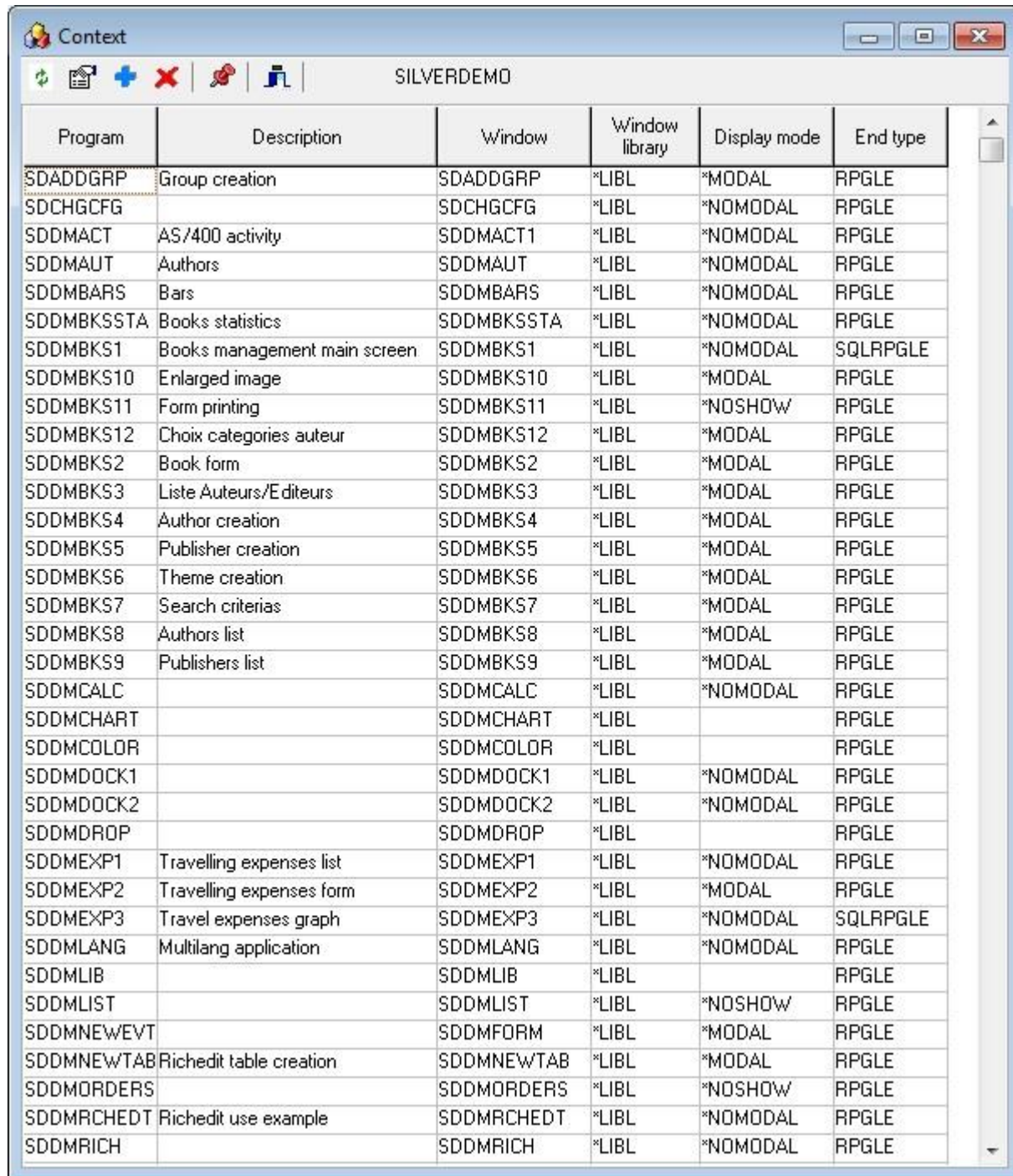
Using a context

Use the "Tools/Context..." menu to display the context list.



Click on OK to select a context.

The context's program list is displayed in a grid.

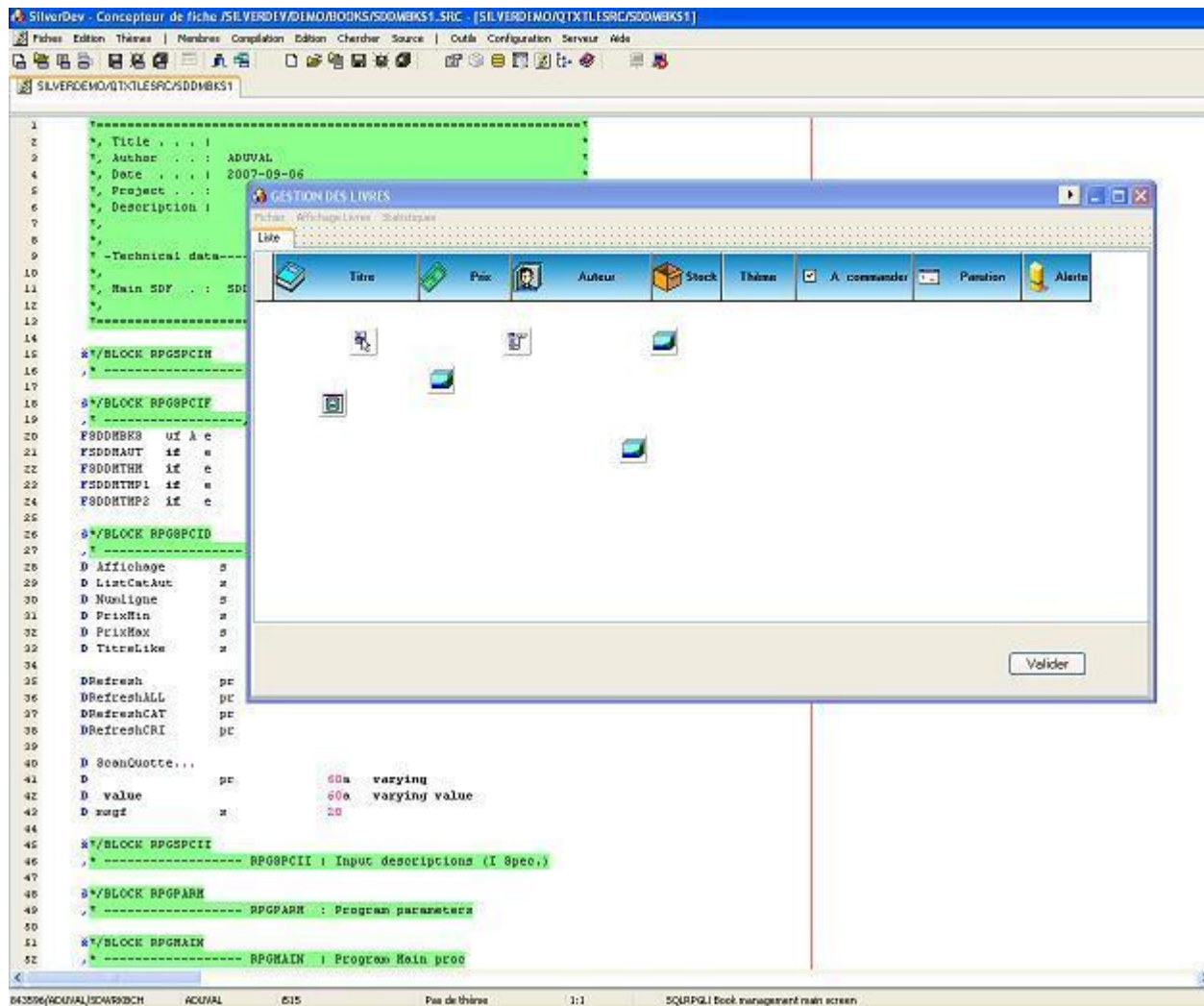


The screenshot shows a window titled "Context" with a toolbar and a table of application metadata. The table has six columns: Program, Description, Window, Window library, Display mode, and End type. The first row, "SDADDGRP", is highlighted with a dashed border.

Program	Description	Window	Window library	Display mode	End type
SDADDGRP	Group creation	SDADDGRP	*LIBL	*MODAL	RPGLE
SDCHGCFG		SDCHGCFG	*LIBL	*NOMODAL	RPGLE
SDDMACT	AS/400 activity	SDDMACT1	*LIBL	*NOMODAL	RPGLE
SDDMAUT	Authors	SDDMAUT	*LIBL	*NOMODAL	RPGLE
SDDMBARS	Bars	SDDMBARS	*LIBL	*NOMODAL	RPGLE
SDDMBKSSTA	Books statistics	SDDMBKSSTA	*LIBL	*NOMODAL	RPGLE
SDDMBKS1	Books management main screen	SDDMBKS1	*LIBL	*NOMODAL	SQLRPGLE
SDDMBKS10	Enlarged image	SDDMBKS10	*LIBL	*MODAL	RPGLE
SDDMBKS11	Form printing	SDDMBKS11	*LIBL	*NOSHOW	RPGLE
SDDMBKS12	Choix categories auteur	SDDMBKS12	*LIBL	*MODAL	RPGLE
SDDMBKS2	Book form	SDDMBKS2	*LIBL	*MODAL	RPGLE
SDDMBKS3	Liste Auteurs/Editeurs	SDDMBKS3	*LIBL	*MODAL	RPGLE
SDDMBKS4	Author creation	SDDMBKS4	*LIBL	*MODAL	RPGLE
SDDMBKS5	Publisher creation	SDDMBKS5	*LIBL	*MODAL	RPGLE
SDDMBKS6	Theme creation	SDDMBKS6	*LIBL	*MODAL	RPGLE
SDDMBKS7	Search criterias	SDDMBKS7	*LIBL	*MODAL	RPGLE
SDDMBKS8	Authors list	SDDMBKS8	*LIBL	*MODAL	RPGLE
SDDMBKS9	Publishers list	SDDMBKS9	*LIBL	*MODAL	RPGLE
SDDMCALC		SDDMCALC	*LIBL	*NOMODAL	RPGLE
SDDMCHART		SDDMCHART	*LIBL		RPGLE
SDDMCOLOR		SDDMCOLOR	*LIBL		RPGLE
SDDMDOCK1		SDDMDOCK1	*LIBL	*NOMODAL	RPGLE
SDDMDOCK2		SDDMDOCK2	*LIBL	*NOMODAL	RPGLE
SDDMDROP		SDDMDROP	*LIBL		RPGLE
SDDMEXP1	Travelling expenses list	SDDMEXP1	*LIBL	*NOMODAL	RPGLE
SDDMEXP2	Travelling expenses form	SDDMEXP2	*LIBL	*MODAL	RPGLE
SDDMEXP3	Travel expenses graph	SDDMEXP3	*LIBL	*NOMODAL	SQLRPGLE
SDDMLANG	Multilang application	SDDMLANG	*LIBL	*NOMODAL	RPGLE
SDDMLIB		SDDMLIB	*LIBL		RPGLE
SDDMLIST		SDDMLIST	*LIBL	*NOSHOW	RPGLE
SDDMNEW EVT		SDDMFORM	*LIBL	*MODAL	RPGLE
SDDMNEW TAB	Richedit table creation	SDDMNEW TAB	*LIBL	*MODAL	RPGLE
SDDMORDERS		SDDMORDERS	*LIBL	*NOSHOW	RPGLE
SDDMRCHEDT	Richedit use example	SDDMRCHEDT	*LIBL	*NOMODAL	RPGLE
SDDMRICH		SDDMRICH	*LIBL	*NOMODAL	RPGLE

Figure 32

Double-click the name of an application to open the source and the associated window.



Caution: if the library window is *LIBL, your library list must be configured correctly.

Members menu

The editor can be used for a non-SilverDev source using the Members menu. Choose the Members/Open menu to open a source.

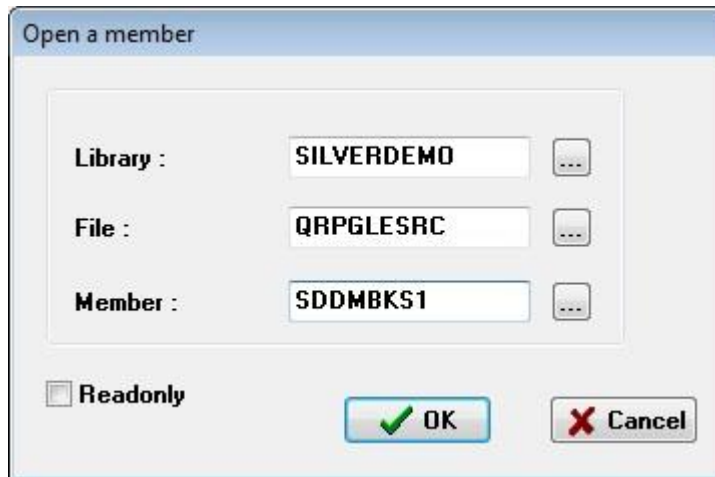


Figure 33

Associated window

If the source was opened from the context, a window is associated with the source.

If the source was opened from the Members menu, it is still possible to associate it with a window using the Members/Link to a screen menu.

When a window and a source are linked, if you double-click the property inspector on an event, the editor focusses on the event manager level. If the manager does not exist, it will be created in the source.

To switch between the source and the window, use F10.

Compilation

Command

If the source was opened via the Members menu, the compilation command will depend on the type of source.

If the source was opened from a context, the compilation command will depend on the context configuration.

In all cases, the program can be compiled using the Compilation menu.

Integrity check

If the source is linked to a window, upon compilation, the editor checks that all the window's events have an event manager and that all the event managers correspond to one of the window's events.

If this is not the case, an information message is displayed but it does not prevent compilation.

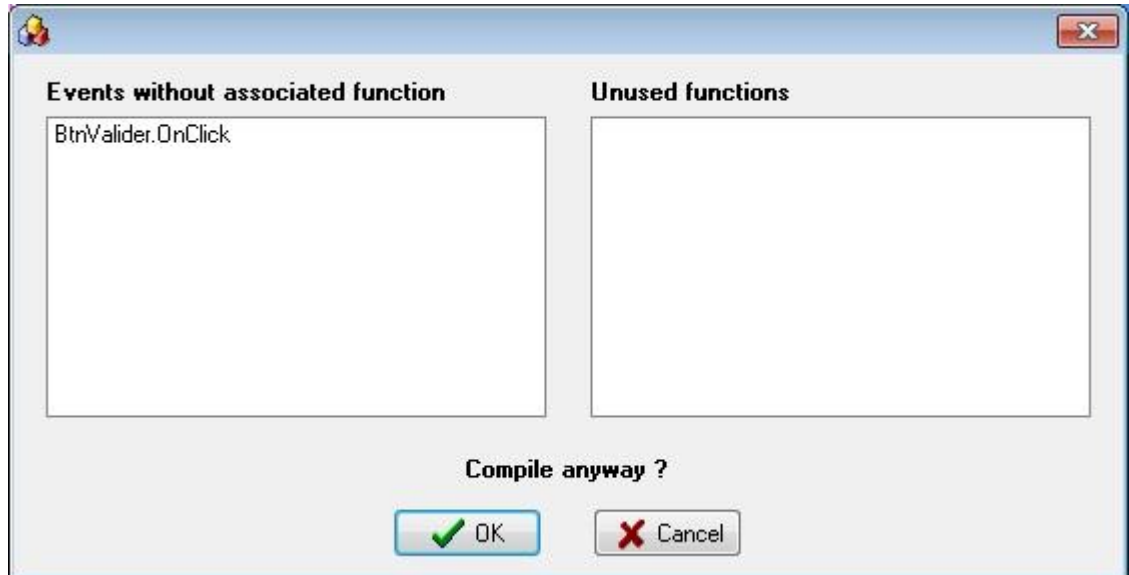


Figure 34

Compilation results, spools and jobs submitted

If the compilation is submitted, the job submitted is displayed. It is then possible to display the job spools via a right click.

If the compilation is not submitted, the editor displays the compilation results.

Id	Line	Column	Level	Message	Source file
RNS9308	0	0	50	Arrêt de la compilation. Erreurs de gravité 30 détectées dans le pro	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7030	1966	36	30	Le nom ou l'indicateur CPT n'est pas défini.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7503	1966	36	30	L'expression contient un opérande non défini.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF5277	2366	26	10	L'Extension d'opération N est ignorée s'il ne s'agit pas d'un fichier en mise à jour.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF5277	2360	26	10	L'Extension d'opération N est ignorée s'il ne s'agit pas d'un fichier en mise à jour.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF5277	2395	26	10	L'Extension d'opération N est ignorée s'il ne s'agit pas d'un fichier en mise à jour.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF5277	2497	26	10	L'Extension d'opération N est ignorée s'il ne s'agit pas d'un fichier en mise à jour.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF5277	2741	26	10	L'Extension d'opération N est ignorée s'il ne s'agit pas d'un fichier en mise à jour.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF5409	1974	1	0	L'appel prototype renvoie une valeur perdue lors de l'utilisation de CALLP.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF5409	2001	1	0	L'appel prototype renvoie une valeur perdue lors de l'utilisation de CALLP.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF5409	2712	1	0	L'appel prototype renvoie une valeur perdue lors de l'utilisation de CALLP.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF5409	2468	1	0	L'appel prototype renvoie une valeur perdue lors de l'utilisation de CALLP.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF5409	2158	1	0	L'appel prototype renvoie une valeur perdue lors de l'utilisation de CALLP.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7031	1534	7	0	Le nom ou l'indicateur SDPDFBE... n'est pas référencé.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7031	119	7	0	Le nom ou l'indicateur SDOPTIMIZE n'est pas référencé.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7031	916	7	0	Le nom ou l'indicateur SDNEXTPAGE n'est pas référencé.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7031	659	7	0	Le nom ou l'indicateur SDNODEAT n'est pas référencé.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7031	1526	7	0	Le nom ou l'indicateur SDPDFDR... n'est pas référencé.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7031	1543	7	0	Le nom ou l'indicateur SDPDFNE... n'est pas référencé.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7031	1517	7	0	Le nom ou l'indicateur SDPDFRE... n'est pas référencé.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7031	1538	7	0	Le nom ou l'indicateur SDPDFEN... n'est pas référencé.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7031	1509	7	0	Le nom ou l'indicateur SDPDFLINE n'est pas référencé.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7031	489	7	0	Le nom ou l'indicateur SDNEWMAP n'est pas référencé.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7031	1409	7	0	Le nom ou l'indicateur SDLAUNCH n'est pas référencé.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7031	734	7	0	Le nom ou l'indicateur SDLOADF... n'est pas référencé.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7031	904	7	0	Le nom ou l'indicateur SDLASTPAGE n'est pas référencé.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7031	622	7	0	Le nom ou l'indicateur SDITEM n'est pas référencé.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7031	36	7	0	Le nom ou l'indicateur SDITEMA... n'est pas référencé.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7031	1406	7	0	Le nom ou l'indicateur SDMSGDEBUG n'est pas référencé.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7031	809	7	0	Le nom ou l'indicateur SDNEWEVENT n'est pas référencé.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7031	665	7	0	Le nom ou l'indicateur SDMOVENODE n'est pas référencé.	QTEMP/QSQLTEMP1(SDDMBKS1)
RNF7031	836	7	0	Le nom ou l'indicateur SDMERGE n'est pas référencé.	QTEMP/QSQLTEMP1(SDDMBKS1)

Figure 35

Errors are shown in bold and information messages are in normal characters.

Double-click one of the lines to open the source at the error.

Use the  and  buttons to display the spools and log messages created during compilation.

Comment:

For a SQLRPGLE type source, the line numbers in the compilation results refer to a temporary file (QTEMP/QSQLTEMP1) created by the compiler. There is no point in modifying this source since it will be created again at the next compilation.

Code completion

Functions

Use the Ctrl+Space shortcut to display the list of SilverDev functions.

The scroll list enables insertion of a function; this scroll list only displays the functions starting with the word being typed.

```

658      C          UPDATE      FBooks
659      C          endif
660      C          Eval          i= sdGetInt(F1:'SFL1':'NextModified')
661      C          enddo
662      C          callp          sdR
663      C          callp          sdR
664      C          endif
665
666      /*EVENT mnuPdf_OnClick
667      *
668      * -----
669      * Description :
670      * -----
671      D Parameters      ds
672      D Win
673      D Evt
674      *
675      D filename        s          25
676      C          eval          fileName = sdSavePd
677      C          if            fileName <> ''
678      C          callp          sdExecutePc('open':filename:''W_NORMAL)
679      C          endif

```

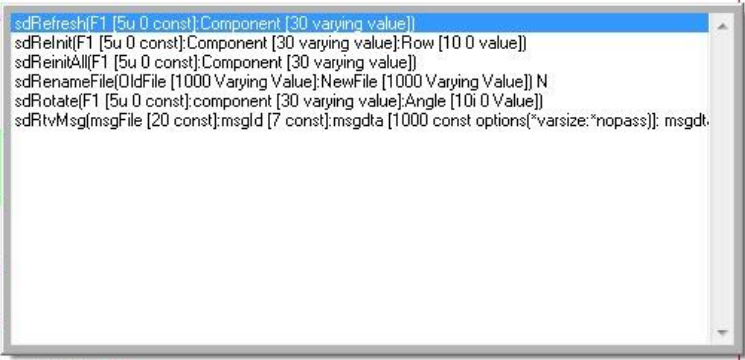


Figure 36

Window tree structure

Use F2 to display a tree structure containing the objects and properties of a window. If a window is linked to a source, this window is displayed in the tree.

```

657      C          'PUBDAT':i)
658      C          UPDATE      FBooks
659      C          endif
660      C          Eval          i= sdGetInt(F1:'SFL1':'NextModified')
661      C          enddo
662      C          callp          sdReinitAll(F1:'SFL1')
663      C          endif
664
665      /*EVENT mnuPdf_OnClick
666      *
667      * -----
668      * Description :
669      * -----
670      D Parameters      ds          based(pevtl
671      D Win
672      D Evt
673      *
674      D filename        s          250      varying
675      C          eval          fileName = sdSavePd
676      C          if            fileName <> ''
677      C          callp          sdExecutePc('open':
678      C          endif
679      /*EVENT MnuExport_OnClick
680      *
681      * -----
682      * Description :
683      * -----
684      D fileName        s          1000      varying
685      C          eval          fileName = sdExport
686      C          if            fileName <> ''
687      C          callp          sdExecutePc('open':
688      C          endif
689

```




Figure 37

Other windows can be added using the "Source/Load a compiled screen" or "Source/Load a design screen" menus.

Press F5 or use the context menu to refresh a node.

Comment:


Do not confuse the window associated with the source with the windows loaded.

Only one window can be associated, but several windows can be loaded. The associated window is loaded automatically.

List of functions and moments

To display the list of functions in a RPGLE source, use the "Search/Function list" menu.

Double-click a line in the list to move the cursor to the function definition.



Line	Name	Type
15	RPGSPCIH	Block
18	RPGSPCIF	Block
26	RPGSPCID	Block
45	RPGSPCII	Block
48	RPGPARM	Block
51	RPGMAIN	Block
54	RPGSR	Block
57	RPGPGMSTART	Block
66	RPGBEFORECREATE	Block
69	RPGAFTERCREATE	Block
72	RPGBEFORESHOW	Block
75	RPGAFTERSHOW	Block
78	RPGPROCDEF	Block
80	SCANQUOTE	Function
110	RefreshALL	Function
168	RefreshCAT	Function
245	RefreshCRI	Function
434	Refresh	Function
450	RPGTABLE	Block
453	MenuAuts_OnClick	Event
459	MnuCatAut_OnClick	Event
472	MenuFiche_OnClick	Event
492	MenuNew_OnClick	Event
507	MenuPrint_OnClick	Event
513	MnuCriteres_OnClick	Event
529	MenuRefresh_OnClick	Event
535	MenuStatThemes_OnClick	Event
541	MenuSup_OnClick	Event
565	Menuthemes_OnClick	Event
571	MenuTous_OnClick	Event
579	sf11_OnEllipsis	Event
599	BtrValider_OnClick	Event
666	mnuPdf_OnClick	Event
679	MnuExport_OnClick	Event

Figure 38

Bookmarks

Bookmarks enable lines of text to be flagged to make them easier to find. Each source can have up to 10 bookmarks. Use the Shift+Ctrl+n shortcut to add a bookmark (n being the bookmark number) and Ctrl+n to go to a bookmark.

To display the bookmarks, use the "Search/View bookmarks" menu.

