STZHLP – STEEZWARE HELP

USER MANUAL

VERSION 2020-04
20 APRIL 2020 BY SCOTT MCDONELL

Contents

Overview

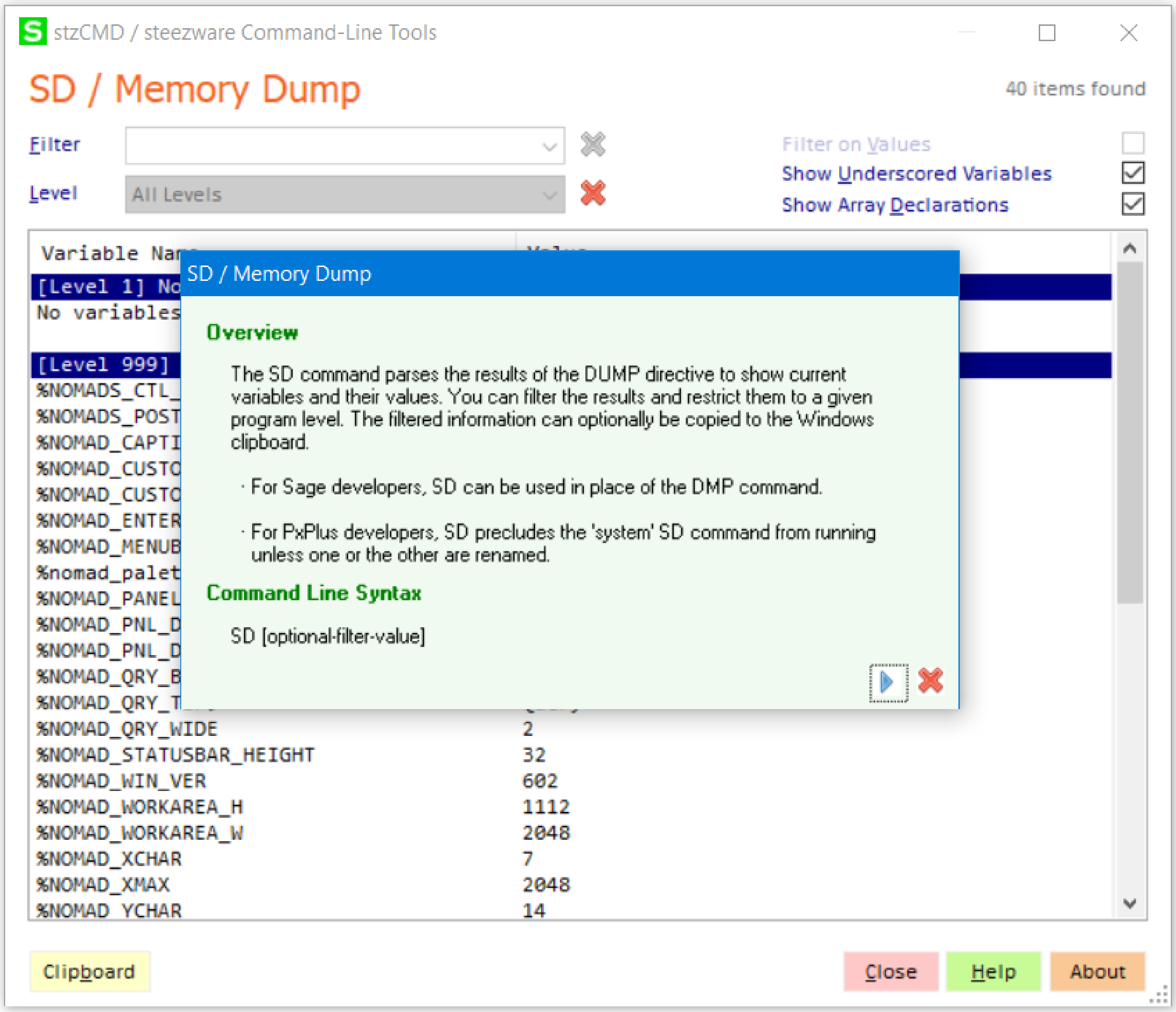
stzHLP – *pronounced steeze-Help* – provides control-positioned Nomads-based help windows for ProvideX and PxPlus applications. Rather than a help panel being limited to display centered on the entire screen or relative to the location of a parent window, stzHLP automatically positions help windows relative to a specified control on the parent window.