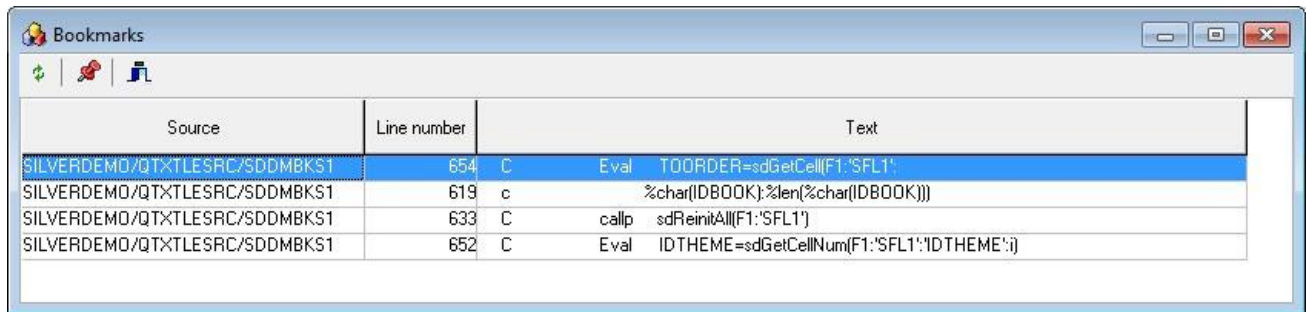


Figure 39

Shortcuts

Ctrl + Space	List of SilverDev functions
F2	Window tree structure
F10	See the associated window
F8	Show/hide the ruler
Ctrl + n	Go to bookmark n
Shift + Ctrl + n	Add bookmark n
Shift + Ctrl+ N	Normal selection
Shift + Ctrl + C	Column selection
Shift + Ctrl + L	Line selection
Shift + Ctrl + I	Block indentation
Shift + Ctrl + U	Block de-indentation
Shift + Ctrl + B	Find the corresponding bracket
Ctrl + z	Undo
Ctrl + shift + z	Redo

Searching an expression in a source file

Use the menu item Tools/Search in members

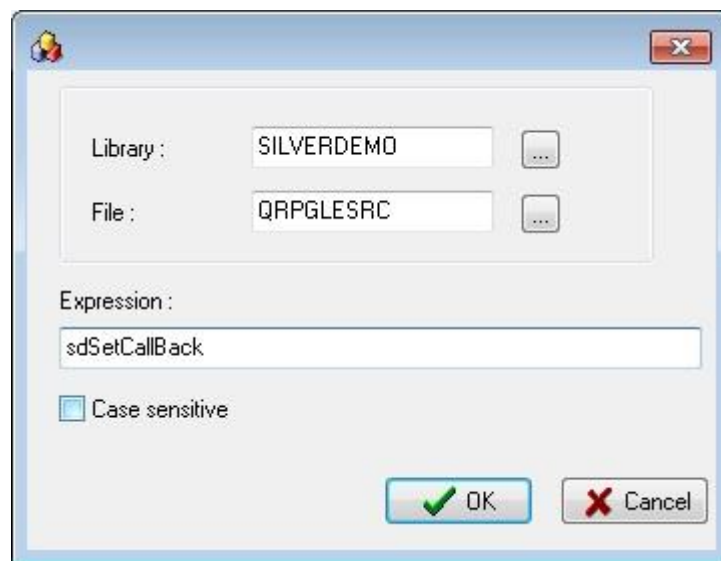
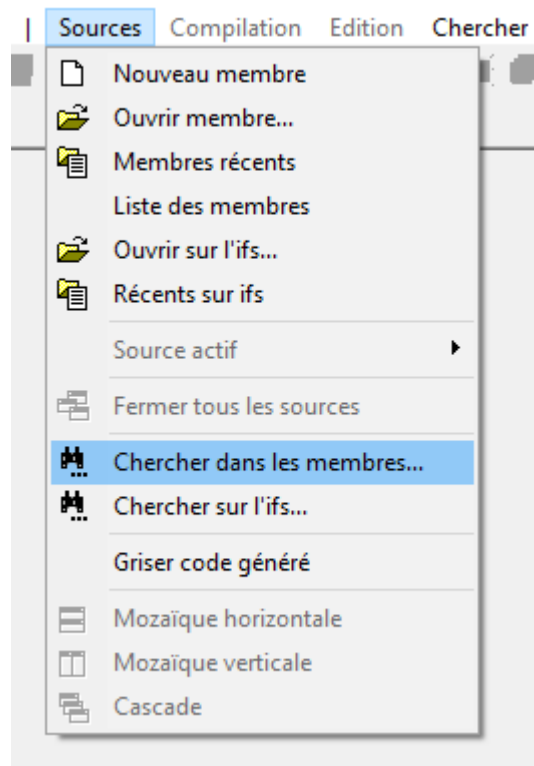
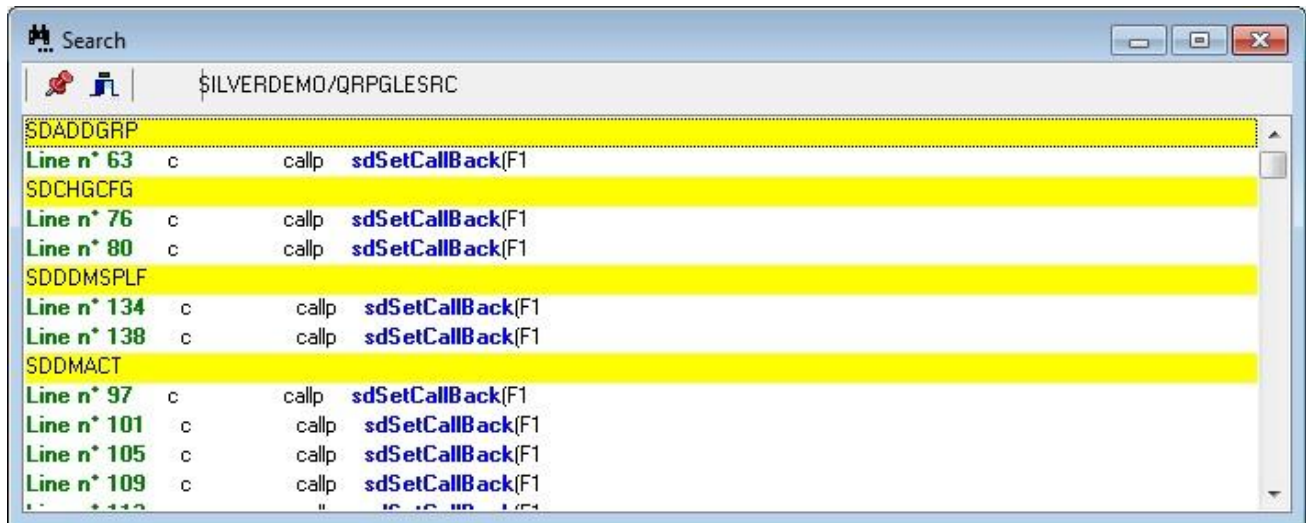


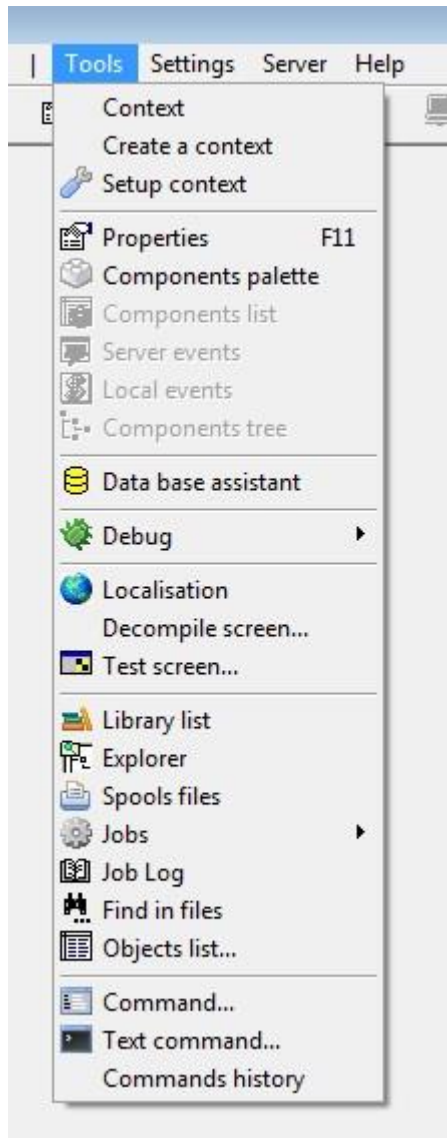
Figure 40

Press Esc to stop the search.

**Figure 41**

Double-click a line to open the member.

As well as the editor function, the designer has a number of tools, including source display, display of jobs according to specified criteria, display of a job log, search in a source, command execution, etc. All these functions are found in the Tools menu.

**Figure 42**

Chapter 6. Generator

RPG IV source

The code entered by the developer and the code generated are stored in different sources (QTXLTESRC and QRPGLSRC respectively).

Thus, when each generation is run, the developer's code is not overwritten.

The window display and associations of events with functions (Callback) are generated. The developer can thus concentrate on the actions to be carried out for each event.

Furthermore, it is possible to request generation of read and write functions. For this purpose, some components have properties that are specific to the generator. These properties are gathered in an object property called OptionsGeneration. See OptionsGeneration

When generating from a source, the generator uses the window description (presented in option 8) and the QTXLTESRC source member.

The member of the RPG file generated in QRPGLSRC will have the same name as the QTXLTESRC member.

***/BLOCK**

The various parts of the QTXLTESRC source are identified by special comments:

***/BLOCK**

The blocks correspond to the various traditional sections of an RPG program.

A */BLOCK comment is followed by a space and the name of the block.

Block name	Description
RPGSPCIH	H specs.
RPGSPCIF	F specs.
RPGSPCID	D specs.
RPGSPCII	I specs.
RPGPARM	Program start, used to define the parameters
RPGPGMSTART	Main body of program
RPGSR	Sub routines
RPGBEFORECREATE	Before window creation

RPGAFTERCREATE	After window and callback creation
RPGBEFORESHOW	Just before the sdShow or sdShowmodal call. Ideal for refreshing screen content. Comment: this moment can be used even if the window's display mode is *NOSHOW.
RPGAFTERSHOW	Just after the sdShow or sdShowModal call.
RPGPROCDEF	Definition of "user" procedures (non events)
RPGTABLE	CTDATA table data

Example:

```
*/BLOCK RPGSpCiD
D Cpt          S          5u 0
```

Event managers: */EVENT

As well as the */BLOCK blocks, there are */EVENT blocks for adding code to the event procedures. These are called event managers.

The generator creates an */EVENT block for each screen event ticked in Designer.

For example:

```
*/EVENT SFL1_OnChangeCellValue
* -----*
* Description :
* -----*
D Parmameters      ds              based(pevtinf)
D Win              5u 0
D Evt              48a
* Modifiable
D Allow            N
* Non modifiable
D ColName          100a  varying
D Row              10i 0
D Value            100a  varying
*
```

Comment:

If the event has parameters, they are added by the generator.

Comment:

*The generator produces the names of */EVENT block names from the main form.*

The total length of an event manager is limited to 41 characters.

*As for */BLOCK, */EVENT blocks are not case-sensitive.*

For example:

```
*/BLOCK RPGSpcid
D Cpt              S              5u 0
*/EVENT Timer1_OnTimer
* -----*
* Description :
* -----*
D Parmameters      ds              based(pevtinf)
D Win              5u 0
```

```
D Evt                                48a
C          Eval          Cpt = Cpt + 1
C          Callp          SdSetString(F1:'Label1': 'Caption':CUSTNAME)
*/EVENT ButtonRefresh_OnClick
* -----*
* Description :
* -----*
D Parmameters      ds              based(pevtinf)
D Win              5u 0
D Evt              48a
C          Callp          Timer1_OnTimer(Pform)
```

OptionsGeneration

OptionsGeneration is a property that is specific to the generator.

This property is an object that has properties itself.

They differ according to the component.

GenerateGet	Determines whether or not the generator adds the code corresponding to this component to the read function.
GenerateSet	Determines whether or not the generator adds the code corresponding to this component to the write function.
DefaultGenerateGet	Determines whether or not the generator adds the code corresponding to the subjacent components to the read function (or to the columns in the case of a CSFL).
DefaultGenerateSet	Determines whether or not the generator adds the code corresponding to the subjacent components to the write function (or to the columns in the case of a CSFL).

The values are set to true by default for all components, but to false for the window. The window's optionsGeneration properties must therefore be modified so that the generator can create read and write functions.

The generator creates the read and write instructions in specific procedures for all the components concerned (see the subjacent components). The name starts with WRITE or READ, followed by an underscore and the name of the parent component.

For example:

```
d Write_SCREEN1    pr
d Read_SCREEN1     pr
```

```
p Write_SCREEN1    B
d                  PI
c                  Callp    sdSetString(F1
c                  : 'EDIT1 '
c                  : 'TEXT '
c                  :EDIT1)
c                  Callp    sdSetNum(F1
c                  : 'EDITNUM1 '
c                  : 'VALUE '
c                  :EDITNUM1)
p                  E
```

```
p Read_SCREEN1     B
```

```

d          PI
c          Eval      EDIT1=
c          sdGet(F1
c          : 'EDIT1 '
c          : 'TEXT')
c          Eval      EDITNUM1=
c          sdGetNum(F1
c          : 'EDITNUM1 '
c          : 'VALUE')
p          E

```

This implies that fields Edit1 and Editnum1 exist.

If you have created the components by dragging them from the database assistant, these fields exist once you have declared the files in your program.

For CSFL components, the procedures receive the number of lines to be processed as a parameter.

QRPGLESRC protection

The generator uses the qtxtlesrc source to generate a source in qrpglesrc. The generator does not overwrite the qrpglesrc source if it was not generated by silverdev.

```

00  20/04/10  18:13:05,400000  SDGENRPG      SILVERDEV  *STMT      GSVDESCRT
SILVERDEV  *STMT
  From module . . . . . :    SDGENRPG
  From procedure . . . . . :    SDGENRPG
  Statement . . . . . :    7800
  To module . . . . . :    GSVDESCRT
  To procedure . . . . . :    GSVDESCRT
  Statement . . . . . :    613
  Message . . . . . :    The source member already exists but has not been
                        generated by SILVERDEV. REPLACE(*YES) ignored, generation cancelled.
  Source member already exists, but has not been recongnized as generated by
                        SilverDev. REPLACE(*YES) ignored. &NTo protect your sources, generation
                        process has been aborted. &NGive another member name and retry
40  20/04/10  18:13:05,400344  GSVDESCRT      SILVERDEV  *STMT      GSVDESCRT
SILVERDEV  *STMT
  From module . . . . . :    GSVDESCRT
  From procedure . . . . . :    GSVDESCRT
  Statement . . . . . :    615
  To module . . . . . :    GSVDESCRT
  To procedure . . . . . :    _QRNP_PEP_GSVDESCRT
  Statement . . . . . :    *N
  Message . . . . . :    Source generation Failed *****.
  Cause . . . . . :    This message is used by application programs as a general
                        escape message.

```

PSVDCMP

The PSVDCMP file enables definition of the functions to be generated by the generator. This file should not require modification.

Example:

```

Component Type:          CEDITNUM
Component Default property: VALUE
Component Set Function:  sdSetNum
Component Get Function:  sdGetNum
Component Level:         01          Component Default gen: Y

```

This record indicates that for CEditNum type components, SilverDev will generate the callp sdSetNum (F1 :'editnum1' :'value' :editnum1) code in the write function and the eval editnum1=sdGetnum(F1 :'editnum1' :'value') code in the read function.

Generated source

The programs are written in RPG ILE.

To compile a program, you need the SilverDev library in the library list.

Basic program elements

A SilverDev program always starts with:

```
H BNDDIR('SILVERDEV') DFTACTGRP(*no)
```

This links the program to the SILVERDEV/SDSRVPGM program service where all the useful functions are defined.

In a D specification, the following line must be added:

```
/COPY H,SILVERDEV
```

The SILVERDEV/H/SILVERDEV member contains the prototypes of the functions defined in SILVERDEV/SDSRVPGM.

The program comprises the following lines:

```
C          callp      sdStart(%paddr('INIT'))
C          eval        *inlr = *on
```

The rest of the program is composed of sub-procedures, notably the Init procedure which is the procedure called by sdStart.

In the Init procedure, start with a call to the sdCreateForm function which sends an initial form. The program will stop when the user closes the first form (or if you close it by programming).

After creating a form, you must associate the events to be received at a function that you create, called the event manager. This association is achieved using the sdSetCallBack function.

The Init function will thus look like this:

```
P Init          B
D              PI
C              eval      F1 =sdCreateForm('*LIBL/SDDMBKS1')
C              callp      sdSetCallBack(F1 :
C                          'BtnValider.OnClick':%paddr('VALIDATION'))
P              E
```

Note 1: Field F1 is declared as follows:

```
D F1          S          5u 0
```

Note 2: If you want to complete processing immediately after sending the first window (field initialisation, for example), you can do so in the *Init* function after the *sdSetCallback* calls.

In the above example, the *sdSetCallBack* function associates the event manager *VALIDATION* with the *BtnValider.OnClick* event.

Note 3: The name of the *VALIDATION* function must be written in capital letters.

For the list of events to be notified, use the "see/Recap" menu in Designer.

The *VALIDATION* function is a traditional RPG ILE sub-procedure that is run each time the user clicks on *BtnValider*.

All the functions required to handle the components on the form are provided in the *SDSRVPGM* program service. See the list in the *SilverDev.hlp* help file.

The program ends for the client when the main window is closed (either by the user or by code). The main window is the first window sent to the client.

sdStart

When the *sdStart* function is called for the first time, this function calls the function whose address is sent as a parameter, then enters a loop that ends when the client part indicates that the main window has been closed.

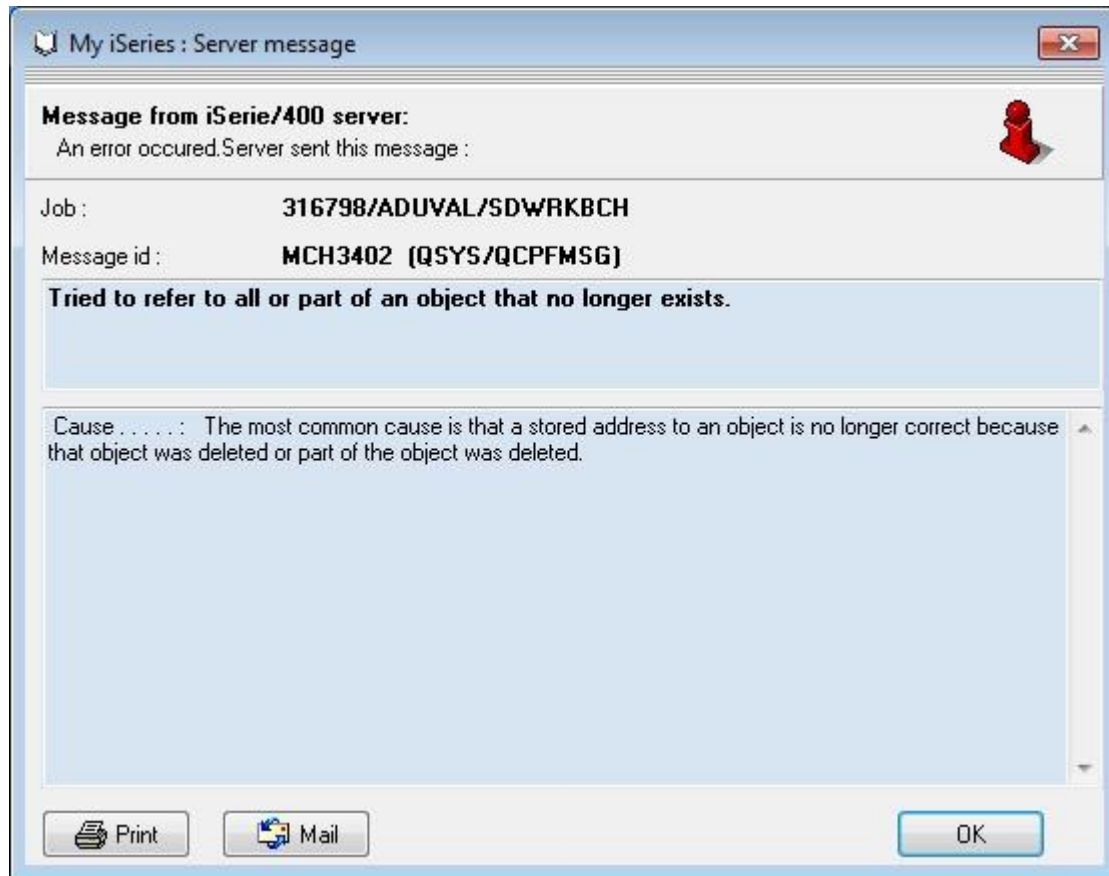
If the *sdStart* function is then called again (in a second program for example), the function merely calls the function whose address is sent as a parameter, then ends. This means that when a second program is called, it is loaded into the memory, then the program is ended.

Therefore, if the program is in the **NEW* activation group, the activation group is destroyed. The program is then unloaded from the memory.

Comment:

*Never compile your programs in the *NEW activation group.*

If your program is called by another, the following error will occur:

**Figure 43**

For example, recompile the sddmthm program with the actgrp(*NEW) instruction.

Chapter 7. RT mode or LR mode

Choosing LR mode in a silverdev program is delicate.

If you chose the LR mode, it is recommended to release the window in the onclose event to free the memory used by the client side window. Also call the sdFreeForm function to inform the server that the window does not exist anymore on the client side (Be careful of the difference between sdFree and sdFreeForm)

```

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)

```

Summary:

	RT	LR
Non modal	A window is created when the program is called the first time. The same window is used for every call of the program.	<p>The most complicated case.</p> <p>A window is created for each call of the program. You can have several occurrences of the window at the same time. So, don't use the F1 variable. Use the win parameter within the events.</p> <p>Free the window in the onClose event by assigning the value 2 to the action parameter. (In order to avoid leak memories on the client side)</p> <p>Call the sdFreeForm function to inform the server that the window does not exist anymore on the client side (To avoid leak memories on the serveur side and to avoid mistakes with sdGetForm... functions)</p>

		If you need to store some information for each occurrence of a window, you have to put them in an array. (See Erreur ! Source du renvoi introuvable.)
Modal	<p>A window is created when the program is called the first time. The same window is used for every call of the program.</p>	<p>A window is created for each call of the program.</p> <p>Since the window is modal, you can't call the program a second time while the window is displayed. You then can have only one occurrence of the window. You can use the F1 variable.</p> <p>Free the window in the onClose event by assigning the value 2 to the action parameter. (In order to avoid leak memories on the client side) Call the sdFreeForm function to inform the server that the window does not exist anymore on the client side (To avoid leak memories on the serveur side and to avoid mistakes with sdGetForm... functions)</p> <p>If you need to keep some information for each occurrence of a window, you have to put them in an array. (See Erreur ! Source du renvoi introuvable.)</p>

Handling the components

The components can be modified by the functions provided in the sdsrvpgm program service.

sdSetString, sdGet

String type properties can be modified by the sdSetString function.

```
d sdSetString      pr
d  form              5u 0
d  component        30   varying value
d  propertyPath    256a   varying value
d  value           9999a   varying value
d  Trim            N     value options(nopass)
```

Example of the text property of a CEdit component:

```
C          callp      SdSetString(F1:'TITLE':
C          'Text':'Mystic river')
```

Note: if the function is to be applied to a window property, the component parameter value must be '*FORM'

The property can be queried using the sdGet function:

```
d sdGet            pr          9999a   varying
d  form              like(tWHandle)
d  component        30   varying value
d  propertyPath    256a   varying value
```

```
C          Eval      titre=sdGet(form:'Title  ':'Text')
```

SdSetNum, sdGetNum

```
d sdSetNum      pr
```

d	Pform		5u 0
d	Component		30 varying value
d	Property		256 varying value
d	Valeur		30s 5 value

d	sdGetNum	pr	30s 5
d	Pform		5u 0
d	Component		30 varying value
d	Property		256 varying value

For numeric properties, use the sdSetNum and sdGetNum functions:

C	callp	SdSetNum(F1: 'IDPUB' : 'Value' : 0)
---	-------	-------------------------------------

C	eval	PRIX=sdGetNum(F1: 'PRICE' : 'Value')
---	------	--------------------------------------

sdSetBool, sdgetBool

For Boolean properties, use the sdSetBool and sdGetBool functions.

d	sdSetBool	pr	
d	form		like(tWHandle)
d	component		30 varying value
d	propertyPath		256a varying value
d	value		n value

d	sdGetBool	pr	N
d	form		like(tWHandle)
d	component		30 varying value
d	propertyPath		256a varying value

C	callp	sdSetBool(F1: 'Button1' : 'visible' : *off)
---	-------	---

C	if	sdGetBool(F1: 'Date' : 'ValidDate')=*off
---	----	--

The SDSRVPGM service program provides more than 300 functions to handle components.

Chapter 8. Debug

Server side debug with designer

Designer enables debugging of a program as another job.
All types of program, including non-SilverDev programs, can be debugged.

Comment: only one job can be debugged at a time.

Launching a debug

Use the Tools/Debug/SilverDev jobs menu.

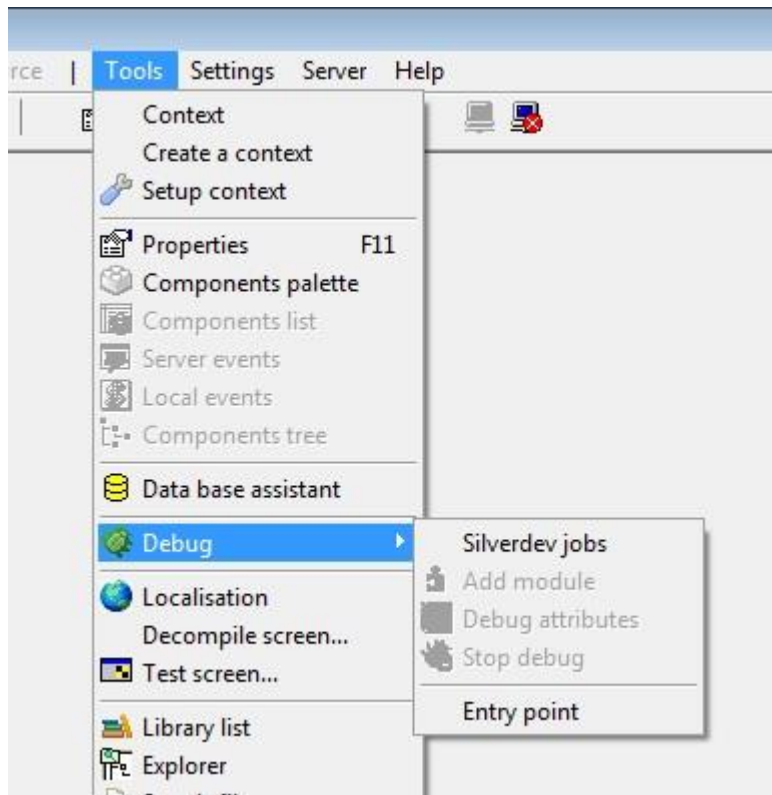


Figure 44

The screen below shows the SilverDev server jobs.

Rank	Job name	User	Job number	State	Active state	Function	Function type	Outq	Silverdev command	User	IP address
10	SDWRKBCH	ADUVAL	316897	*ACTIVE	TIMW	SDWRKBCH	P	*DEV			
9	SDWRKBCH	ADUVAL	316896	*ACTIVE	TIMW	SDWRKBCH	P	*DEV			
8	SDWRKBCH	ADUVAL	316888	*ACTIVE	SELW	SDWRKBCH	P	*DEV	/Examples/Books/sddmbks	ADUVAL	192.168.0.110
7	SDWRKBCH	ADUVAL	316876	*ACTIVE	SELW	SDWRKBCH	P	*DEV	*MYDESK	ADUVAL	192.168.0.110
6	SDWRKBCH	ADUVAL	316807	*ACTIVE	RUN	SDWRKBCH	P	*DEV	*DESIGNER	ADUVAL	192.168.0.110
5	SDWRKBCH	ADUVAL	316778	*ACTIVE	TIMW	SDWRKBCH	P	*DEV			
4	SDWRKBCH	ADUVAL	316776	*ACTIVE	SELW	SDWRKBCH	P	*DEV	*DESIGNER	ADUVAL	192.168.0.103
3	SDWRKBCH	ADUVAL	309360	*ACTIVE	TIMW	SDWRKBCH	P	*DEV			
2	SDWRKBCH	ADUVAL	309357	*ACTIVE	TIMW	SDWRKBCH	P	*DEV			
1	SDWRKBCH	ADUVAL	307935	*ACTIVE	TIMW	SDWRKBCH	P	*DEV			

Figure 45

Comment:

To modify the list of jobs displayed, use the button.


Right click the job to be debugged and click "Debugger"

	Silverdev command	User	IP address
	/Examples/Books/sddmbks	ADUVAL	192.168.0.110
	*MYDESK	ADUVAL	
	*DESIGNER	ADUVAL	
	*DESIGNER	ADUVAL	

Figure 46

Choosing the program

If the program to be debugged is in the call pile, double-click a line corresponding to the program.

If the program to be debugged is not in the call pile, use the  button.

In both cases, the following window will be displayed:



Figure 47

Comment: The api supplying the call pile does not specify the object type (*PGM or *SRVPGM), so even if you choose in the call pile, you will have to specify the type in this window.

Choosing the module

If the program (or program service) comprises several modules, the list of modules is displayed:

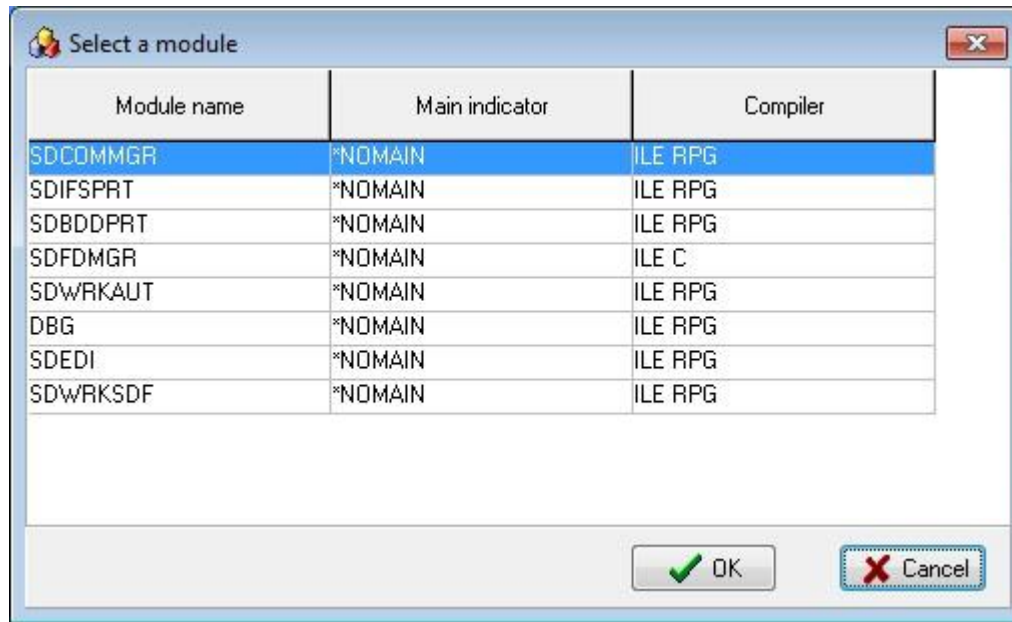


Figure 48

In our example (silverdemo/sddmbks1), there is only one module, so the module selection window is not shown.

Choosing the view

Once the module to be debugged has been selected, another window enables choice of view type:

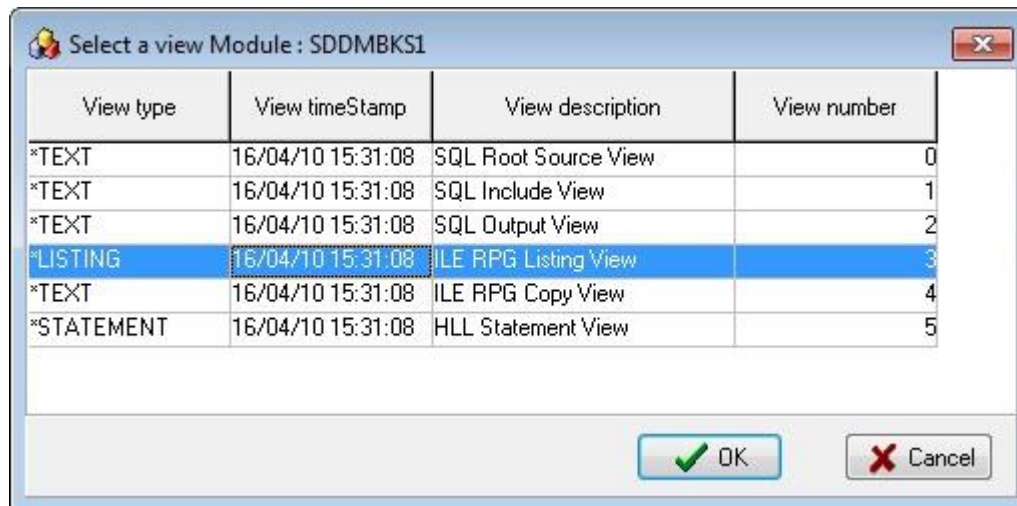


Figure 49

The number of views available depends on how the program was compiled (dbgview option).

For example, the *Listing view is available if the module was compiled with the DBGVIEW *LIST or DBGVIEW *ALL option.

Functions in a view

Here, we have selected the *LISTING view.

The blue dots in the margin indicate that the line is executable.

Use F5 to find the line where the program stopped.

Click in the margin to add or delete a breakpoint.

For more options, use the right click.

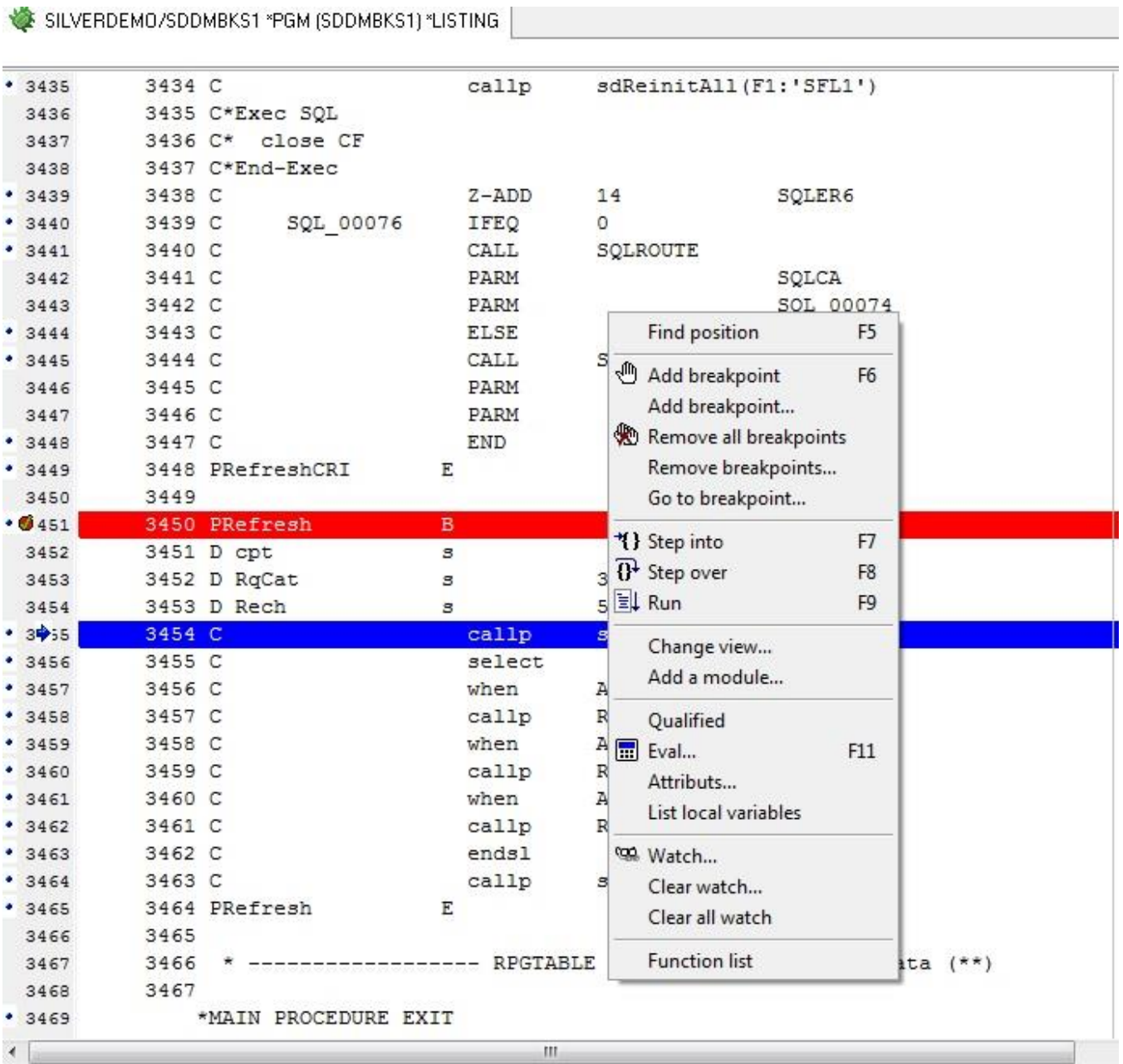


Figure 50

Description of the popup menu options:

Search position	Places the cursor in the program's location. The line is displayed in blue.
Add breakpoint	Adds a breakpoint directly at the line on which the cursor is positioned.
Add a breakpoint...	Displays a prompt to add a conditional breakpoint.
Remove all	Removes all the breakpoints.

breakpoints	
Remove breakpoints...	Displays a table with all the breakpoints. Select one or more lines in the table and click OK
Go to breakpoint...	Displays a table with all the breakpoints. Select a line and click OK.
Run	Enables program execution to be continued when at a breakpoint.
Step into	Step by step, entering into the functions.
Step over	Step by step without entering into the functions.
Submit debug command...	Displays a prompt that enables any debug instruction to be run. The result of the instruction is displayed in a tool window. See the Debug Instructions chapter.
Change views...	Displays the view selection table, as at the start of the debug operation. Only one view can be displayed at a time for a module. When changing views, the list of breakpoints is preserved.
Change modules...	Displays the list of the program (or program service) modules in the current view. Several modules of a single program (or service program) can be displayed at the same time. To display a module of another program, use the main menu. Debug/Program selection.
Qualified	Enables execution of the QUAL command for the line on which the cursor is positioned. The QUAL instruction enables the scope of validity for the eval instruction to be determined.
EVAL	Enables display of the contents of the value of the expression on which the cursor is positioned.
Attributes	Displays the variable type.
List local variables	Displays all the local variables.
Watch	Use Watch to add a variable to be monitored. The program stops when a monitored variable is modified.
Clear Watch...	Displays a table of all the monitored variables and enables deletion of these variables.
Remove all Watch...	Removes all the monitored variables
List of functions	Displays a window containing the list of all the functions of the module displayed. Double-click in the window to move the cursor to the start of the function.

Remarque : lorsque vous sélectionnez une option de ce menu, la commande équivalente est affichée dans la zone de commandes de debug.

Debug command field

At the bottom of the text, a field allows to execute a debug command.

To see debug commands, see chapter Debug instructions .

When you select an option in the popup menu or double click the command is displayed in this field.

You can go from text to debug command field with the tab key.

Click and Double clicks

To go faster, you can click in the gutter to add/remove a breakpoint.

When you double click on the text, an eval of the selected text is executed.

If you click on a qualified field such as Ds1.Fields, click on Ds1 to eval the whole field, and click on Field1 to eval Ds1.Field1.

Debug log

The debug log window displays the results of the debug commands.

EVAL command results in particular:

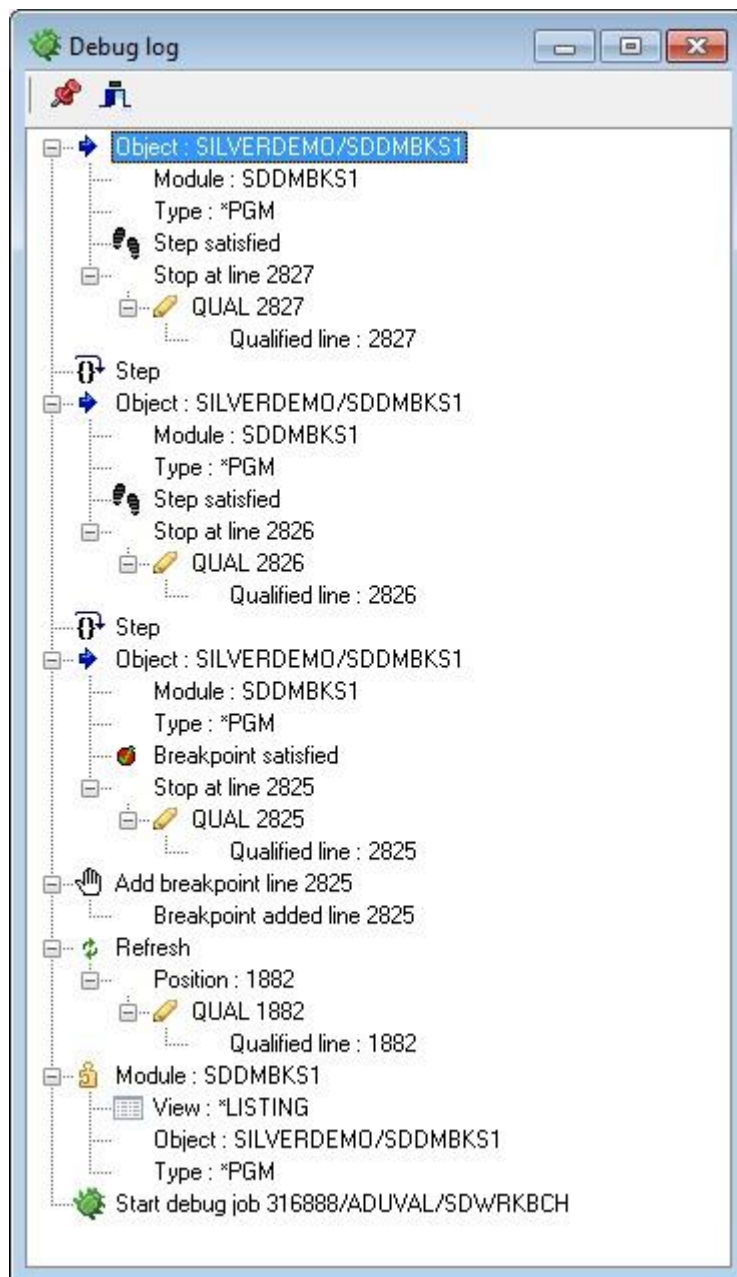


Figure 51

This window is opened automatically when a debug operation is launched.

List of functions

A window displays the list of functions.

Double-click a function to move the cursor to the start of this function in the view.



Figure 52

This window is opened automatically when a debug operation is launched.

Adding a module

Use the Tools/Debug/Add module menu to add a module:

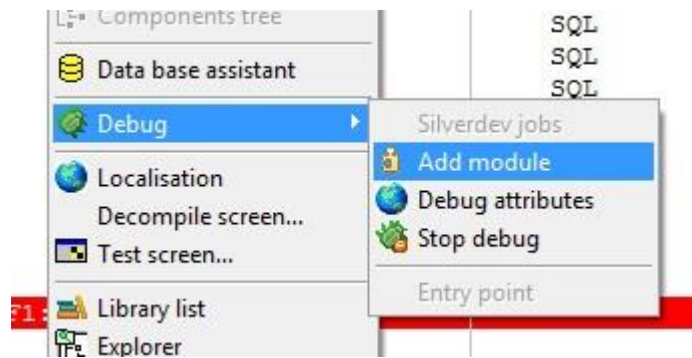


Figure 53

Ending the debug

To stop the debug operation, use the main menu "Debug/End debug"
All the debug windows are closed automatically except the log window.

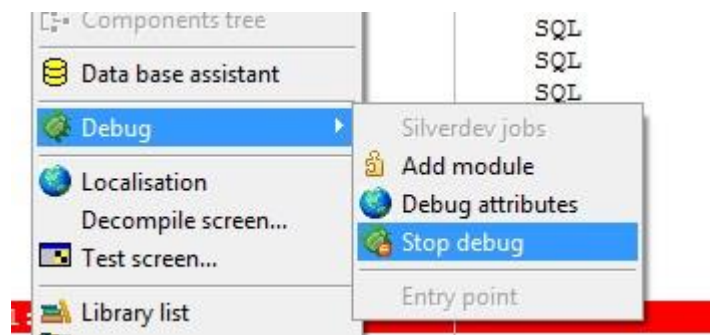
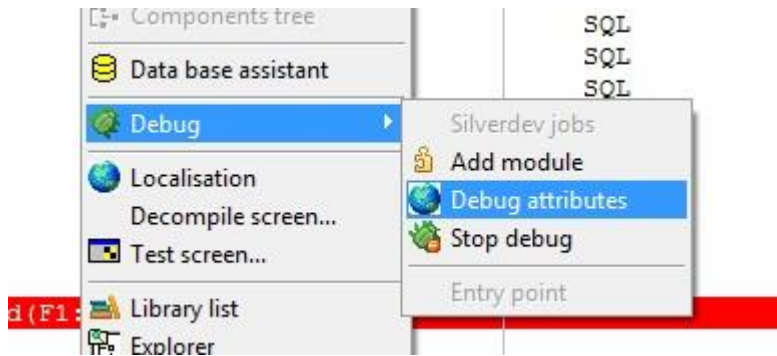


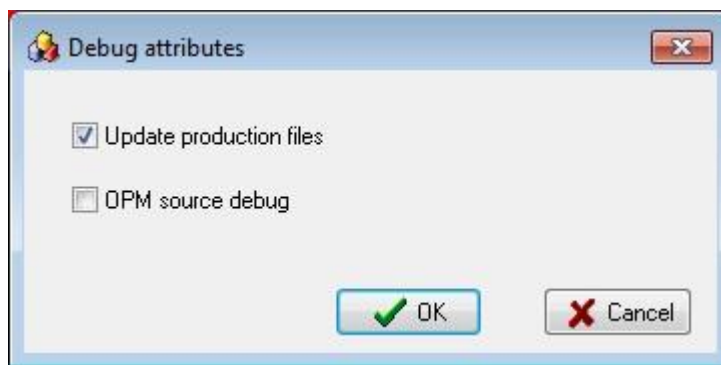
Figure 54

Debug attributes

To change the debug attributes at any time, use the Debug/Debug attributes menu.

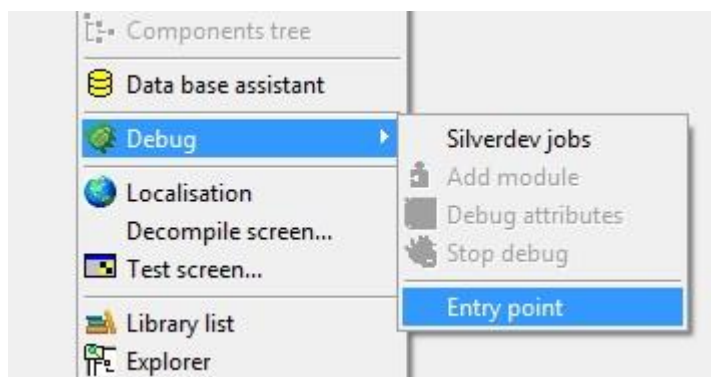
**Figure 55**

The window below enables modification of the debug attributes.

**Figure 56**

Entry point

To debug a job that has not yet started, use the Debug/Entry point menu.

**Figure 57**

The windows enabling selection of the program, module and view are displayed.

When you click in the gutter a SBREAK command is executed.

The debug log window is displayed:

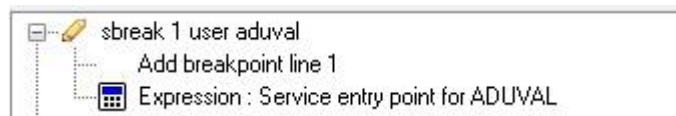


Figure 58

When the program is started by the user identified in the sbreak command, a window is displayed.



Figure 59

You have one minute to click OK.

If you click OK, the new job will be debugged.

Re start a program

When a set of programs is running, you can avoid to start every program from the start when you modify one of them.

The program that you want to start again must be in a named activation group and preferably containing only this program.

In the debugger, in the active program selection window, right click and select "Reclaim an activation group."

Warning, if some forms created by this program are still visible, you cannot use them.

You cannot reclaim the activation group silverdev, and you cannot reclaim an activation group if it contains a program that is in the call stack. (First program or program with a modal form)

Client side debug

Start the Debug.exe program, which is in the SilverDev directory.

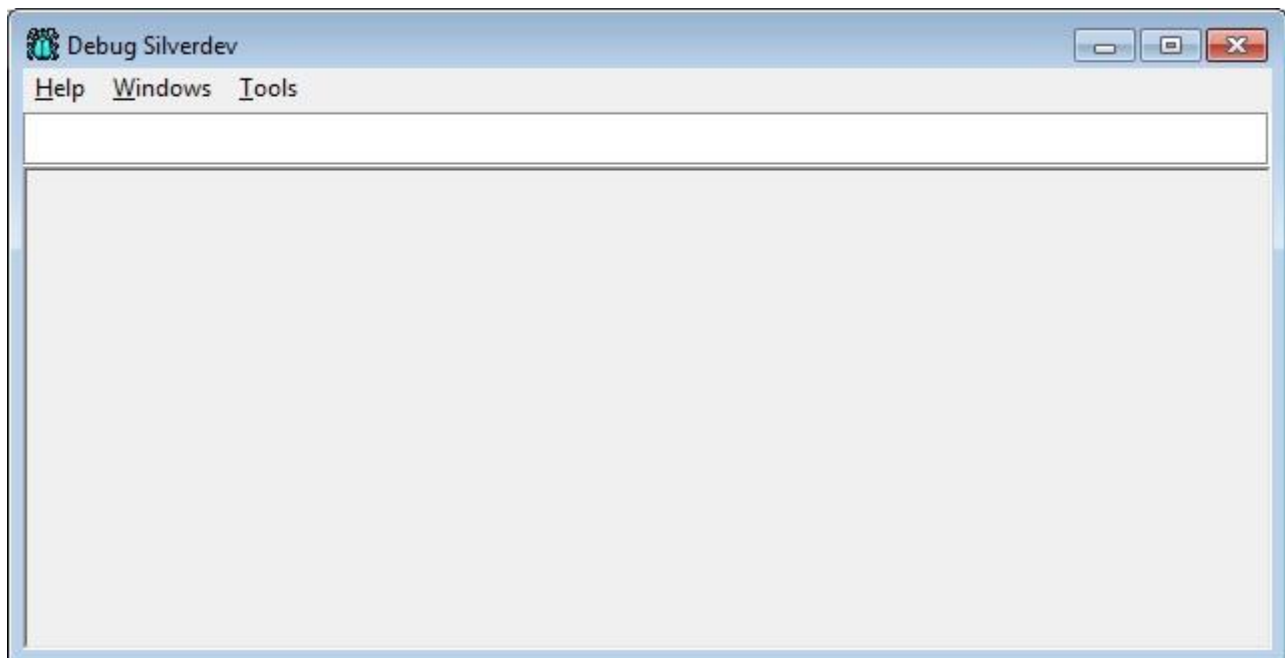


Figure 60

Then, start the SilverDev application to be debugged.

For each new SilverDev application, a window opens in the Debug.exe application.