The real power of stzHLP is that its help panels are always positioned based on the parent window current size and location, so if the parent window is moved or resized stzHLP uses its current coordinates to position the help panels.

Under program control, each help panel is attached to a control on the parent window at the panel top-left, top-right, bottom-left, bottom-right corners - or centered over the parent window.

Example

Using the stzCMD SD – Memory Dump tool as an example, when help is invoked an initial ‘overview’ help panel displays centered on the window. Since this is the first help panel, the only buttons it has are [Next] and [Exit].



```
Variable Na
[Level 1] No
No variables

[Level 999]
%NOMADS_CTL_
%NOMADS_POST
%NOMAD_CAPTI
%NOMAD_CUSTO
%NOMAD_CUSTO
%NOMAD_ENTER
%NOMAD_MENUB
%nomad_palet
%NOMAD_PANEL
%NOMAD_PNL_D
%NOMAD_PNL_D
%NOMAD_QRY_B
%NOMAD_QRY_T
%NOMAD_QRY_WIDE              2
%NOMAD_STATUSBAR_HEIGHT      32
%NOMAD_WIN_VER               602
%NOMAD_WORKAREA_H            1112
%NOMAD_WORKAREA_W            2048
%NOMAD_XCHAR                 7
%NOMAD_XMAX                  2048
%NOMAD_YCHAR                 14
```

SD / Memory Dump

Overview

The SD command parses the results of the DUMP directive to show current variables and their values. You can filter the results and restrict them to a given program level. The filtered information can optionally be copied to the Windows clipboard.

· For Sage developers, SD can be used in place of the DMP command.

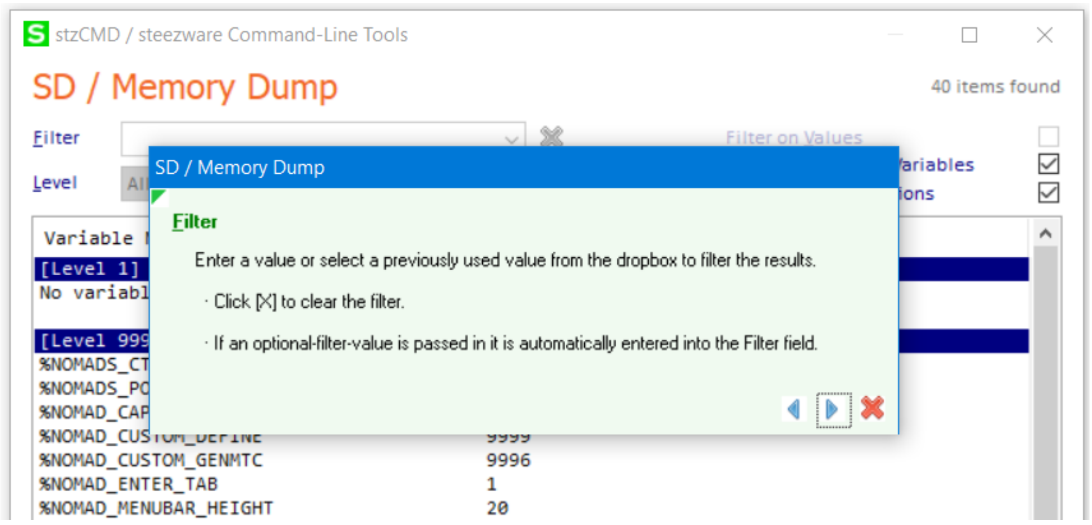
· For PxPlus developers, SD precludes the 'system' SD command from running unless one or the other are renamed.

Command Line Syntax

SD [optional-filter-value]

Clicking [Next] brings up the next help panel, positioned over the Filter control since it is defined in code to have its top-left corner over that control. Notice this panel has a [Previous] button displayed since it is the second help panel. The stzHLP class displays or hides the prev/next buttons based on if there is a preceding or following panel.

Also, note the green pointer at the top-left of the help panel, to direct the user's attention to the control for which the help applies.