Note:

If the application is run but the debug is not running, the application can still be debugged



Figure 61

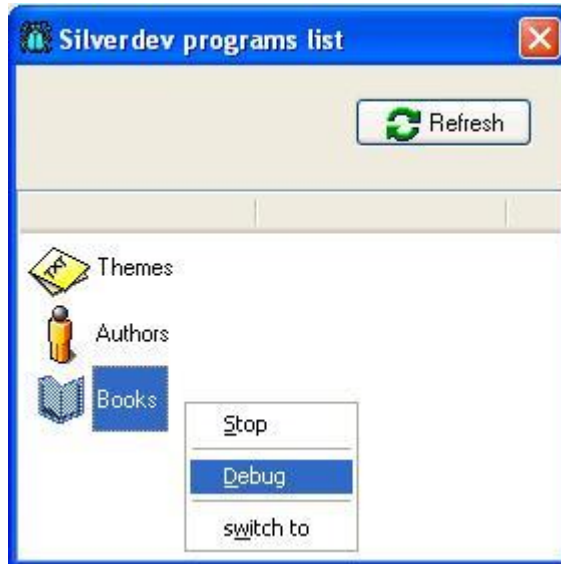


Figure 62

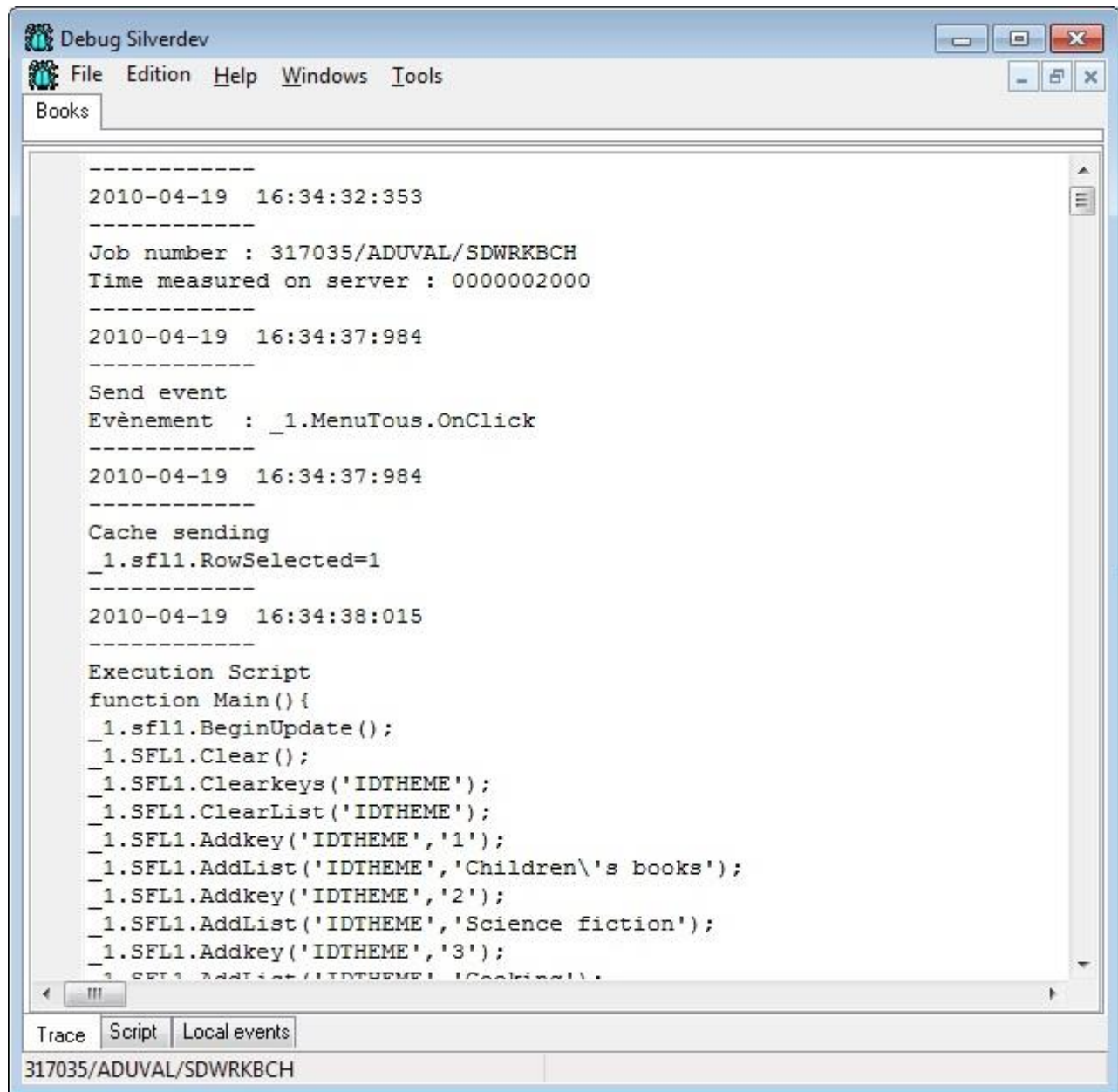


Figure 63

This window contains the Trace and Script tab sheets.
Information about the application is sent to the Trace tab sheet.

Trace tab sheet

The first piece of information sent is the job number associated with the application.
It is therefore possible to run a debug on the System i.

The picture shows that the associated job is 553113/QPGMR/SDWRKBCH

Use the STRSRVJOB 553113/QPGMR/SDWRKBCH command, then use the StrDbg command, as for a traditional program.

Note:

The job number is also included in the status bar at the bottom of the Debug window.

To copy the job number onto the clipboard, right click on the status bar and use the Copy menu.

In the Trace tab sheet, all exchanges between the server and launcher are displayed. This makes it easier to determine the cause of an error.

Script tab sheet

In the "Script" tab sheet, you can write JavaScript code directly and execute it.

Each time a window is received, the list of components is displayed.

The name of the component is accompanied by the name of the window. This name is not known when the window is designed. It is the window's identifier.

To insert this name in the script screen, double click on the name.

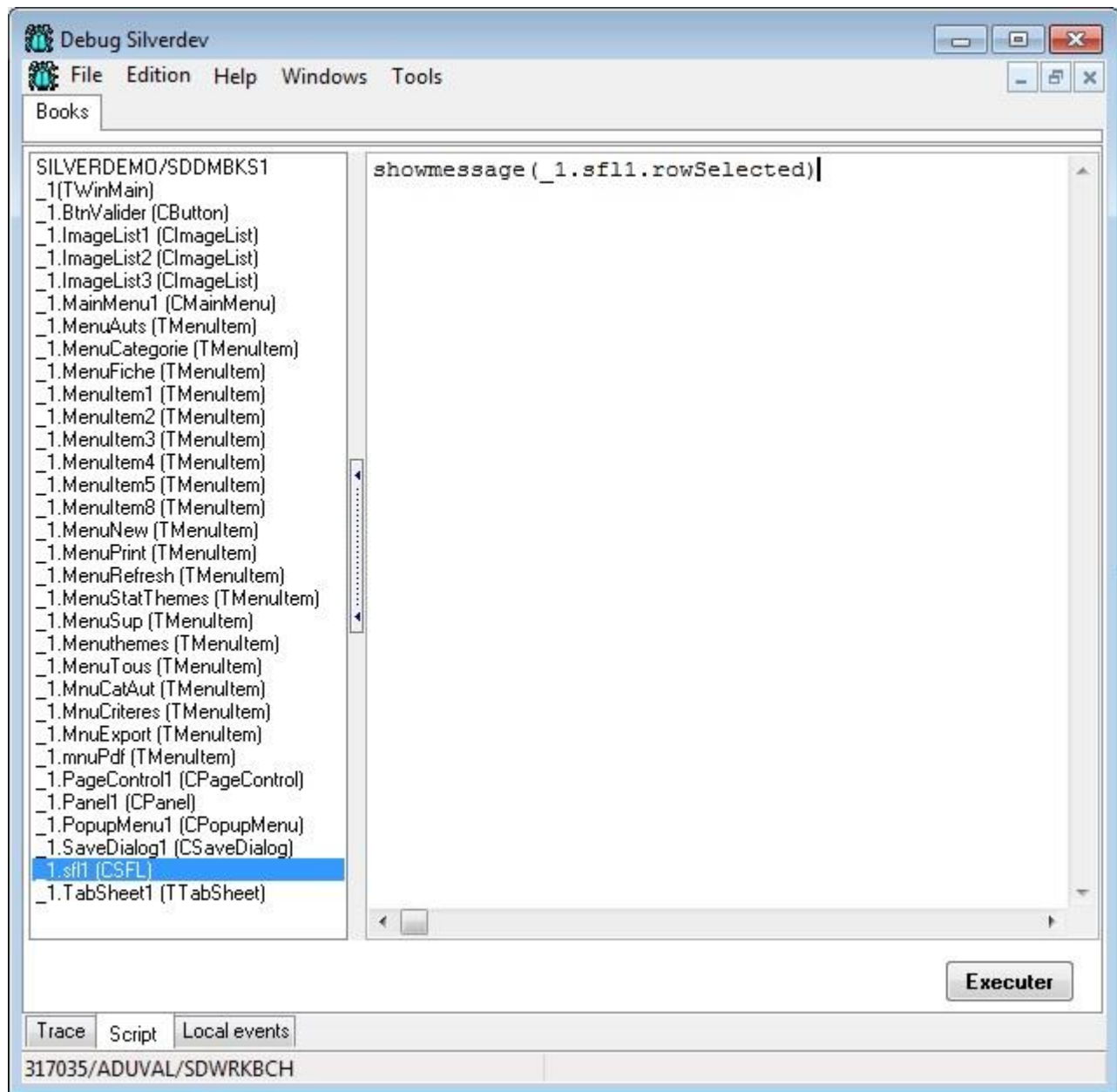


Figure 64

It is then possible to write JavaScript code to run the tests.

Thus, if a line of code in the Trace part appears to be causing problems, you can test it in the Script part.

Similarly, a field can be queried by writing, for example:

```
showmessage(_1.SFL1.rowselected)
```

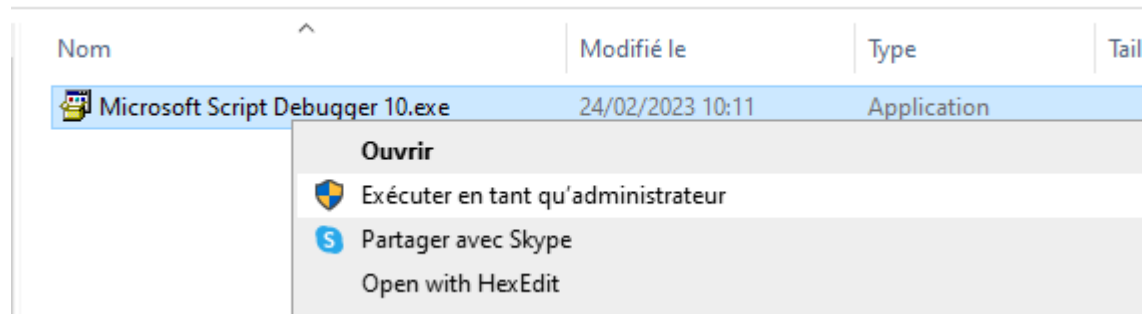
(Click on Execute, the message is then displayed in the application and not in the debugger)

Microsoft Script Debugger

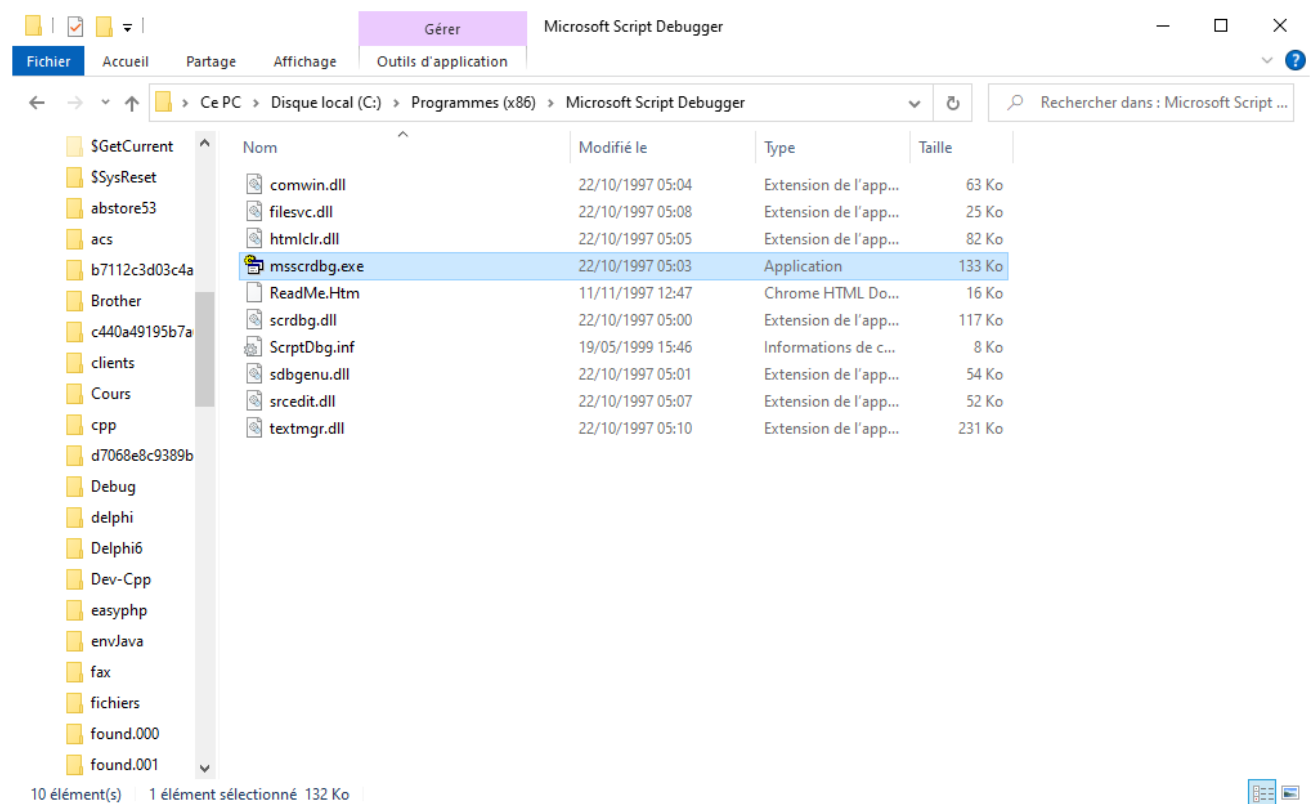
In release folder, there is a sub folder named.

It allows you to install microsoft script debugger

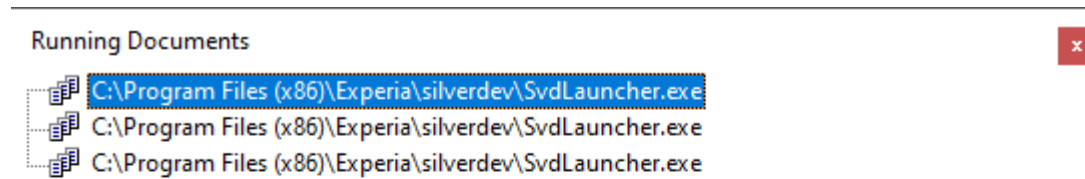
Launch the install program with an administrator profile on your desktop



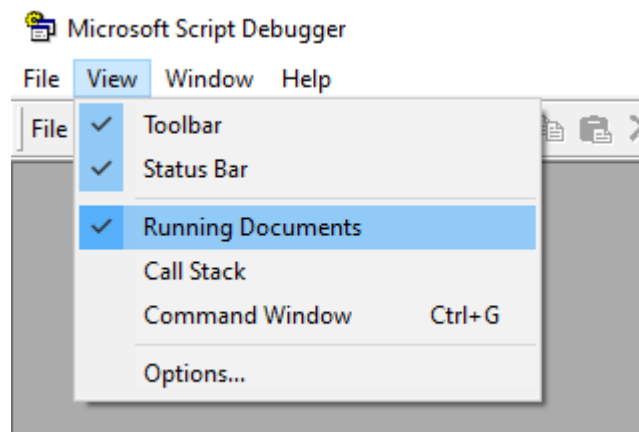
Then start the installed program :



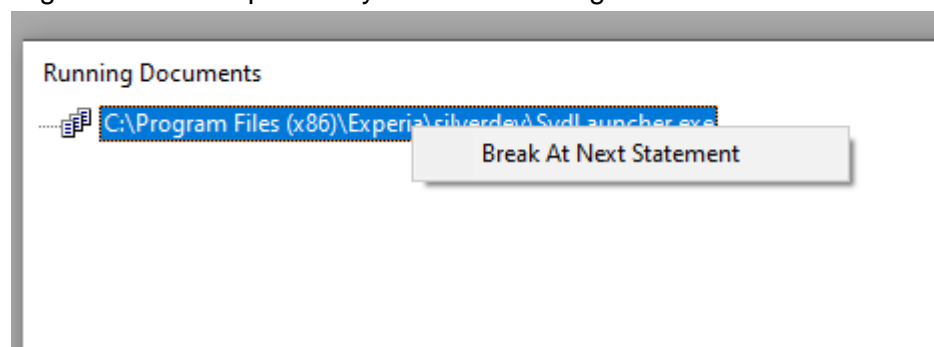
The programm detects all instances of SvdLauncher :



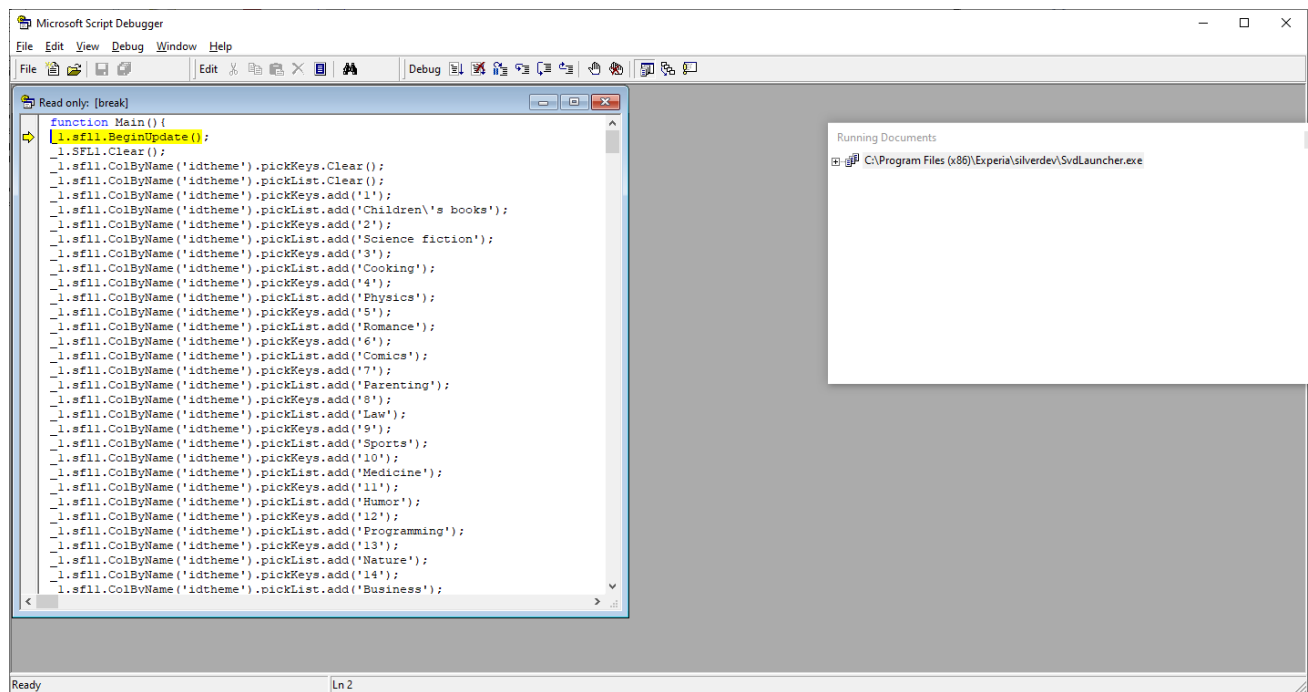
Note : if the window is not visible, check the Running documents option :



Right click on the process you want to debug :



Déclenchez un évènement, le code envoyé depuis le serveur se trouve en debug :



You can debug the program step by step with the F8 key.
Note that local events can't be debugged this way.

Server side debug with 5250

The server side program runs in a batch job.
To debug the job, type in the wrksdjob command.

This command displays the list of the server's jobs.
The number of jobs displayed can be limited by the command parameters.

```

Work with SilverDev jobs (WRKSDJOB)

Type choices, press Enter.

User profile . . . . . *ALL          Name, *ALL, *CURRENT
Select jobs with status . . . . *ALL      *ALL, *ACTIVE, *OUTQ
Include server jobs . . . . . N          Y, N
Include 'MyDesk' jobs . . . . . Y          Y, N
Include Designer jobs . . . . . Y          Y, N

```

The SilverDev server jobs are displayed.

```

Work with SilverDev jobs

SilverDev server is running

2=Change   3=Hold   4=End   5=Work with   6=Release   8=Work with spooled files
10=Job log  11=Call stack  12=Job locks  13=Library list...

Current
Opt Job                                user prf   Stt   Function
  316329/QPGMR/SDWRKBCH                ADUVAL    SELW  *SilverDev-MyDesk
  316330/QPGMR/SDWRKBCH                ADUVAL    SELW  *SilverDev-Designer
  316474/QPGMR/SDWRKBCH                ADUVAL    SELW  /Examples/Books/sddmbks

Bottom

==>
F3=Exit    F4=Prompt  F5=Refresh  F7=Active jobs  F8=Kill MSGW  F10=QCMD
F15=End service job  F16=Sort    F17=Msg *SYSOPR  F23=More options

```

Jobs marked *silverdev-Designer correspond to a connection with Designer.

Jobs marked *silverdev-MyDesk correspond to a connection with MyDesk.

Jobs with no other indication correspond to pre-started jobs.

The other jobs correspond to a SilverDev application.

Function key F8 enables all jobs to be stopped in *MSGW.

Having identified the job, use the strsrvjob command, followed by the number of the job to be debugged.

STRSRVJOB 316474/QPGMR/SDWRKBCH

Option 18 runs this command.

If there are lots of jobs and you cannot identify the one you want, you can use the debug.exe program on the client workstation to identify the job number (see next paragraph).

Then, run the strdbg command:

```
STRDBG PGM(SILVERDEMO/SDDMBKS1)
```

Entry point

If you want to debug a job before launching the application, run a first debug operation:
STRDBG PGM(SILVERDEMO/SDDMBKS1)

Add a service entry breakpoint using the sbreak debug command:

```
SBREAK 1 user myuser
```

When the program is started by the user identified in the sbreak command, a window is displayed.

Display Program Messages

```
Subsystem library list entry QSYS2924 in library list.  
Job 327117/ADUVAL/QPADEV0009 started on 23/04/10 at 14:27:28 in subsystem QB  
Service Entry Point has stopped at line 193 in program SILVERDEMO/SDDMBKS1
```

You can hit F1 on the line for more details:

Additional Message Information

```

Message ID . . . . . : CPI1903          Severity . . . . . : 10
Message type . . . . . : Notify
Date sent . . . . . : 23/04/10          Time sent . . . . . : 14:40:23

Message . . . . . : Service Entry Point has stopped at line 192 in program
                     SILVERDEMO/SDDMBKS1 in job 327008/ADUVAL/SDWRKBCH.
Cause . . . . . : Service Entry Point has stopped at line 192 in program
                     SILVERDEMO/SDDMBKS1 in job 327008/ADUVAL/SDWRKBCH.
Recovery . . . . . : This program must be debugged from a servicing job. Do a
                     Start Service Job (STRSRVJOB JOB(327008/ADUVAL/SDWRKBCH)). Then do Start
                     Debug (STRDBG) on the spawned job from where the STRSRVJOB was done. Set a
                     local breakpoint at or after the Service Entry Point. Return to the original
                     job and press enter to release the spawned job.

```

In another session, run the commands:

```

STRSRVJOB 680493/QSECOFR/SDWRKBCH
and
STRDBG PGM(SILVERDEMO/SDDMBKS1)

```

Comment:

The SBREAK debug command was introduced in version V5R2M0

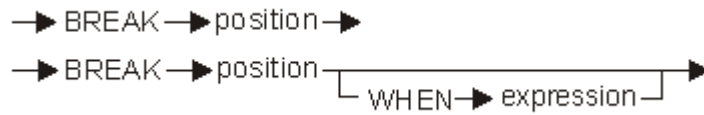
Debug instructions

ATTR

→ ATTR → variable →

The ATTR instruction enables display of a variable's attributes.

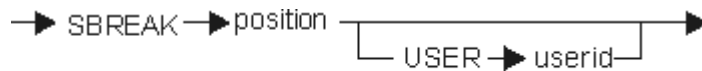
The last call to the QUAL instruction determines the variable's location.

Break

The break instruction enables addition of a breakpoint.

The conditional parameter is optional.

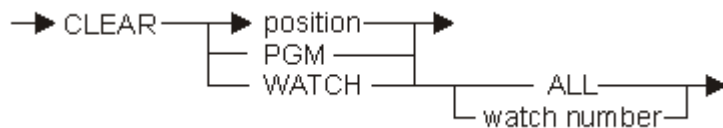
One breakpoint can be replaced by another. The most recent breakpoint is the active one.

Sbreak

The sbreak instruction enables a service entry breakpoint to be entered.

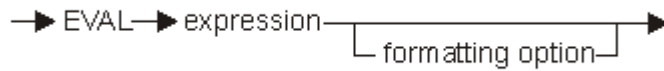
UserId determines the profile under which the job must be run to activate the entry point. If UserId is not defined, the job profile is used.

A service breakpoint and a breakpoint cannot exist at the same time in the same position.

Clear

The Clear instruction enables deletion of a breakpoint or a watch variable.

EVAL



The Eval instruction displays the contents of a value.

The variables can be displayed when the program is suspended. The program is suspended when it comes upon a breakpoint, runs a step by step or if a watch variable is modified.

The format displayed depends on the formatting option parameter.

:c	Character string ebcdic
:x	Hexadecimal format. For a pointer, the reference value is displayed.
:s	Character string (only for modules compiled in C or C++ language)
:f	Character string (only for modules compiled in C or C++ language)
:a	Ascii character string
:u	Unicode format

The formatting option parameter can be followed by a figure to indicate the number of bytes to be displayed. If this figure is omitted, the default values are as follows:

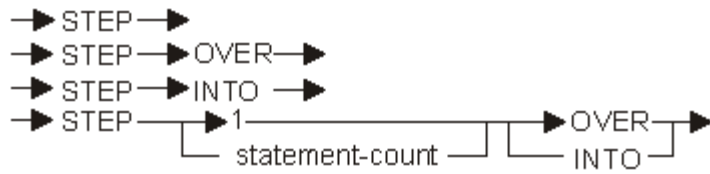
:c	1
:x	Variable length
:s	30
:f	1024
:a	1024
:u	1024

QUAL



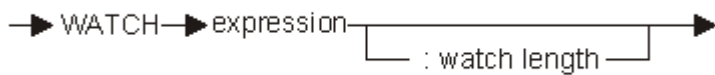
This instruction determines the location for the Eval instructions that follow.