SD / Memory Dump

Filter

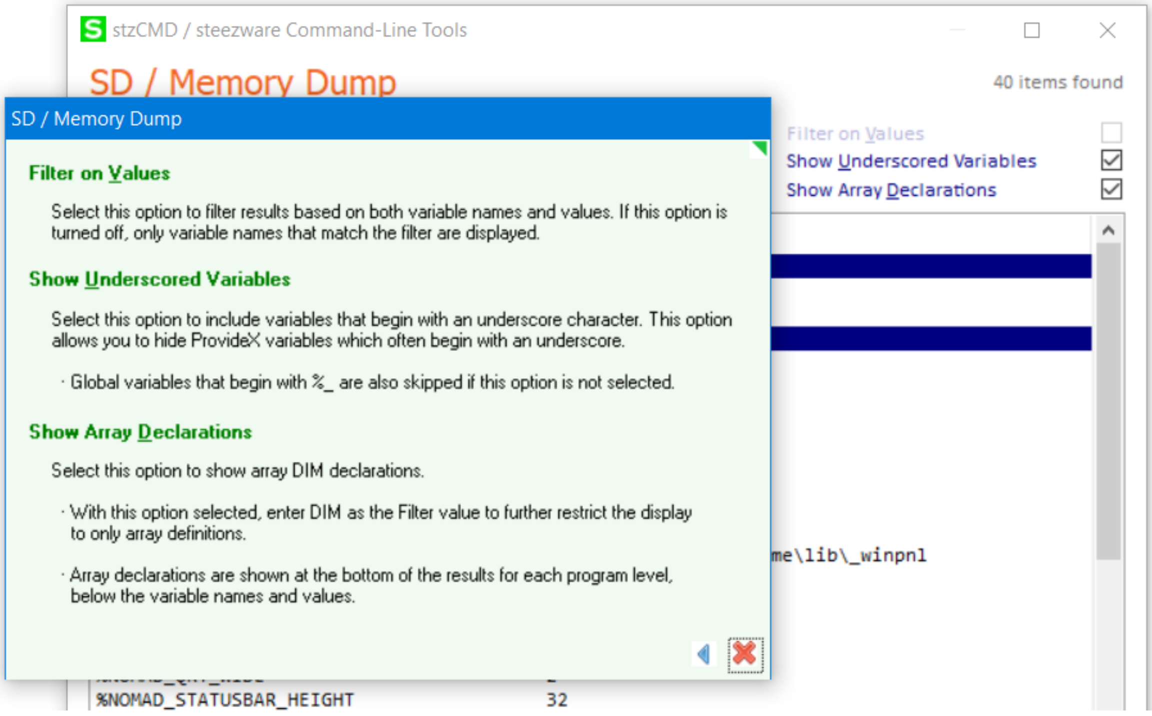
Enter a value or select a previously used value from the dropbox to filter the results.

· Click [X] to clear the filter.

· If an optional-filter-value is passed in it is automatically entered into the Filter field.

```
Variable
[Level 1]
No variabl
[Level 999
%NOMADS_CT
%NOMADS_PO
%NOMAD_CAP
%NOMAD_CUSTOM_DEFINE          9999
%NOMAD_CUSTOM_GENMTC          9996
%NOMAD_ENTER_TAB              1
%NOMAD_MENUBAR_HEIGHT         20
```

The final panel is defined to be located with its top-right cornier relative to the Filter-on-Values checkbox. In this panel, a green pointer at the top-right of the help panel draws attention to the options at the upper-right area of the screen.



SD / Memory Dump

Filter on Values

Select this option to filter results based on both variable names and values. If this option is turned off, only variable names that match the filter are displayed.

Show Underscored Variables

Select this option to include variables that begin with an underscore character. This option allows you to hide ProvideX variables which often begin with an underscore.

· Global variables that begin with %_ are also skipped if this option is not selected.

Show Array Declarations

Select this option to show array DIM declarations.

· With this option selected, enter DIM as the Filter value to further restrict the display to only array definitions.

· Array declarations are shown at the bottom of the results for each program level, below the variable names and values.

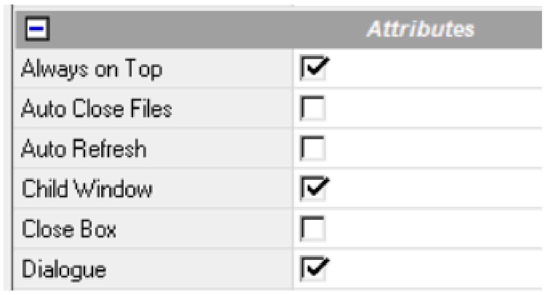
```
me\lib\_winpnl
%NOMAD_STATUSBAR_HEIGHT       32
```

Remember, if the SD window is moved or resized its stzHLP panels will position themselves relative to the control to which they are associated. However, even if you set the help panels as always-on-top it may be possible to move or resize the underlying parent window while help is showing, and any subsequent help panels may not be correctly located, until help is closed and re-opened, that is.