Step Statement



The Step instruction enables step by step progression in the program.

Watch



The Watch instruction enables addition of a watch variable.

If a watch variable is modified, the program is suspended.

Chapter 9. Errors

Server side errors

When an unmonitored error occurs in your program, the program is ended. The associated job is in MSGW.

The user is informed by a dialogue box.

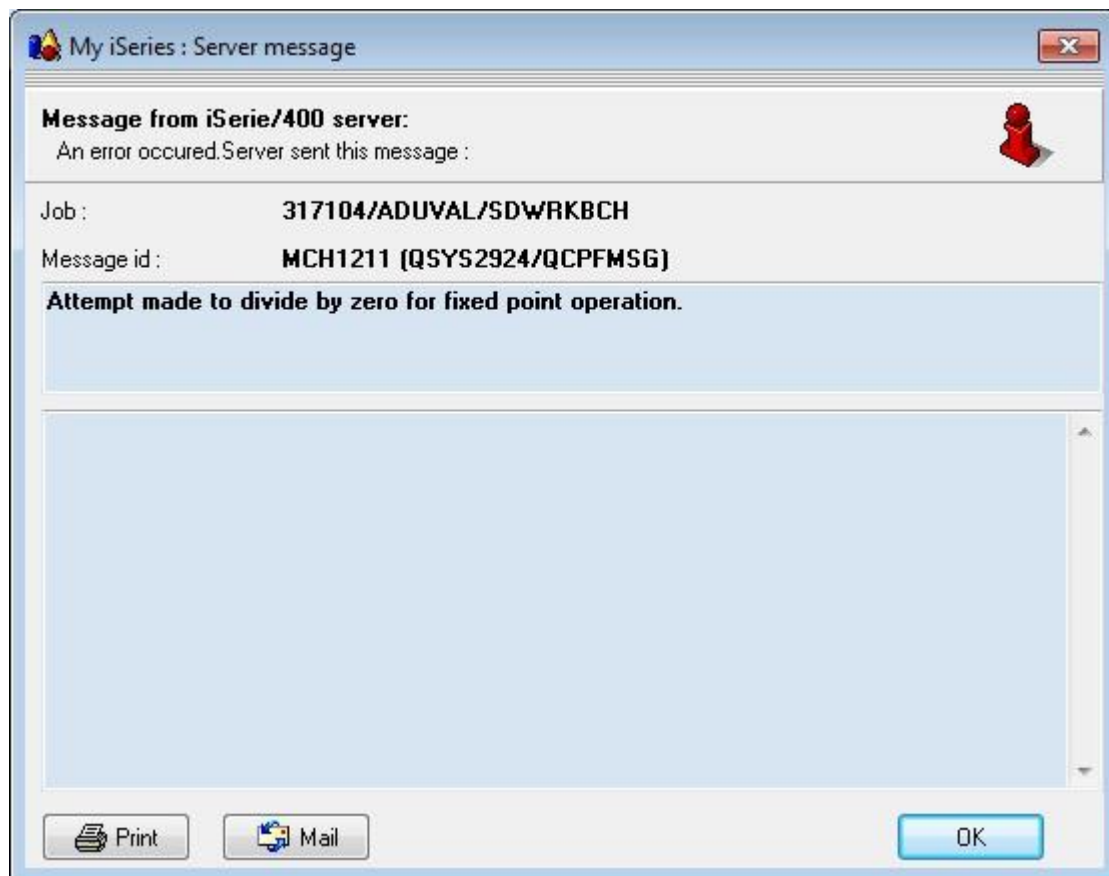


Figure 65

The Mail button enables the report to be sent to an address stored in the CFMAILADM field.

Client side errors

If an error occurs on the client side, message SVD0716 is sent to the program.

An error occurred on the client side

SilverDev ended abnormally

Function check. SVD0112 unmonitored by ABC3 at statement 0000002653,
instruction X'0000'.

The call to SDSTART ended in error (C G D F).

Additional Message Information

Message ID : SVD0721 Severity : 30

Message type : Escape

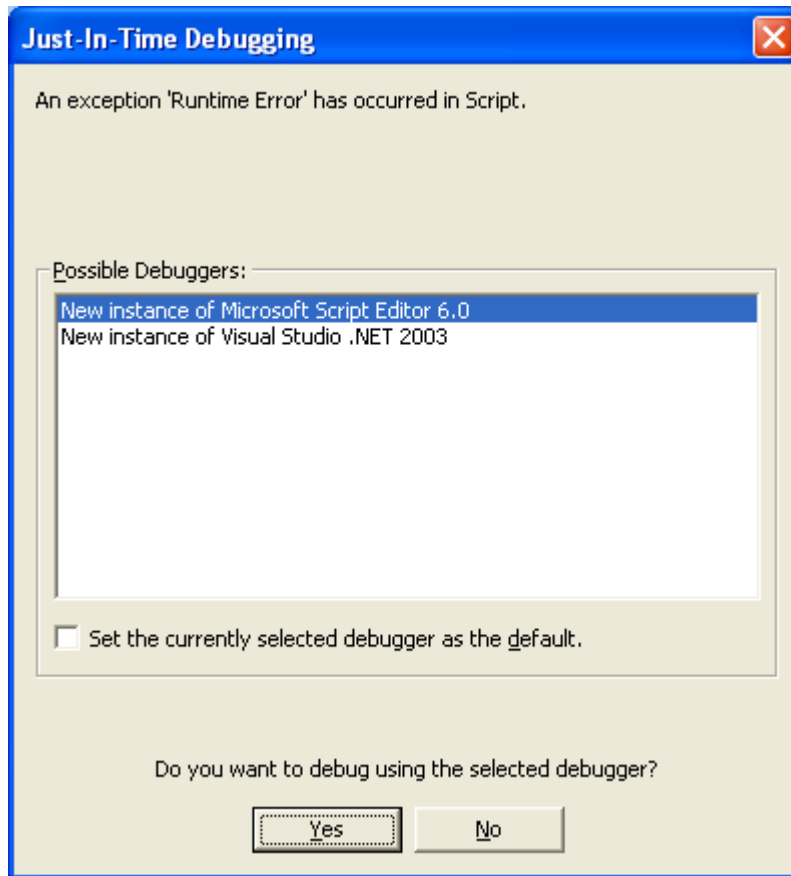
Date sent : 23/04/10 Time sent : 14:48:33

Message : An error occurred on the client side

Error reportStatement : _1.button1.widz=10;Description : Object doesn't
support this property or method

Just-in-time debugging:

If a javascript just-in-time debugger is installed on the workstation, the following window will be displayed. Click No to return to the previous dialogue box.

**Figure 66**

Chapter 10. JOBD programs in MSGW

When a program is in error, the job is in MSGW. Some files can then be locked. To avoid this :

First solution :

Modify the jobd silverdev/silverdev:

```
CHGJOB JOB(SILVERDEV/SILVERDEV) INQMSGRPY(*DFT)
```

Second solution:

If a Silverdev program is in msgw, the exit program associated with the SVDEXCEPT exit point is called.

You can associate an exit program using the addexitpgm command.

The exit program must have the following parameter list:

The Stop parameter can be modified with the 'Y' value, which stops the job.

Third solution:

Add to the program:

```
C      *pssr      begsr
C                      return
C                      endsr
```

Chapter 11. Screen sources

The screen sources are saved on the ifs. These sources can be opened using a text editor. This is very useful in certain cases.

- You want to see the properties that have been modified. Only the properties whose values are different to the default values are saved in the file.
- You want to modify a property on several screens. You can search using a traditional editor to make the modifications directly in the source (in which case, remember to compile the screen with Designer).
- In an extreme case, you can replace one component with another; for example, a CButton with a CBitBtn. In this case, be careful with the properties of the old component that the new component does not have (see Note).

Note:

Be very careful when you modify sources manually.

A syntax error could make it impossible to open the form in Designer.

For example, if you make a mistake on a component type:



Figure 67

or a property name:

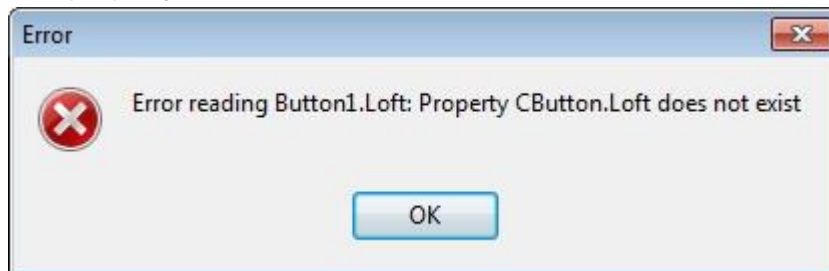


Figure 68

In the latter case, it may sometimes be possible to open the form in Designer, but the misspelled property will have its default value (0 for an integer, false for a Boolean and an empty string for a string)

Source examples:

```
object Win: TWin
  Left = 354
  Top = 291
  Width = 809
  Height = 400
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  OldCreateOrder = False
  Events.Strings = (
    'Button1.OnClick')
  PixelsPerInch = 96
  TextHeight = 13
  object Button1: CButton
    Left = 40
    Top = 80
    Width = 75
    Height = 25
    Caption = 'Button1'
    TabOrder = 0
  end
  object Edit1: CEdit
    Left = 32
    Top = 32
    Width = 121
    Height = 21
    Color = clOlive
    TabOrder = 1
    Text = 'Edit1'
  end
end
```

Chapter 12. Events

Event manager

The components have events. When these events occur, your code will be executed on the as400. The client part indicates that an event has occurred and the program on the as400 runs the procedure associated with this event.

Events are activated during the design stage by ticking the event in the property inspector. Later, we will see that you can also activate/deactivate events during execution.

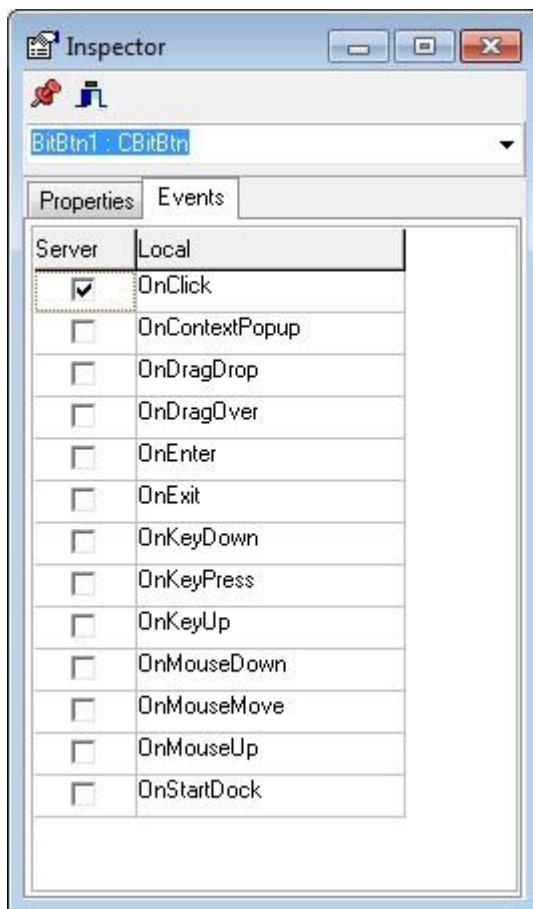


Figure 69

In the RPG program, you associated a procedure to this event with the `sdSetCallBack` function.

```
c          callp      sdSetCallBack(F1
c          : 'BtnUp.OnClick'
c          : %paddr(
c          'BTNUP_ONCLICK'))
```

Event parameters

The procedure associated with the event must have at least the following prototype:

```
pBtnUp_OnClick    B
d                  PI
d PevtInf          *   const options(*nopass)
```

By consulting the Help function, you will see that some events have parameters.

These parameters are actually values passed on by the client program to the RPG program.

In the Help function, you can also see that some parameters can be modified. This means that the value can be changed in the RPG program and the new value will be sent to the client program at the end of the event manager.

If you are using the generator, the list of parameters will be added automatically to the `/*EVENT` type blocks.



For example:

To drag and drop, we have to use these parameters.

The drag and drop is carried out in three stages:

_The user starts the drag from a source component. To do so, change the `dragMode` property of the source component to `dmAutomatic` (you can also leave the value `dmManual`, in which case, you have to call the `sdBeginDrag` function in the `onMouseDown` event).

_The user moves over the destination component.

You can then decide whether the drop is authorised using the `Drag` parameter, which is a modifiable parameter. The cursor will be modified accordingly:  or 

```
pListBox2_OnDragOver...
p                  B
```

```

d                                PI
d PevtInf                        *    const options(*nopass)
d parm                          ds      based(pevtinf)
d Win                            5u 0
d evt                            48a
d drag                           n
d name                           100a   varying
d PosX                           10i 0 overlay(PosXA)
d PosY                           10i 0 overlay(PosYA)
c                                If      Name='ListBox1' and Win=F1
c                                eval    drag = *On
c                                Else
c                                eval    drag = *Off
c                                End
p                                E

```

_ If in the DragOver event, you have authorised the drop, then the OnDragDrop event will occur when the user releases the mouse button.

```

pListBox2_OnDragDrop...
p                                B
d                                PI
d PevtInf                        *    const options(*nopass)
* -----
d List                          s      *    Inz(*Null)
d Index                         s      10  0
d Nbr                           s      10u 0
d Val                           s      20a   Varying
c                                Eval    List=SdGetList(F1:'ListBox1')
c                                If      List = *Null
c                                callp   sdShowMessage('pb')
c                                return
c                                endif
c                                Eval    Nbr=SdGetListCount(List)
c                                Callp   SdBeginUpd(F1:'ListBox2':'Items')
c                                For      Index=0 to Nbr-1
c                                If      SdGetListSelected(List:Index)
c                                Eval    Val= SdGetListLabel(List:Index)
c                                Callp   SdSlAdd(F1:'ListBox2':'Items':Val)
c                                EndIf
c                                Callp   SdEndUpd(F1:'ListBox2':'Items')
c                                Callp   SdFreeList(List)
c                                EndFor
p                                E

```

Some event parameters are of the set type (see Chapter 19).
To see if a value is part of a set, use the `sdIsInSet`.

For example:

The `onMouseDown` event contains the parameter `Shift` which can contain the following values:

0	The SHFT key is pressed.
1	The Alt key is pressed.
2	The Ctrl key is pressed.
3	The left mouse button is pressed.
4	The right mouse button is pressed.
5	The central mouse button is pressed.
6	A double click has been made.

```
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
D PosX          10i 0
D PosY          10i 0
C          if          sdIsInSet(Shift:3)
C          ...
C          endif
p          E
```

Note:

Several values can be contained in a set.

```
*/EVENT SFL1_OnChangeCellValue
d parm          ds          based(pevtinf)
```

```

d Win          5u 0
d evt          48a
  *Modifiable
d Allow        N
  *Non modifiable
d ColName      100a  varying
d Row          10i 0
d Value        100a  varying

```

In the onchangeCellValue event, if you query the value of the cell using sdGetCell, you will not obtain the same value as that passed in the value parameter.

In fact, this event occurs before validation, which enables modification of the allow parameter.

Dynamic allocation of events

It is also possible to activate an event during execution using the sdEnableEvt function.

Prototype:

```

d sdEnableEvt  pr
d pform          5u 0 const
d eventName    48a  varying value
d Enabled      N    value

```

Note:

A procedure must correspond to each event activated in Designer or using the sdEnableEvt function. This association is made using the sdSetCallBack function.

Examples of use:

Case 1:

When you modify the value of the checked property of a CCheckBox component by code, the onclick event is triggered. If you do not want the event to be triggered, use the sdEnableEvt function to temporarily deactivate the onclick event.

```

C          callp  sdEnableEvt (F1: 'CheckBox1.onclick': *OFF)
C          callp  sdSetBool (F1: 'CheckBox1': 'Checked': *ON)
C          callp  sdEnableEvt (F1: 'CheckBox1.onclick': *ON)

```

Case 2:

Although SilverDev was not designed for this, it is possible to create components dynamically. You can thus request notification of events for these dynamically created components.

```

D i          s          10u 0
D j          s          10u 0
D name1      s          50    varying
D name2      s          50    varying
C            for        i=0 to 5
C            eval       Name1='Mnu'+%char(i)
C            callp      sdCreate(F1: 'TMenuItem':Name1)
C            callp      sdAddMnuItem(F1: 'menu1.items':F1:Name1)
C            callp      sdSetString(F1:Name1: 'caption':
C            'Elt'+%char(i))
C            for        J=0 to 5
C            eval       Name2='Sub'+%char(i)+'_'+%char(j)
C            callp      sdCreate(F1: 'TMenuItem':Name2)
C            callp      sdAddMnuItem(F1:Name1:F1:Name2)
C            callp      sdSetString(F1:Name2: 'caption':
C            'Sous Elt'+%char(i))
C            callp      sdEnableEvt(F1: Name2+'.OnClick':*ON)
C            callp      sdSetCallBack(F1:Name2+'.OnClick':
C            %PADDR('EVT1'))
C            EndFor
C            EndFor

```

Chapter 13. Modal forms

Introduction

Some windows can be displayed in modal. Modal display is blocking, which means that within an application, only this form is accessible to the user.

ModalResult

To know the user's action, query the `modalResult` property of the form or the return value of the `sdShowModal` function.

Caution:

Do not confuse the `modalresult` property of a `CButton` component with that of the form. These two properties are related but do not have the same meaning.

For the window, changing the `modalResult` property to a value other than `mrNone` closes the window.

For a button, the `modalResult` property means that the button will transmit its value to the `modalResult` property of its window when the user clicks on the button.

Example

In an initial program, you can call a second program whose window is modal by clicking on a button:

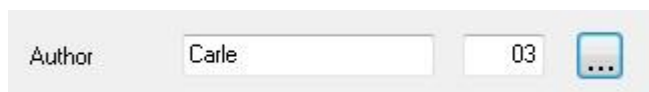


Figure 70



Figure 71

Program called: (modal)

C	*entry	plist		
C		parm	\$F1ModRes	
C		parm	PID	2 0

~/EVENT BtnOk_OnClick				
, * ----- *				
, * Description :				
, * ----- *				
C	Eval	PID	=sdGetNum (F1: 'Liste': 'KeySelected')	
C	callp	sdSetInt (F1: '*FORM': 'ModalResult': Mrok)		

Caller pProgram:

~/EVENT BtnAuthors_OnClick				
, * ----- *				
, * Description :				

```

, * ----- *
DModRes      s      10u 0
C      call      'SDDMBKS3'
C      parm      ModRes
C      parm      PID      2 0
C      if      ModRes=MrOk
C      callp      sdSetNum (F1: 'IDAUT': 'value':PID)
C      endif

```

Note:

The modal window can have the following properties:

Position = poScreenCenter to make it appear in the middle of the screen

BorderIcons = only biSystemMenu activated

Two buttons:

A button marked “cancel” whose modalResult property is mrCancel and the Cancel property is true. Thus, when the user presses “Esc”, the window is closed.

A button marked “OK” whose default property is true. The modalResult property of the OK button can be mrOk, but it is often useful to make it mrNone and to trigger the button’s onclick event, run tests, then modify the modalResult property of the window by code.

Execution only

A window can only be displayed in modal if it is not visible.

If you try to display a visible window in modal, you will see the following message:

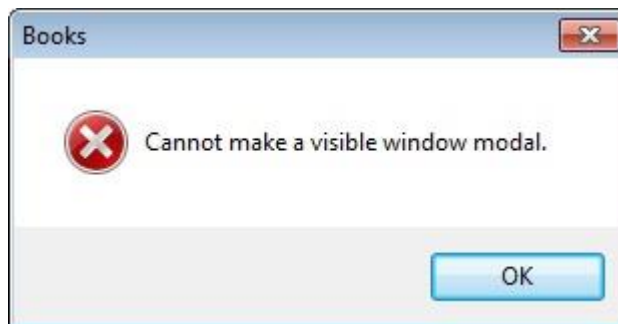


Figure 72

The first window in a SilverDev application is automatically visible when it is created.

In certain cases, you will create an application with a modal window to be called from another window, but you will want to run it directly from MyDesk.

You will then see the above message, which will appear when you run the application alone. To avoid this message, add this instruction to the initstart moment:

```
*/BLOCK RPGINITSTART
* ----- RPGINITSTART : Begining of Initializations procedure
C          callp      sdHideMainForm
```

Add this instruction to the Aftershow moment:

```
*/BLOCK RPGAFTERSHOW
* ----- RPGAFTERSHOW : After show(F1)
C          if          F1=1
C          callp      sdEnd
C          endif
```

Chapter 14. Printouts

Simple printing

In this section, we will distinguish simple printouts from list type printouts. In a simple printout, each band is printed once; in list type printouts, certain bands are printed several times.

The report palette components enable creation of printouts and print previews.

A printout mock-up will be created on a form. The form will not be visible (the user does not need to see the mock-up), however, it has to be created using the `sdCreateForm` function.

On this form, we will place `CReport` component and the `CRepBand` components. On the `CRepBand` components, we will place components like `CRepLabel`, `CReplImage`, `CRepCheckbox`, `CRepShape`, `CRepSysField`, `CRepChart` and `CrepMemo`.

The `sdPrint` and `sdPreview` instructions enabling print launch refer to the `CReport` component.

Print preview

To produce a print preview, use the `sdPreview` function.

A window is displayed for the preview. This window enables the user to browse through the various pages, zoom and select the pages to be printed.

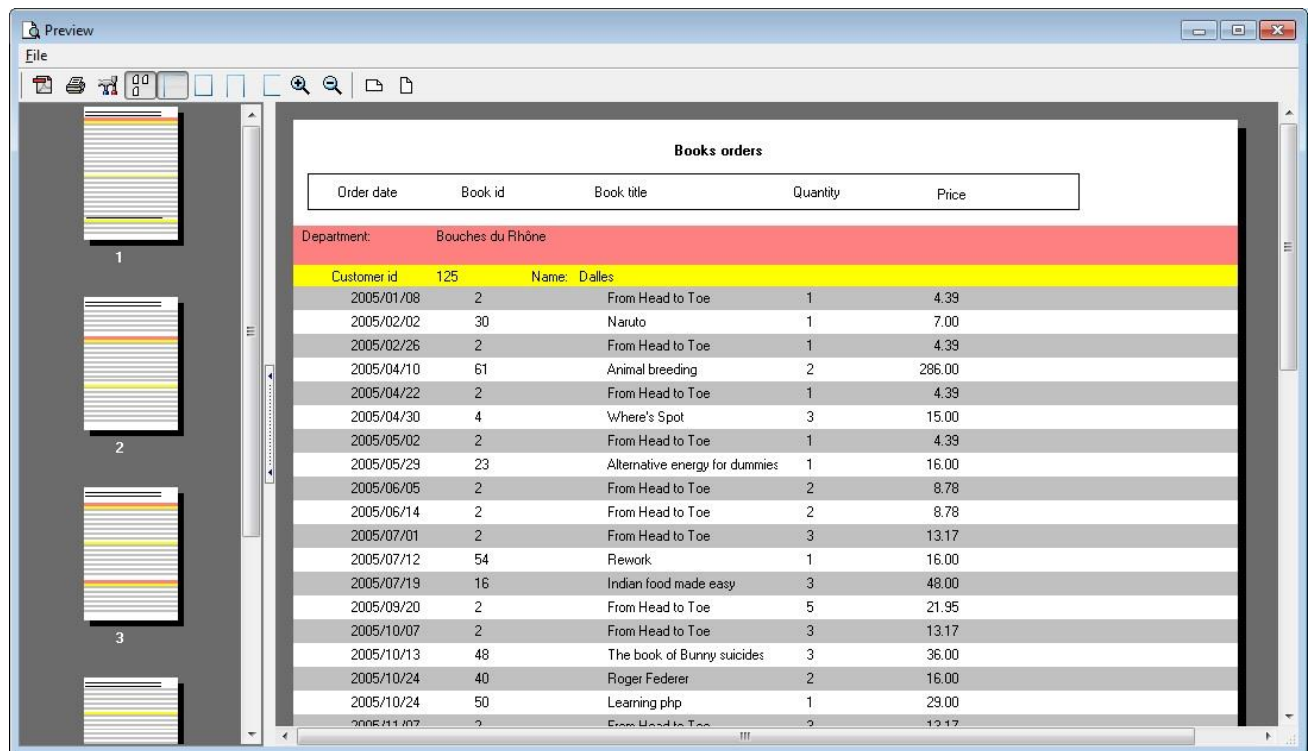


Figure 73

Orientation

The CReport component has an Orientation property that can have one of three values:

PoLandScape	Landscape printing
PoPortrait	Portrait printing
PoUser	Value chosen by the user in the print dialogue box (sdPrintDlg)

Printing a list

If you want to print a list with a number of items that is unknown at the design stage, change the BandType property of the CRepBand component to alr_Detail.

For CRepLabel, CRepImage, CRepCheckBox, CRepChart and CRepMemo components, you can add a list of values using the sdAddValue, sdAddImg, sdAddLines and sdAddChart functions. For each value added by these functions, the band will be printed once. The component will take care of page changes.

Before reloading the list of values, start by emptying all the values in the memory using the sdClearData function.

CRepLabel

The CRepLabel component is the most commonly used component for printing.

To modify its value in execution for a simple printout, change its caption property; for a list type printout, use the sdAddValue function.

CRepMemo:

For the CRepMemo component, for each occurrence, feed the list of a CMemo type component with the standard sdSIAdd functions, then assign the value of the occurrence with the sdAddLines function.

```
p Comments...
P          B
d          PI
C          callp      sdSIclear(F2:'Memo1.lines')
C      A_IDBOOK      SETLL      SDDMCMLV
C      A_IDBOOK      READE      SDDMCMLV          5
C          DOW        NOT *IN58
C          callp      sdSIadd(F2:'Memo1.lines':
C                      %trim(E_LINE))
C      A_IDBOOK      READE      SDDMCMLV          5
C          ENDDO
C          callp      SDADDLINES(F2:'RepMemo1':F2:'Memo1.lines
P          E
```

CRepChart:

For the CRepChart component, feed a CChart type component, then assign the value of the occurrence with the sdAddChart function.

```
p Graph...
P          B
d          PI
DNB          s          10  0
C      KEY1          KLIST
C          KFLD          A_IDBOOK
C          KFLD          YEAR          4  0
C          KFLD          MONTH          2  0
C          eval          YEAR=2010
C          callp      sdSeriesClear(F2:'Series1')
C          eval          MONTH = 1
C          dow          MONTH <= 12
```

```

C          eval      NB=0
C      KEY1  SETLL    SDDMORDER1
C      KEY1  READE    SDDMORDER1
C          dow      not *in57
C          eval      NB=NB+D_QUANTITY
C      KEY1  READE    SDDMORDER1
C          enddo
C          if        NB>0
C          callp      sdAddPie (F2: 'Series1':NB:MONTHNAME (MONTH) )
C          endif
C          eval      MONTH=MONTH+1
C          enddo
C          callp      sdAddChart (F2: 'RepChart1':F2: 'Chart1')
P          E

```

Formatting an occurrence

To change the appearance of a value (background colour, font, etc.), use the optional parameters 4 and 5 in the `sdAddValue` functions. These parameters refer to a `CRepAttributes` type component. The occurrence will be printed with the properties of the `CRepAttributes` component.

```

C          if        A_STOCK =0
C          callp      sdAddValue (F2: 'STOCK':%char (A_STOCK)
C                      *ON:F2: 'Attr1')
C          else
C          callp      sdAddValue (F2: 'STOCK':%char (A_STOCK)
C          endif

```

Note:

For a simple printout, change the component properties directly.

Group printing

To organise a break in a list type printout, proceed as follows:

- Create an `alr_detail` type band and an `alr_link` type band
- Add an element in the “Band Detail” Groups collection.

- Specify the DataSource property for this element to determine the field in which the break will occur. The DataSource property must indicate a field in the detail band.
- Fill in the GroupFooter and GroupHeader properties of this element to determine the bands printed at each break.
- Only fill in the fields on the detail type band.

Example 1:

Place 2 bands, one alr_Detail and alr_Link, that we will call "BandDetail" and "BandHeader".

Place a CRepLabel type component called NOCLI on BandDetail.

Place a CRepLabel type component called NOCLI2 on BandHeader.

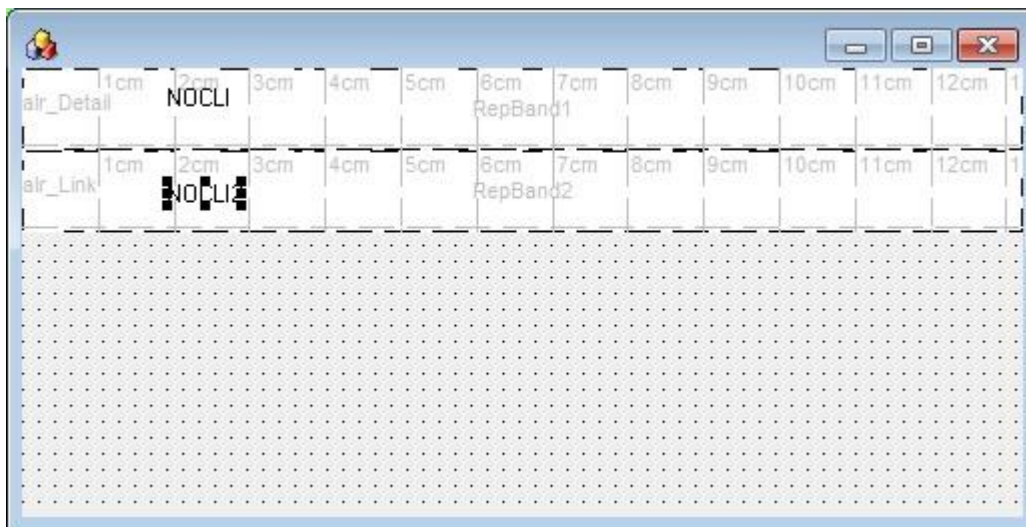


Figure 74

Add an element to the "BandDetail" Groups collection.

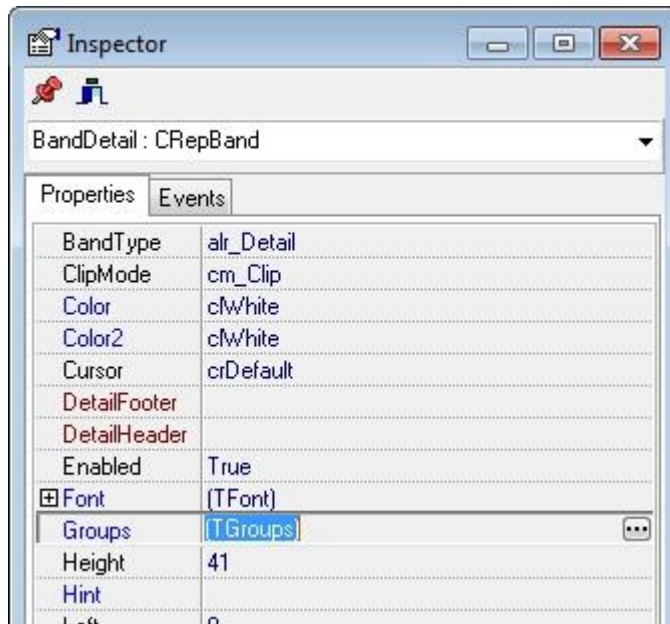


Figure 75

Change the DataSource and GroupHeader properties of this element.

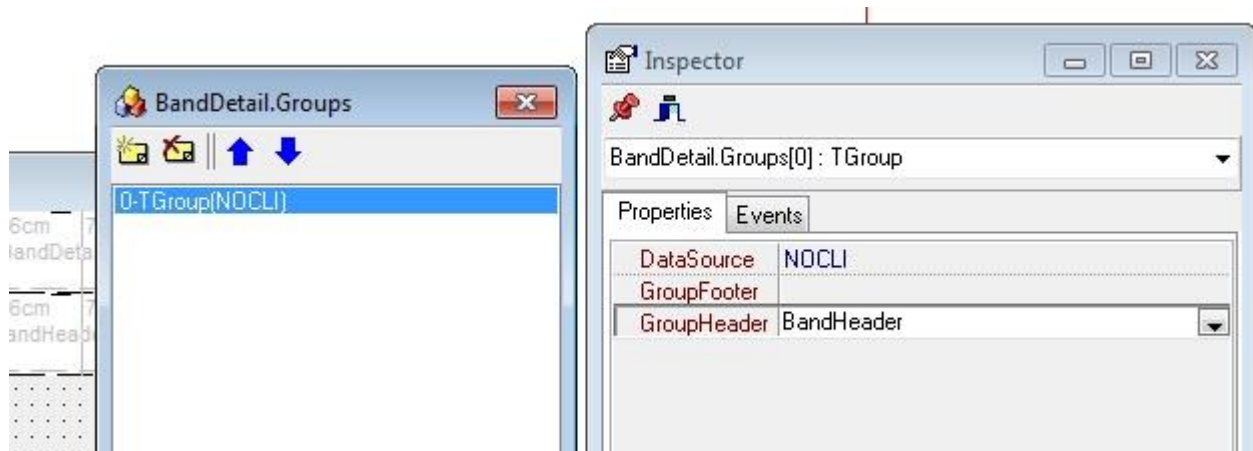


Figure 76

In the program, fill in the NOCLI field only (not NOCLI2) using the sdAddValue function.

```
C                                     callp      sdAddValue (F1 : 'NOCLI' : %char (NOCLI) )
```

Every time the NOCLI value changes, the "BandHeader" will be printed.

Comment:

There may be several breaks. To see an example of a double break, see the demo programme SDDMORDERS, and the SDDMORDERS screen.

Example 2 (corresponding to silverdemo/sddmorders):

Mockup :

Books orders								
Order date	Book number	Book title	Quantity	Price				
Departement : DEP2 Client id: NOCLI2 Name: NOMCLI2								
Date	IDBOOK	TITLE	Quantity	Price	idcust	namecust	TOTAL	DEP
Total				Total				
					Page n°	#Page#		

Figure 77

Result:

Books orders

Order date	Book number	Book title	Quantity	Price
Department: Bouches du Rhône				
Client id: 125 Name: Dalles				
2005/01/08	2	From Head to Toe	1	4.39
2005/02/02	30	Naruto	1	7.00
2005/02/26	2	From Head to Toe	1	4.39
2005/04/10	61	Animal breeding	2	286.00
2005/04/22	2	From Head to Toe	1	4.39
2005/04/30	4	Where's Spot	3	15.00
2005/05/02	2	From Head to Toe	1	4.39
2005/05/29	23	Alternative energy for dummies	1	16.00
2005/06/05	2	From Head to Toe	2	8.78
2005/06/14	2	From Head to Toe	2	8.78
2005/07/01	2	From Head to Toe	3	13.17
2005/07/12	54	Rework	1	16.00
2005/07/19	16	Indian food made easy	3	48.00
2005/09/20	2	From Head to Toe	5	21.95
2005/10/07	2	From Head to Toe	3	13.17
2005/10/13	48	The book of Bunny suicides	3	36.00
2005/10/24	40	Roger Federer	2	16.00
2005/10/24	50	Learning php	1	29.00
2005/11/07	2	From Head to Toe	3	13.17

Figure 78

Drawing a table

To draw a table, there is no CRepGrid component.

Use different CRepShape components, next to one another.

In the example below, square 2 begins exactly where square 1 ends,

The Sides property of square 2 has no siLeft otherwise, there would be a thicker line where the two squares touch.

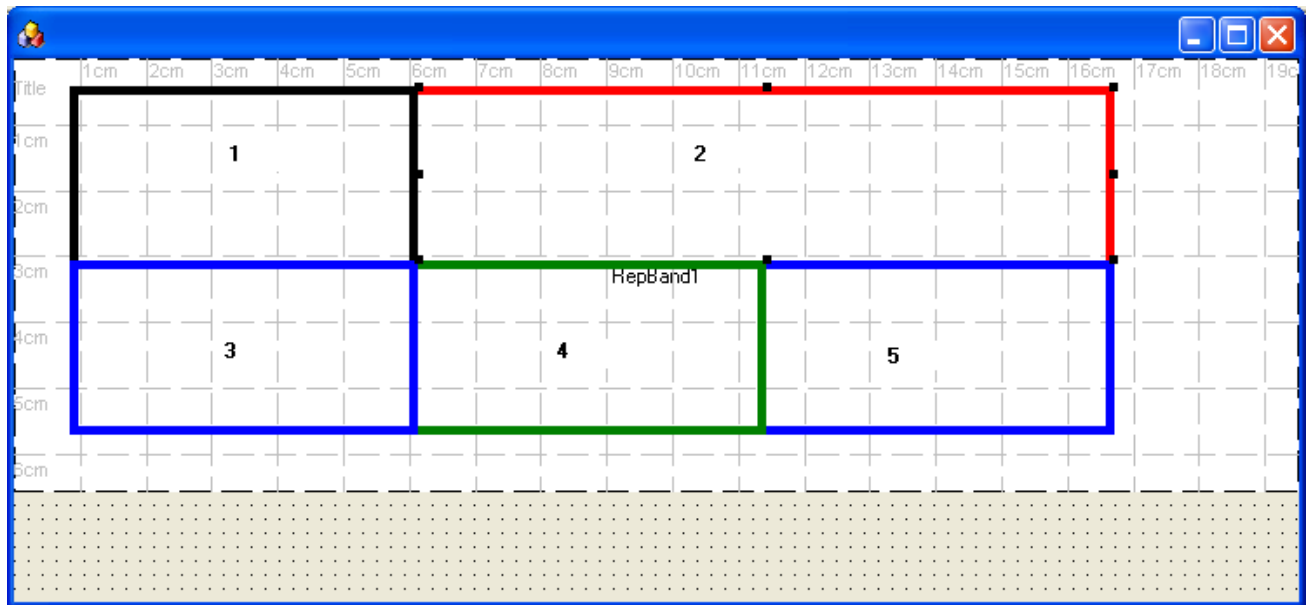


Figure 79

Note:

You may be tempted to overlap the squares to leave only one edge visible. Although the result obtained in Designer may be satisfactory, it may not be so when printed. The two edges supposedly overlapping may form a thicker edge or even two separate lines. This is why we recommend the method explained above.

Value reports:

If you want to display a report as a page header, use a `alr_PageHeader` type band with a `CRepLabel` component. Call the `CRepLabel` component the same name as the Detail band with the “_” character at the end of the field.

Change the Previous property to true for the previous Detail band value.

Page breaks

To make a printout over two pages, place two `alr_title` type bands.

On the second band, change the `NewPage` property to true.

Similarly, for a list type printout, you can start each new page at each group change by changing the `NewPage` property of the `GroupHeader` type band.

Formatting in the middle of a string

There is no equivalent of the `RichEdit` component for printouts.

To display a word with a different font in the middle of a sentence, you can use two CrepLabel components and move the second one as follows:

```
Dlabel1      s      50      varying
C      eval      label1 = sdGetFormName(F1)+'.replabel1'
C      callp      sdSetString(f1:'replabel1':'caption':
C      'City :')
C      callp      sdRefresh(f1:'replabel1')
C      callp      sdSetString(f1:'replabel2':'caption':
C      'Madrid')
C      callp      sdSet(F1:'replabel2':'left':
C      label1+'.left' + '+' label1+'.width')
C      callp      sdSet(F1:'replabel2':'top':
C      label1+'.top' )
C      callp      sdpreview(F1:'report1')
```

PDF printing

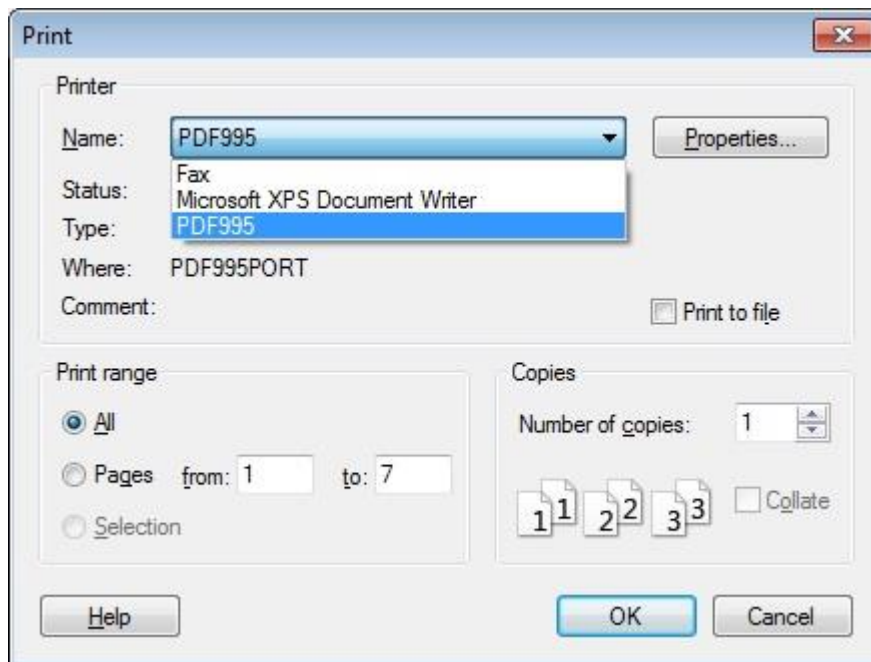
Instead of calling the sdPrint function, call the sdSavePdf function to save the report as a pdf file.

See below for the sdSavePdf function prototype:

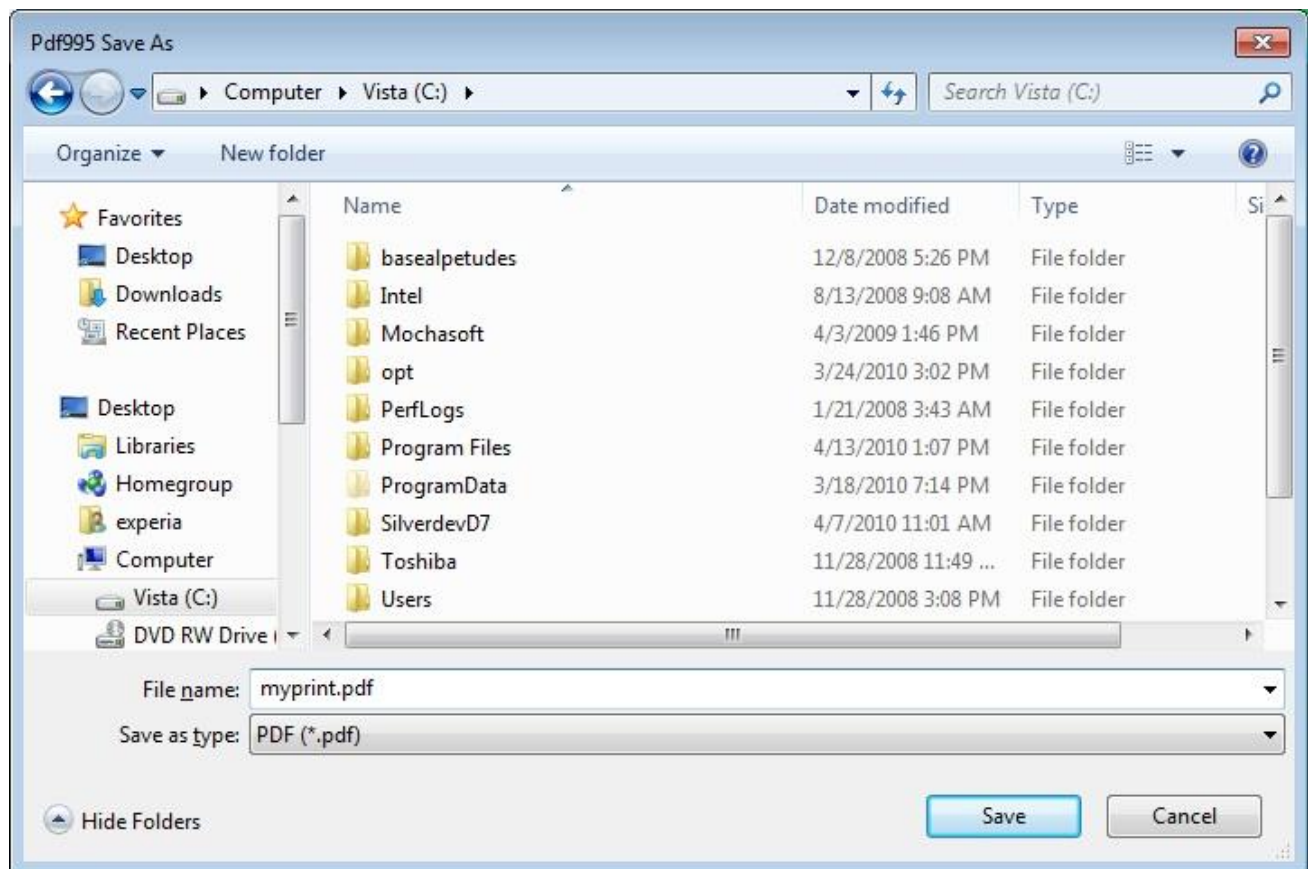
```
d sdSavePdf      pr      1000      varying
d pform      5u 0 const
d Component      30      varying value
d Path      1000      varying value options(*NOPASS)
d Erase      N      value options(*NOPASS)
d Code      10i 0 options(*NOPASS)
```

If the path parameter value is '*SELECT', a dialogue box will ask the user where the file is located.

Another way of generating a pdf document is to use the virtual pdf printer.

**Figure 80**

This printer will ask for a file name under which to save the pdf.

**Figure 81**

Selecting the printer

If you want to allow the user to choose a destination printer, use the `sdDialog` function with a `CPrintDialog` component.

```
C      if      sdDialog(F1: 'PrintDlg1')
C      callp   Load
C      callp   sdPrint(F2: 'Rep1')
C      endif
```

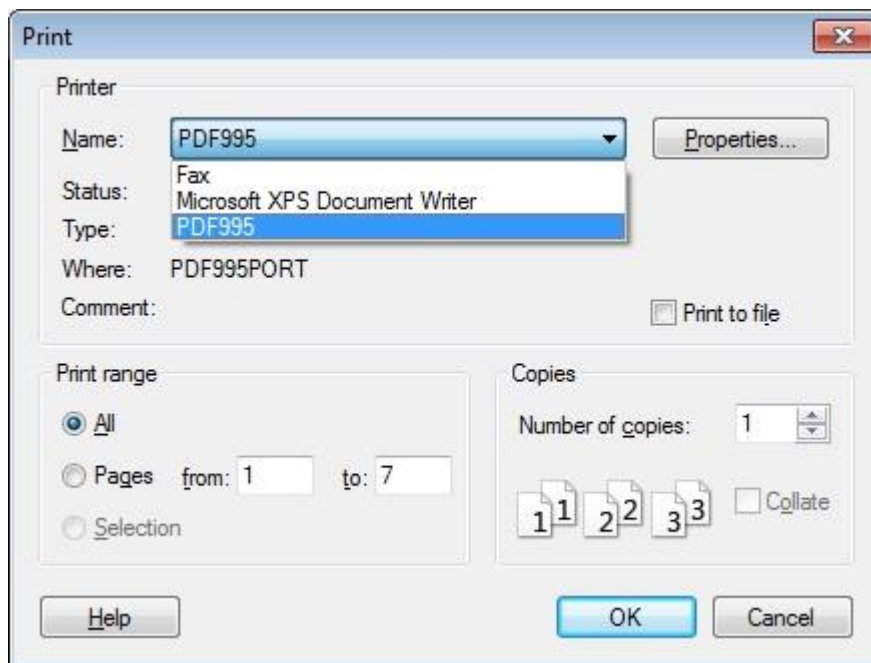


Figure 82

Imposing a printer

To impose a printer, use the `CPrinter` component to handle the printer.

```
Dlist      s      *
D i        s      10u 0
C          eval    list = sdGetList(F1: 'printer1.printers')
C          for     i = 0 to sdGetListCount(List) - 1
C          if      %trim(sdGetListLabel(List:i)) = 'PDFCreator'
C          callp   sdSetInt(F1: 'Printer1': 'PrinterIndex': i)
C          leave
C          endif
C          endfor
```

C	callp	sdPrint(F1:'sfl1')
---	-------	--------------------

FromPage ToPage

To print some of the pages, use the FromPage and ToPage properties.

These properties exist on the CReport component and on the CPrintDialog component, but you must transfer the values from one to the other.

For example:

```

DFromPage      s      10i 0
DToPage        s      10i 0
DprintRange    s      10i 0
C              if      sdDialog(F1:'printDialog1')
C              eval    printRange=sdGetInt(F1:'PrintDialog1':
C                      'PrintRange')
C              if      PrintRange=2
C              eval    FromPage = sdGetInt(F1:'PrintDialog1':
C                      'FromPage')
C              eval    ToPage = sdGetInt(F1:'PrintDialog1':
C                      'ToPage')
C              else
C              eval    FromPage = 0
C              eval    ToPage = 0
C              endif
...
C              callp    sdPrint(F1:'Repl')
C              endif

```

mm/Pixel conversion

Here's a little math.

D is the distance in mm on the printed page.

M is the printer margin in mm (all printers have a margin).

The left property of a component is in pixels.

The default pixel per inch value of a screen (PPP) is 96 pixels/inch.

An inch is 25.4mm.

To print a component at distance Dmm on a page, the component must therefore be placed with the left property:

(D-M)/25.4 x 96

Chapter 15. Tools menu

Command

AS400 commands can be executed.

The list of parameters is constituted dynamically.

Use the page up/page down buttons to scroll through the parameters.

Inactive commands are not permitted in the Designer program.

Example with the CRTPGM command:

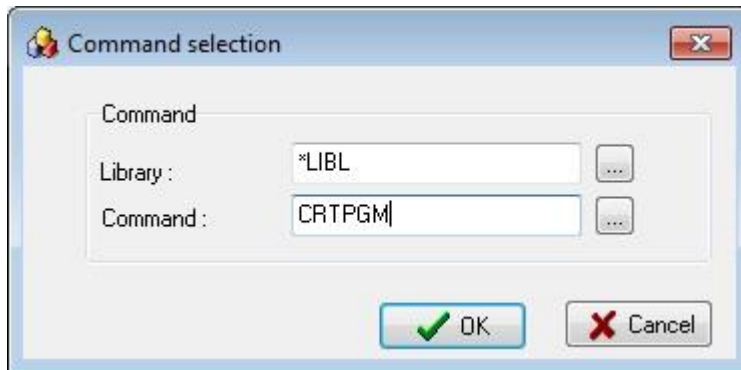


Figure 83

***LIBL/CRTPGM**

Prompt Text

ACTGRP
ALWLIBUPD
ALWRINZ
ALWUPD
AUT
BNDDIR
BNDSRVPGM
DETAIL
ENTMOD
IPA
IPACTLFILE
MODULE
OPTION
PGM
REPLACE
STGMDL
TEXT
TGTRLS
USRPRF

PGM

Programme Nom

Bibliothèque Nom, *CURLIB

MODULE

Module Nom, générique*, *ALL

Bibliothèque Nom, *LIBL, *CURLIB, *USRLIBL

TEXT

Texte 'descriptif' Valeur alpha, *ENTMODTXT...

ENTMOD

Module procédure entrée pgm Nom

Bibliothèque Nom, *LIBL, *CURLIB, *USRLIBL

BNDSRVPGM

Lier programme de service Nom, générique*, *ALL

Bibliothèque Nom, *LIBL

BNDDIR

Répertoire de liage Nom

Bibliothèque Nom, *LIBL, *CURLIB, *USRLIBL

Execute Cancel

Figure 84

Utilisez le menu outils/commandes pour exécuter une commande as400

La liste des paramètres est constituée dynamiquement.

Utilisez les touches page down et page up pour faire défiler les paramètres.

Les commandes interactives ne sont pas admises dans le programme Designer.

Exemple avec la commande CRTPGM :

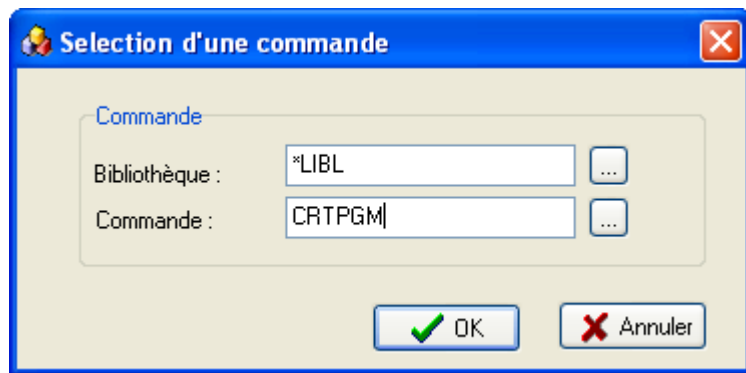


Figure 85

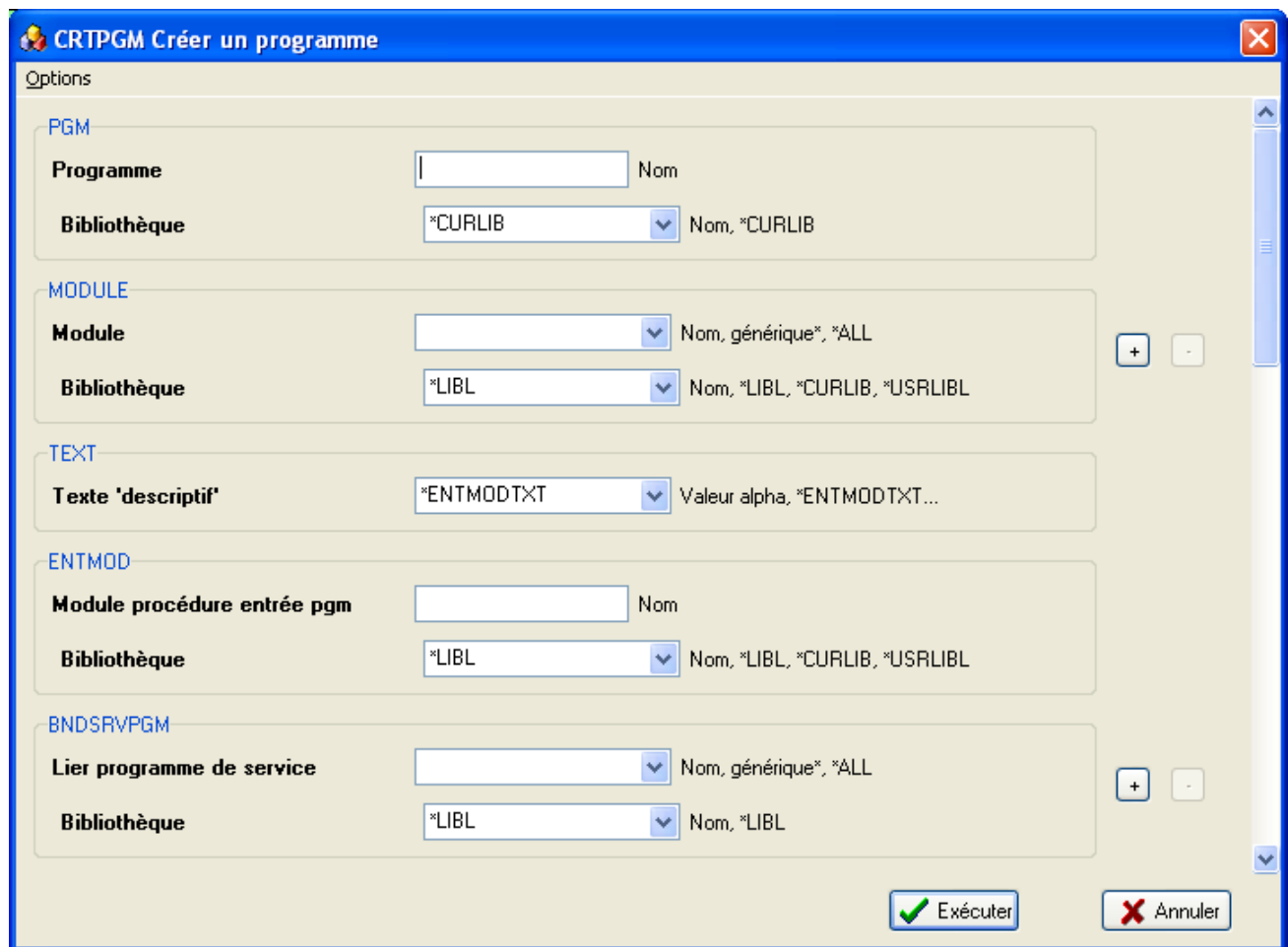
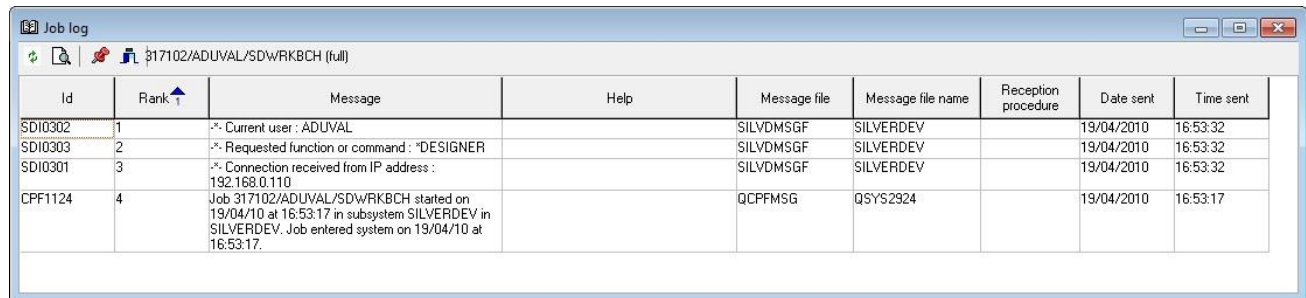


Figure 86

Job log

Use the menu Tools/job log




Id	Rank ↑	Message	Help	Message file	Message file name	Reception procedure	Date sent	Time sent
SDI0302	1	.* Current user : ADUVAL		SILVDMMSGF	SILVERDEV		19/04/2010	16:53:32
SDI0303	2	.* Requested function or command : *DESIGNER		SILVDMMSGF	SILVERDEV		19/04/2010	16:53:32
SDI0301	3	.* Connection received from IP address : 192.168.0.110		SILVDMMSGF	SILVERDEV		19/04/2010	16:53:32
CPF1124	4	Job 317102/ADUVAL/SDWRKBCH started on 19/04/10 at 16:53:17 in subsystem SILVERDEV. Job entered system on 19/04/10 at 16:53:17.		QCPFMMSG	QSYS2924		19/04/2010	16:53:17

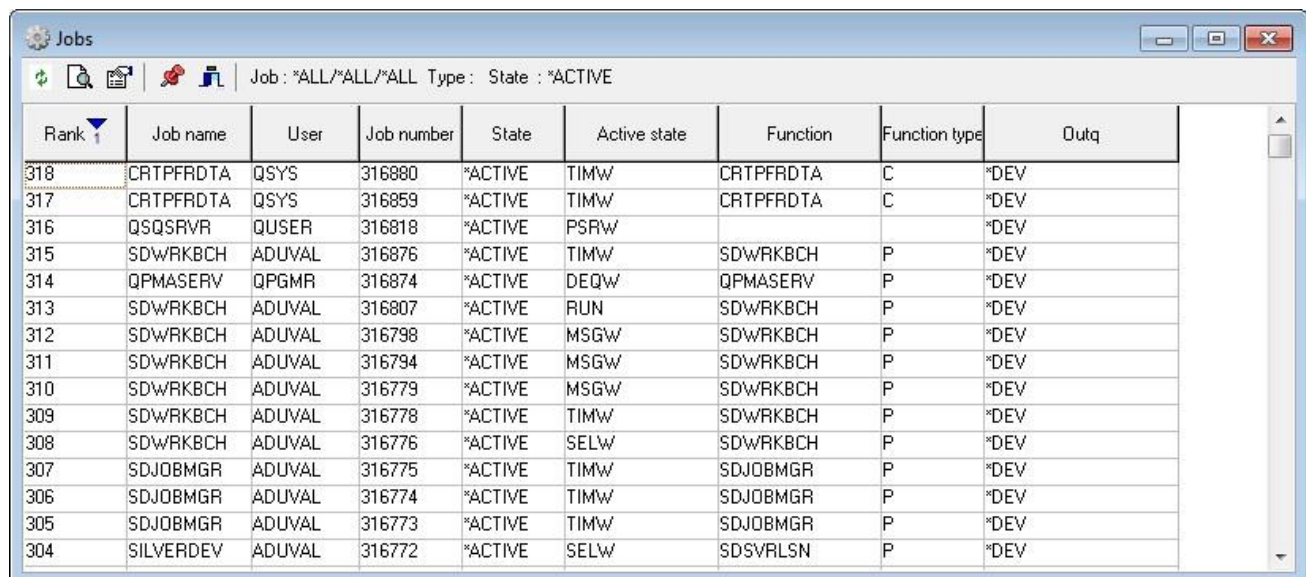
Figure 87

Job list

Use the menu Tools/Jobs to search for jobs.

Use the  button to modify the selection criteria.

Right click to select an action on a job.

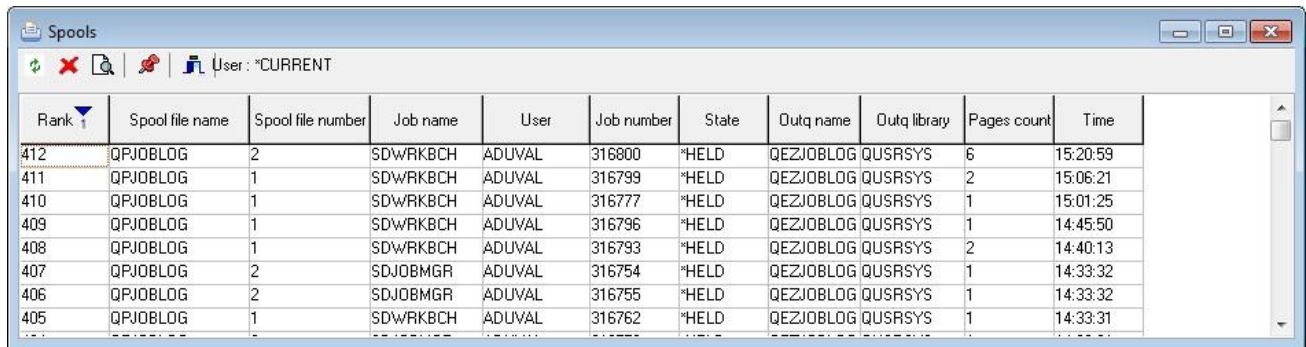


Rank ↓	Job name	User	Job number	State	Active state	Function	Function type	Outq
318	CRTPFRTA	QSYS	316880	*ACTIVE	TIMW	CRTPFRTA	C	*DEV
317	CRTPFRTA	QSYS	316859	*ACTIVE	TIMW	CRTPFRTA	C	*DEV
316	QSQRVR	QUSER	316818	*ACTIVE	PSRW			*DEV
315	SDWRKBCH	ADUVAL	316876	*ACTIVE	TIMW	SDWRKBCH	P	*DEV
314	QPMASERV	QPMGR	316874	*ACTIVE	DEQW	QPMASERV	P	*DEV
313	SDWRKBCH	ADUVAL	316807	*ACTIVE	RUN	SDWRKBCH	P	*DEV
312	SDWRKBCH	ADUVAL	316798	*ACTIVE	MSGW	SDWRKBCH	P	*DEV
311	SDWRKBCH	ADUVAL	316794	*ACTIVE	MSGW	SDWRKBCH	P	*DEV
310	SDWRKBCH	ADUVAL	316779	*ACTIVE	MSGW	SDWRKBCH	P	*DEV
309	SDWRKBCH	ADUVAL	316778	*ACTIVE	TIMW	SDWRKBCH	P	*DEV
308	SDWRKBCH	ADUVAL	316776	*ACTIVE	SELW	SDWRKBCH	P	*DEV
307	SDJOBMGR	ADUVAL	316775	*ACTIVE	TIMW	SDJOBMGR	P	*DEV
306	SDJOBMGR	ADUVAL	316774	*ACTIVE	TIMW	SDJOBMGR	P	*DEV
305	SDJOBMGR	ADUVAL	316773	*ACTIVE	TIMW	SDJOBMGR	P	*DEV
304	SILVERDEV	ADUVAL	316772	*ACTIVE	SELW	SDSVRLSN	P	*DEV

Figure 88

Spool list

Double-click a line to display the spool.




Rank	Spool file name	Spool file number	Job name	User	Job number	State	Outq name	Outq library	Pages count	Time
412	QPJOBLOG	2	SDWRKBCH	ADUVAL	316800	*HELD	QEZJOBLOG	QUSRSYS	6	15:20:59
411	QPJOBLOG	1	SDWRKBCH	ADUVAL	316799	*HELD	QEZJOBLOG	QUSRSYS	2	15:06:21
410	QPJOBLOG	1	SDWRKBCH	ADUVAL	316777	*HELD	QEZJOBLOG	QUSRSYS	1	15:01:25
409	QPJOBLOG	1	SDWRKBCH	ADUVAL	316796	*HELD	QEZJOBLOG	QUSRSYS	1	14:45:50
408	QPJOBLOG	1	SDWRKBCH	ADUVAL	316793	*HELD	QEZJOBLOG	QUSRSYS	2	14:40:13
407	QPJOBLOG	2	SDJOBMGR	ADUVAL	316754	*HELD	QEZJOBLOG	QUSRSYS	1	14:33:32
406	QPJOBLOG	2	SDJOBMGR	ADUVAL	316755	*HELD	QEZJOBLOG	QUSRSYS	1	14:33:32
405	QPJOBLOG	1	SDWRKBCH	ADUVAL	316762	*HELD	QEZJOBLOG	QUSRSYS	1	14:33:31


Figure 89

Library list

Modify the library list of the job associated with the designer via "Tools/Library list". This library list is necessary for program compilation, choosing a development context, etc.

System libraries are displayed in red and cannot be modified,
Production libraries are displayed in blue and cannot be modified.

The current library is displayed in green and can be modified using the  button.

User libraries are displayed in black can be modified using the  buttons.

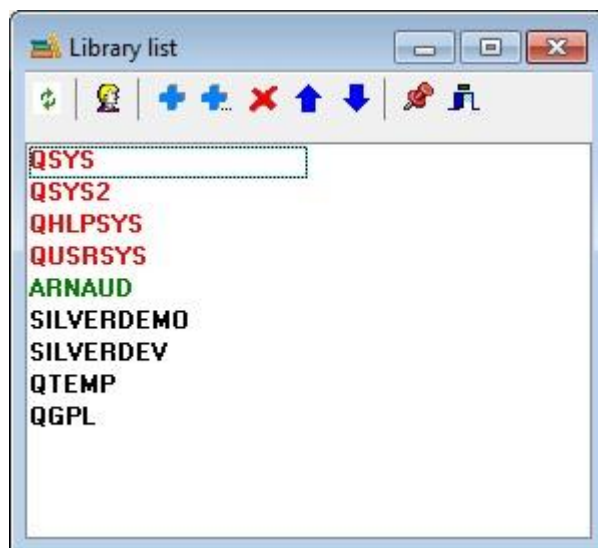


Figure 90

Comment:

Many functions require the SilverDev to be present in the library list.

Assistant

The "Tools/Assistant" menu enables display of the database assistant.

This tool displays the description of a data file on the System i and creation of controls from the file fields.

The controls created depend on the type of field and its properties depend on certain field attributes (name, length, description, type, possible values, etc.)

In the Assistant, a tree structure displays the list of libraries and the data files of the System i.

To develop a node, use the right click to display a scroll menu.

Double-click on the name of a file to see its description.

You can print this description via a right click.

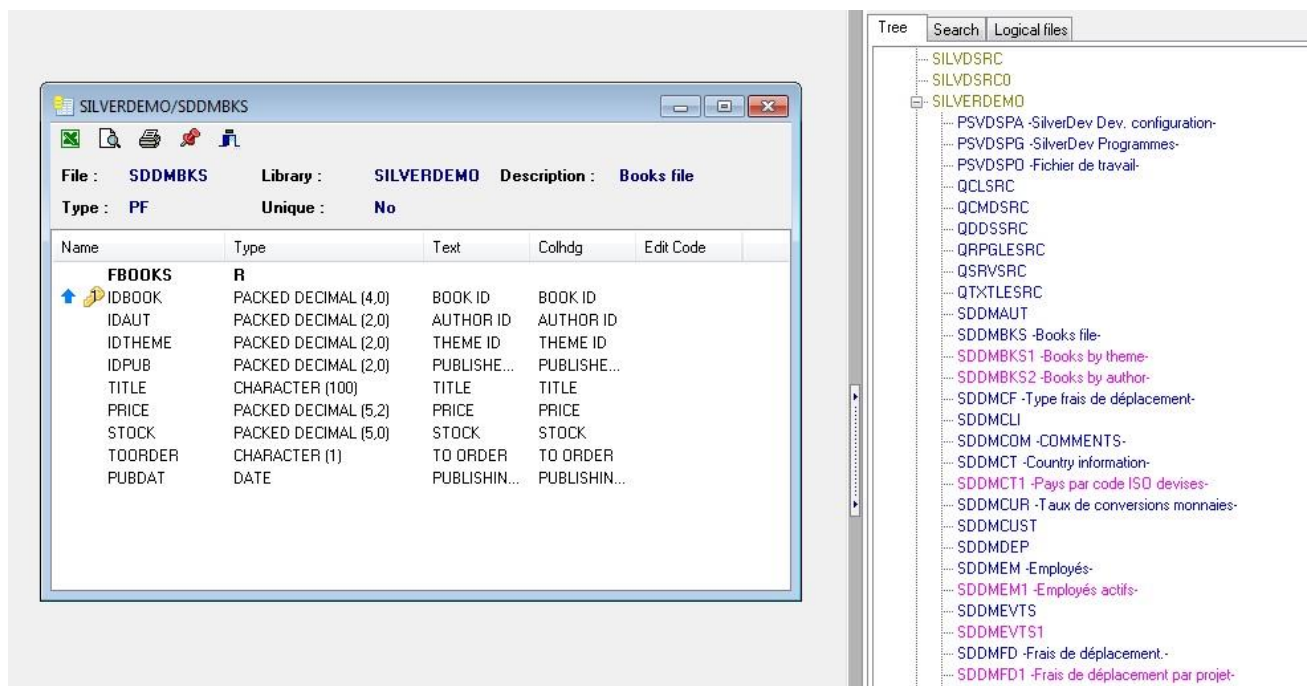
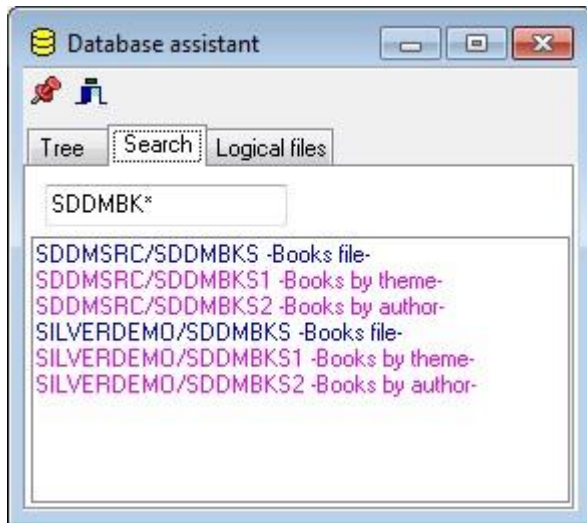


Figure 91

The second tab enables file searches. The generic character * can be used.

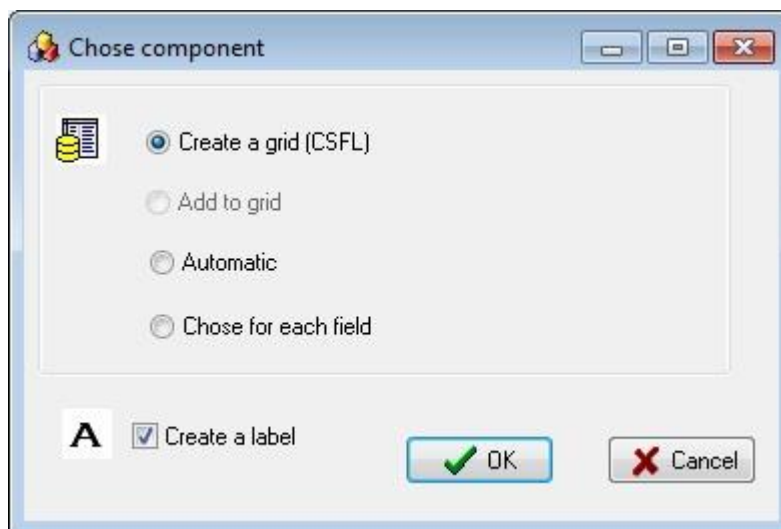
**Figure 92**

The physical files are displayed in blue, the logical files are in purple.

Display all the logical files depending on a physical file using the right click, "Logicals" menu.

Drag and drop

You can drag the fields from the description to create components with the properties corresponding to the field definition.

**Figure 93**

Search

In the bottom part of the window, you can enter a file name.

The list of all the corresponding files is displayed (you can enter a star at the end of the name).

Logical files

From this list, you can find the list of the logical files depending on the file with a left click.

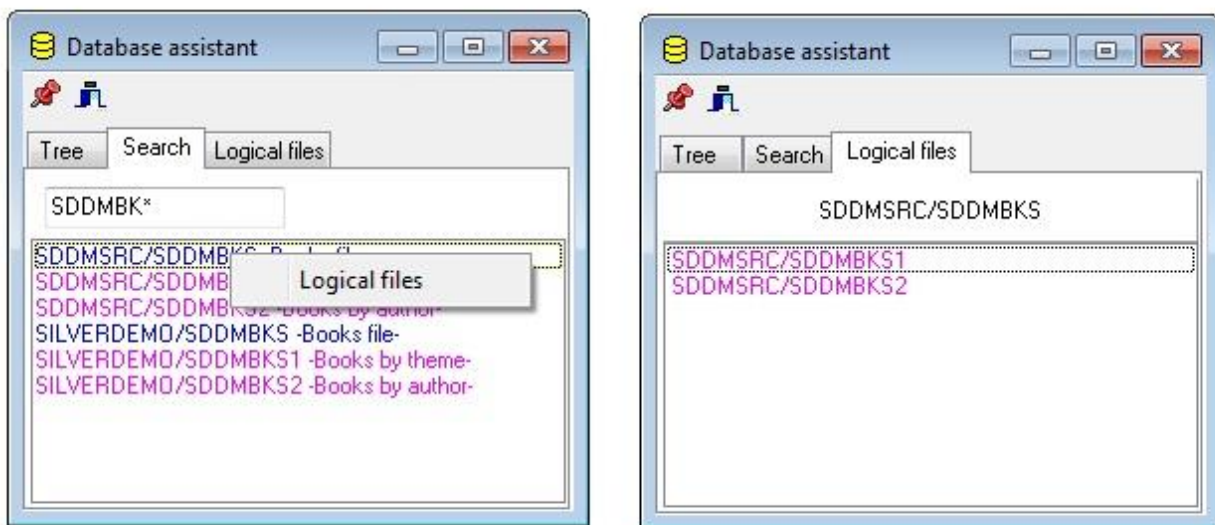


Figure 94

Database diagrams

The menu database diagrams allows to create database diagrams, displaying relations between files.

You can drop a table from database assistant. Fields are displayed.

Referential constraints between files are automatically displayed, so are links between physics and logical files.

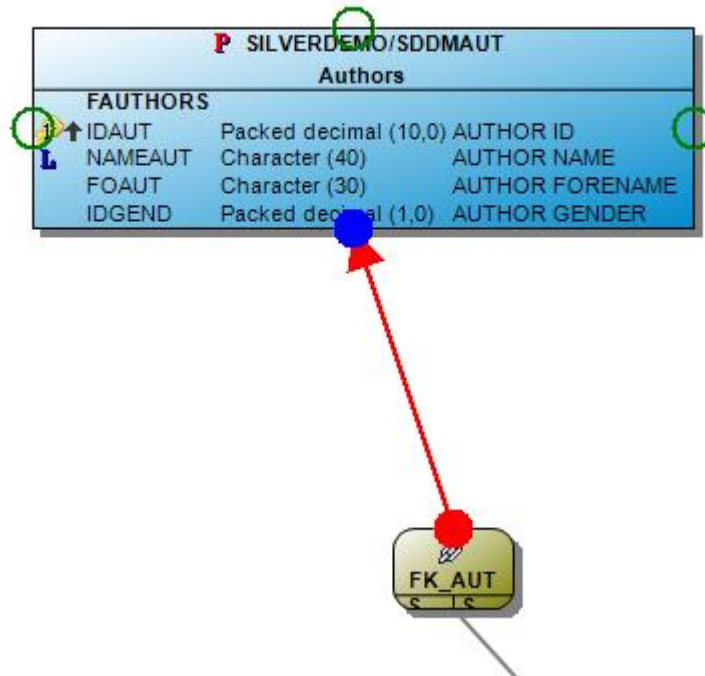
You can add manually relations between files.

The diagram allows to have a display of the database.

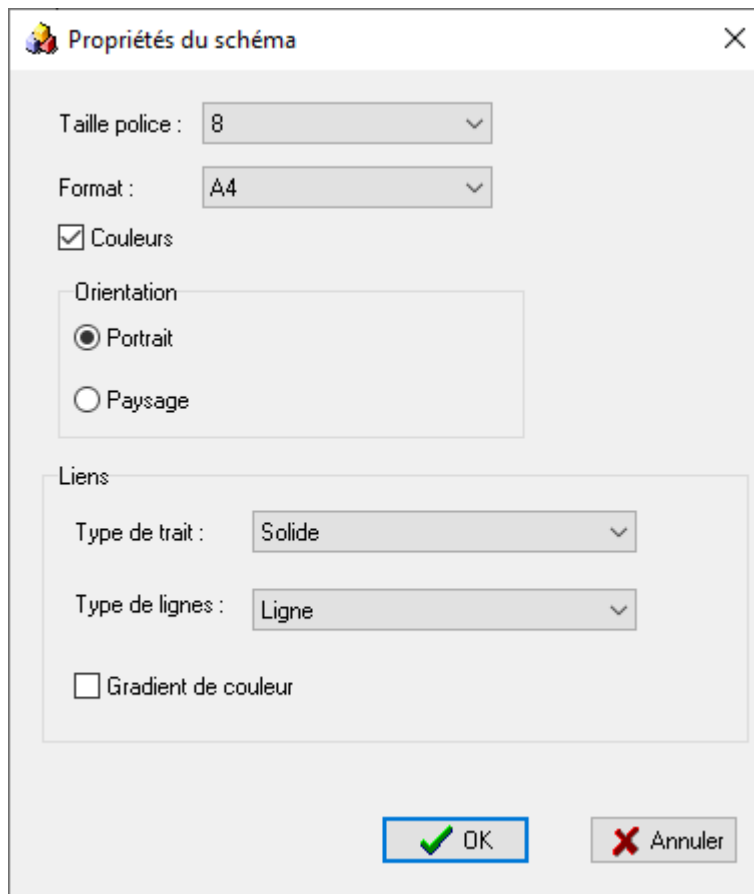
From the diagram, you can create applications automatically.

To do so, see RAD document chapter wizard.

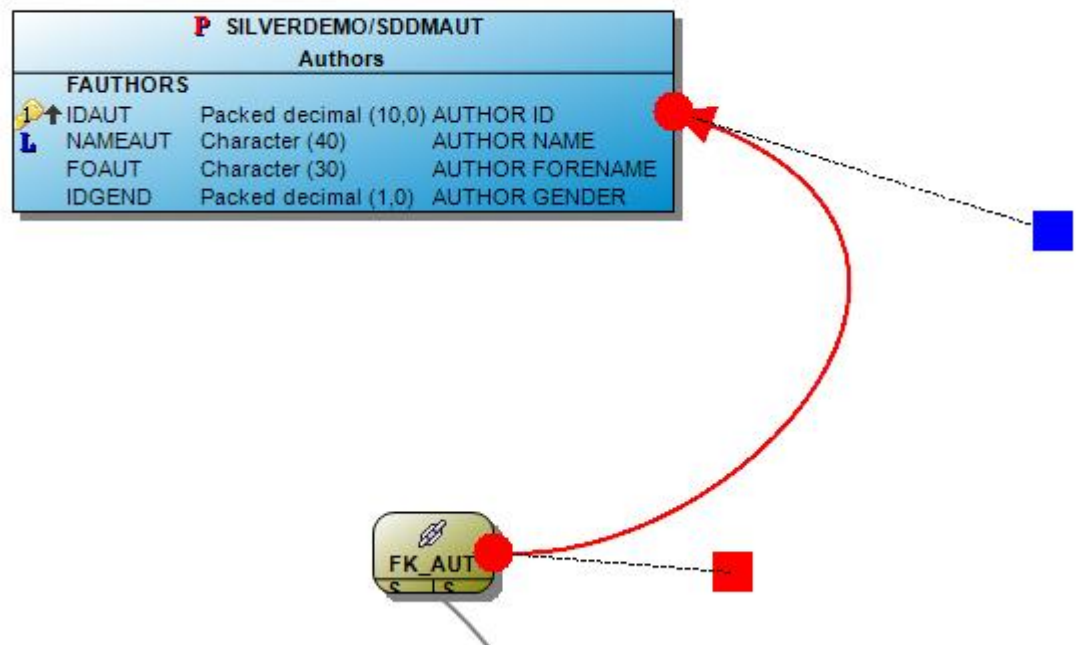
To change anchor points of a relation, select this relation and drag anchor points :



By right clicking on the diagram, you can change properties of the diagram :

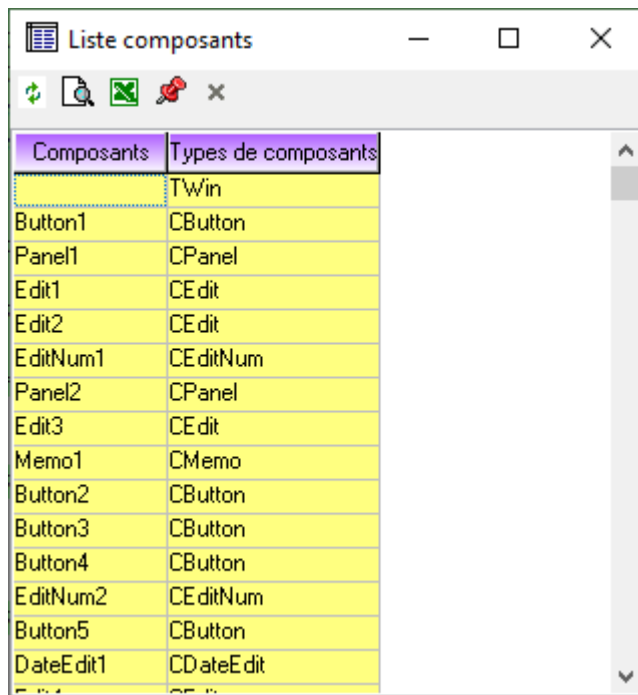


If you select curves for line type, you can select the relations and move control points :



Component list

Menu « tools/component list » allows to display component list of the current form.

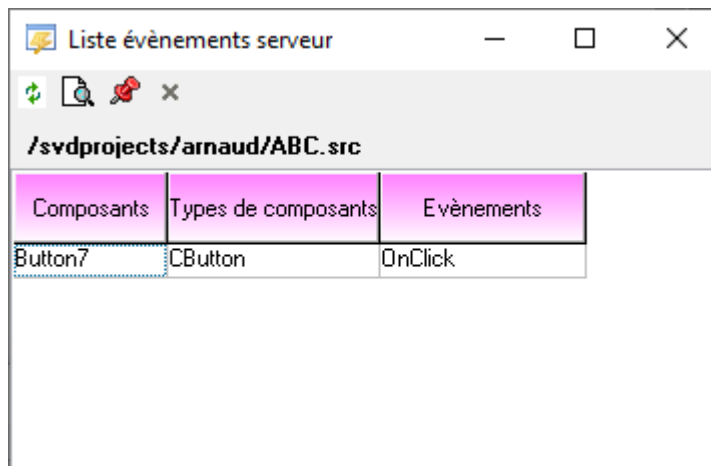


The screenshot shows a window titled 'Liste composants' with a toolbar containing icons for refresh, search, save, and close. Below the toolbar is a table with two columns: 'Composants' and 'Types de composants'. The table lists various UI components and their corresponding classes.

Composants	Types de composants
	TWin
Button1	CButton
Panel1	CPanel
Edit1	CEdit
Edit2	CEdit
EditNum1	CEditNum
Panel2	CPanel
Edit3	CEdit
Memo1	CMemo
Button2	CButton
Button3	CButton
Button4	CButton
EditNum2	CEditNum
Button5	CButton
DateEdit1	CDateEdit

Server event list

Menu « tools/server event list » allows to display the server event list of the current form

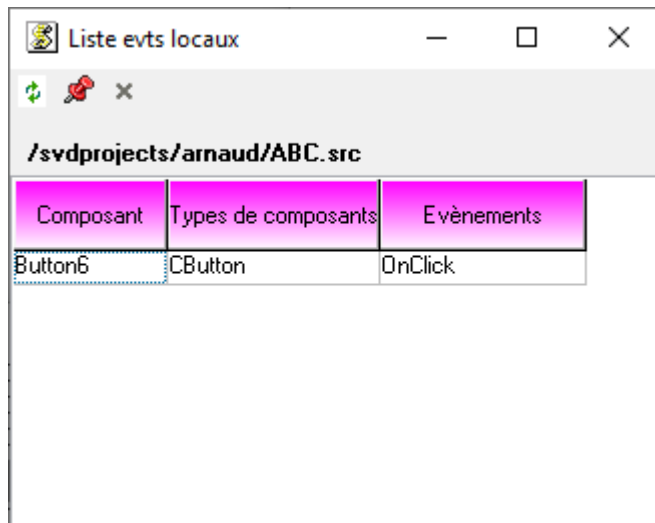


The screenshot shows a window titled 'Liste évènements serveur' with a toolbar containing icons for refresh, search, save, and close. Below the toolbar is a text field showing the path '/svdprojects/arnaud/ABC.src'. Below this is a table with three columns: 'Composants', 'Types de composants', and 'Evènements'. The table lists the server events for a specific component.

Composants	Types de composants	Evènements
Button7	CButton	OnClick

Local event list

Menu « tools/local event list » allows to display local event list of the current form.



Tree structure

The "Tools/Tree structure" menu enables display of the current form in tree structure form. It is then possible to move the controls from one parent to another. This is an alternative to the cut/paste solution to change a control's parent. The advantage of this solution is that the events are not lost.

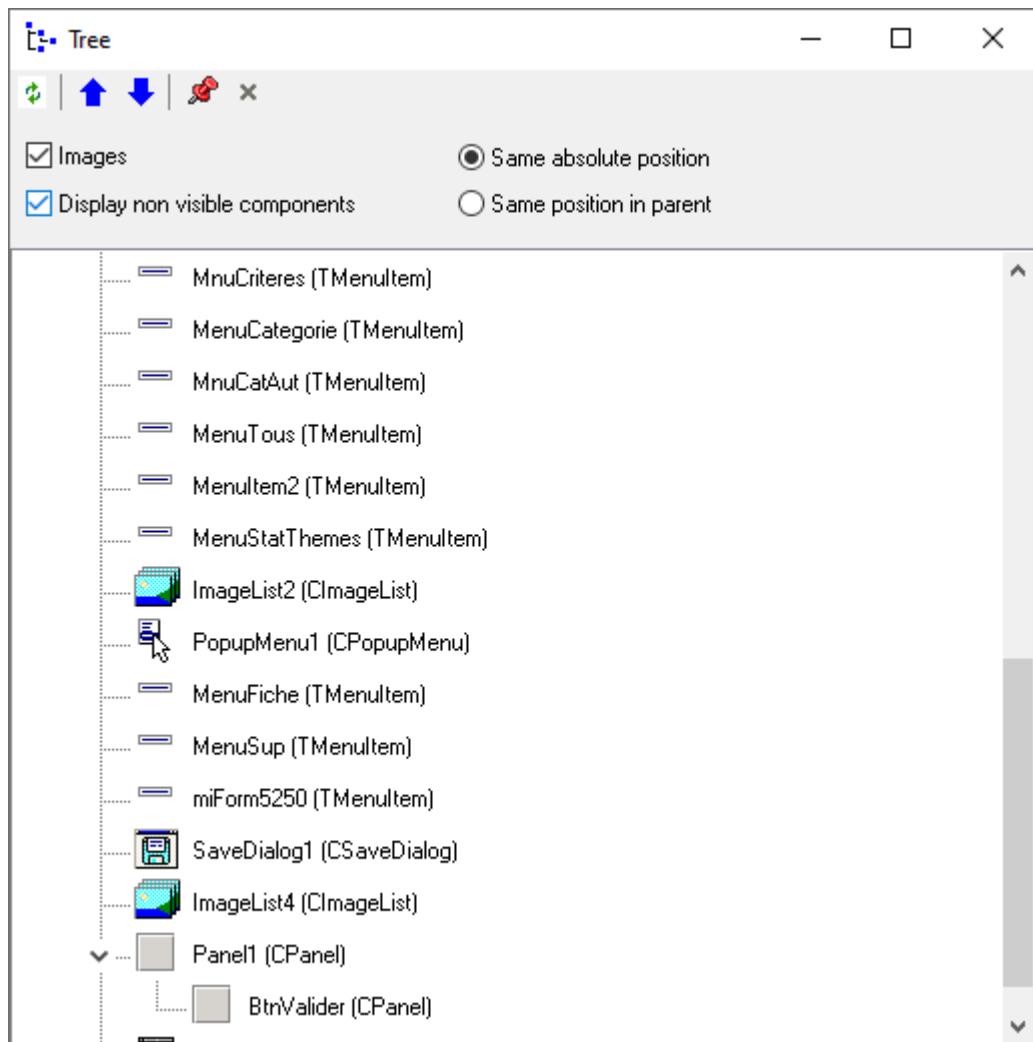


Figure 95

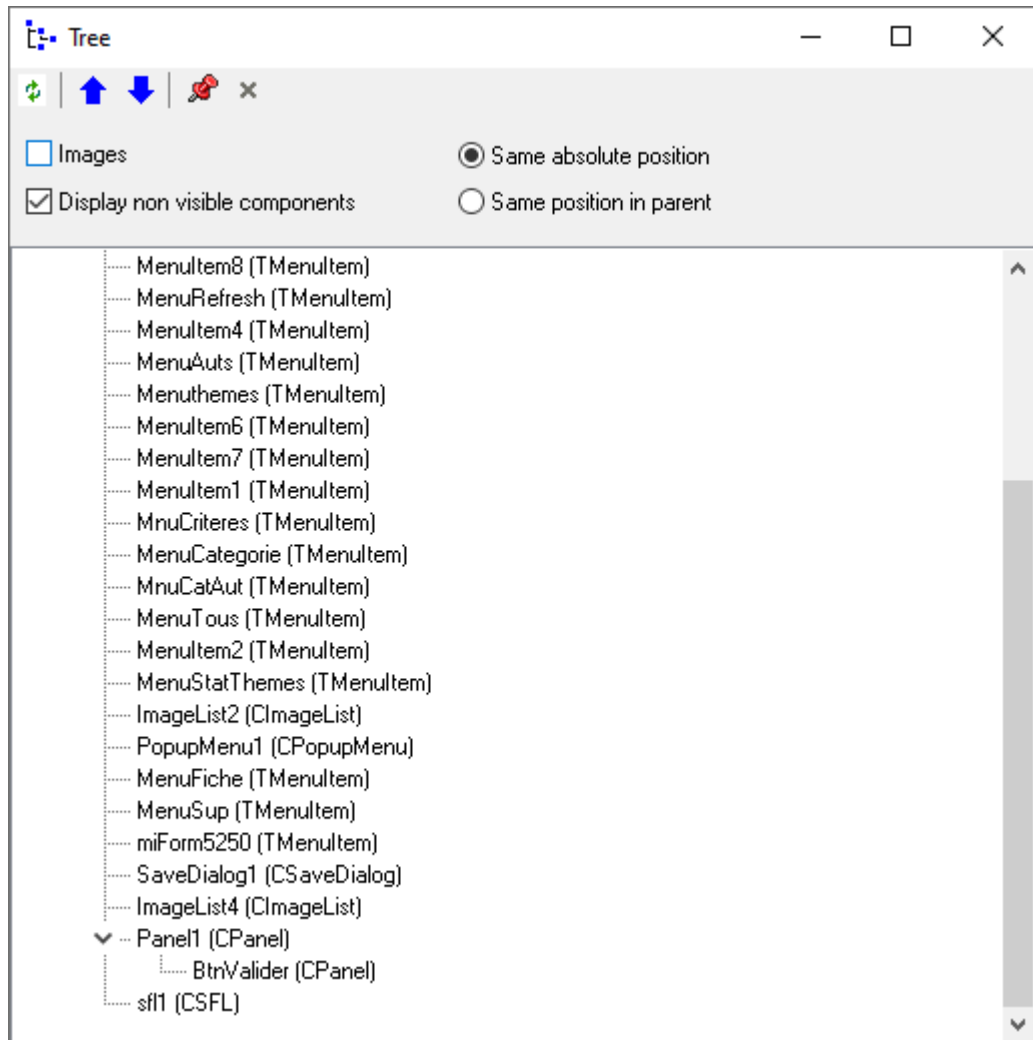


Figure 96

The images tick box determines display of the icon representing the component. Without the images, more components can be displayed in the window.

Radio buttons :

When “same absolute position” is checked, the position relative to the top left corner of the window does not change.

When “Same position in parent” is checked, the position relative to the top left corner of the new parent is the same as the one of the previous parent.

Decompilation

If you lose a source but you still have the screen object, you can decompile the screen using the “Tools/Decompile” menu.

Explorer

Use the explorer to open a member.

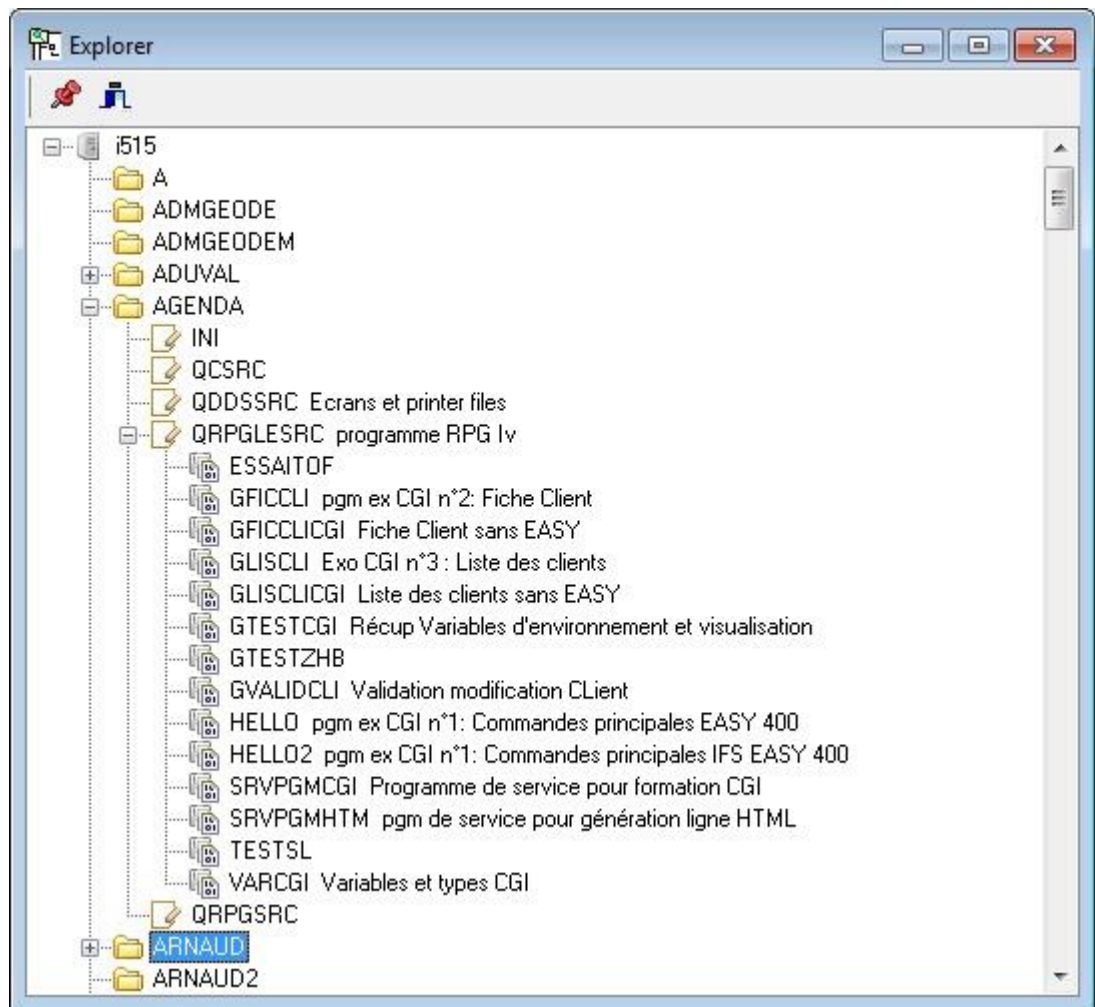


Figure 97

Versionning

See chapter on versionning in administration document

Compare sources

Source comparison tool allows to detects additions, deletings and modifications from a source to another. You can compare sources coming from a member, an ifs file or from history(versionning)

Compare contents

Contents comparison tool allows to compare two libraries , (objects added or deleted), directories (files and sub directories added or deleted) , or source files (member added or deleted)

Chapter 16. Common mistakes

Mistake	Symptom
<p>The name of the event manager function is misspelled in the sdSetCallBack function.</p> <p><u>Note:</u> The name must be entered in CAPITALS.</p>	<p>The program does not compile.</p> <p>Compiler message: "Errors were detected during binding."</p>
<p>The PForm parameter has been declared without the Value key word in the event manager function.</p>	<p>The program crashes during the call to an event manager with the message: "Pointer not defined for referenced memory position"</p>
<p>The name of a component or a property is misspelled.</p>	<p>The error is not detected during compilation, but an error occurs during execution.</p> <p>To identify this type of error, use the Debug tool.</p>
<p>A function call has been passed with a form as a parameter, but the form has not yet been created by the sdCreateForm function.</p>	<p>The application crashes with the message "Form 0 doesn't exist"</p>
<p>A compiled SilverDev program in the *NEW activation group</p>	<p>Crash with error message MCH3402.</p> <p>Access attempt to any object or part of an object that no longer exists.</p>
<p>The "visible" property of the window is false. This is the default value. <u>Only the first window</u> displayed is automatically modified to "visible=true" on creation.</p>	<p>A "secondary" window is not displayed after a call to sdCreateForm. The first window is displayed correctly.</p>
<p>Loss of a screen source.</p> <p>We have a screen decompiler.</p> <p>Please contact us to retrieve your screens.</p>	
<p>Excel corrupted files</p>	<p>The sdExport function generates files with the excel 2.0 format so that any excel program can open them. Since microsoft office 2010 a message "file corrupted" may appear.</p> <p>Go to "Files/options/Trust center/Trust Center Settings/File Block Settings</p> <p>Unchecke all checkboxes in the "open" column.</p>

Chapter 17. Solutions replaced

SilverDev's upgrades result in certain functions, components or techniques being replaced by others. For compatibility reasons, the old solutions are left in place and it is useful to be aware of them.

CStringGrid component

The CStringGrid component has been replaced by the CSFL component with a wider range of possibilities.

SdSelectColor, sdSelectFile, sdPrintDlg

The sdSelectColor function has been replaced by the use of the CColorDialog component and the sdDialog function. Use of the CColorDialog component enables modification of the appearance of the dialogue box (personalised colours, larger box, etc.)

Similarly, the sdSelectFile and sdPrintDlg functions can be replaced by the sdDialog function associated with the COpenDialog and CPrintdialog components.

SdSetSet, sdAddSet, sdDelSet

These functions have been replaced by the sdGetSest and sdUpdSet functions which handle a string of 32 characters.

sdAddNode

The sdAddNode function has been replaced by the sdAddNode2 function for faster loading of the treeview with a larger number of nodes.

Furthermore, the `sdAddNode2` function is easier to use since it does not need a counter.

Chapter 18. Disconnections

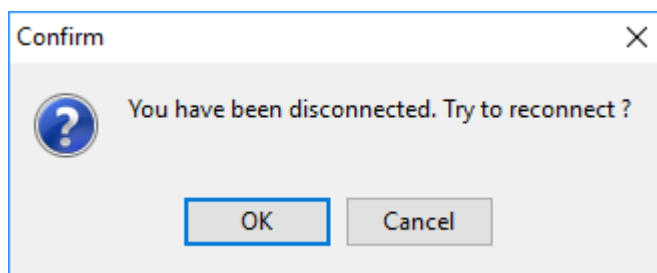
Client part may be disconnected from server part.

This can happen due to a bad network, a router, a proxy, a firewall...

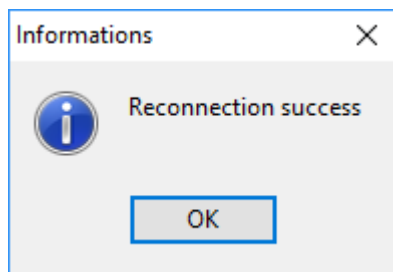
For Silverdev applications, the server job waits for a reconnection.

On a user event, client part detects that it has been disconnected.

The client part prompt for a reconnection.



If reconnection succeeds, the user can resume its work.



The client part is then connected on another port than the one set for the server.

Ports used for reconnection are the 500 next to the one set for the server.

By default, the port used for silverdev is 7003. Ports 7004 to 7503 are used for reconnection.

Chapter 19. Advanced technicals

The advanced technicals document contains following chapters :

Functions of List category**Optimisation****SFL : Copying data locally****Local events****Multi form applications****Mdi applications****Multi forms applications with tabs****Multi form occurrences****Multi form occurrences, sdsrvlst service program****Dialog box functions****PC file category functions****Collections category functions****TStrings category functions****Set category functions****Miscellaneous functions****Tips****Web services****Screen translation****Image category functions****List of forms and events****Colors****Anchoring components****Drag and drop operations****Screen size adaptation****ADO**

Exit points

Interactions with external programs

Sdsrvjson service program

WebBrowser