How it Works

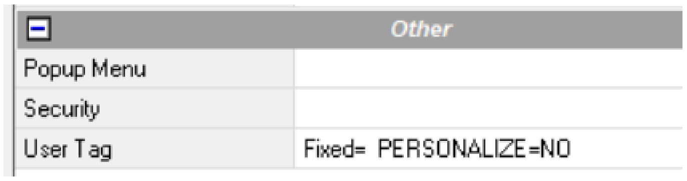
Nomads

Help panels are created using Nomads, with the only requirement being the inclusion of previous, next and cancel buttons that follow stzHLP specs. The panels should be defined as “Always on Top”, “Child Window” and “Dialogue”…



Attributes	
Always on Top	✓
Auto Close Files	☐
Auto Refresh	☐
Child Window	✓
Close Box	☐
Dialogue	✓

For Sage 100 use, assign “PERSONALIZE=NO” into the User Tag…



Other	
Popup Menu	
Security	
User Tag	Fixed= PERSONALIZE=NO

The SD examples above use a green background, dark green headings, special fonts, and pointer images in the appropriate corners … but your panels can be defined in any desired format. stzCMD help panels also include a title bar – so the window can be moved if desired – but even the title bar is optional.

Also noteworthy is that stzCMD uses a separate Nomads library for its help panels, although this is not required – place your help panels in any library you wish.

Code

Specific code must be added to the program or UI class which processes the help button. In summary, this code must follow these steps…

1. Instantiate the stzHLP class.
2. Set a snapshot of the current position and size of the parent window into properties of the stzHLP class.
    - This ensures that stzHLP has the correct coordinates for locating each help panel based on the parent window current position and size.
3. Information about each help panel and the control to which it is attached is set into stzHLP using the ‘AddPanel method.
4. The ‘Go method is called and the help panels are displayed in the sequence and locations which were specified.
5. Drop the stzHLP class.

Complete details on designing the Nomads help panels and implementing the required code is covered later in this document.

Installation

The only stzHLP file is a class named _STZHLP.PVC, found in the zip file downloaded from steezware.com.

It must be saved into the LIB folder, that is, the same folder as the _NOMADS.PVC class.

For Sage 100, save the program into HOME\LIB.

Support

Use the contact form on the steezware website to request support, report issues or make suggestions.

Nomads Help Panels

There are only three required controls that must be placed on each help dialog, for the previous, next and cancel buttons. Their names must be…

stzPrev
stzNext
stzCancel

Except for their names, there are no design constraints placed on the buttons – they can have an image or text or both, their colors, fonts and position on the panel do not matter, etc. The stzCMD help panels set the tab stops to stzPrev > stzNext > stzCancel, but this is also optional.

The size and colors and fonts used by each panel are up to you, the only exceptions being that it is recommended that the header of each panel be defined with…

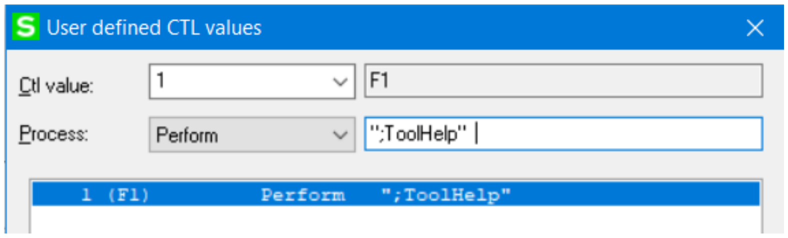
Always on Top
Child Window
Dialogue
User Tag: PERSONALIZE=NO (recommended only for Sage 100 use)

A panel can be defined as resizable, can include a title bar or a close box, or any other attribute you want to use.

Hint To get to know and understand stzHLP panel design, install stzCMD from the steezware website and open the *CMD\stzCMD\stzCMDHLP.SW2 library in Nomads.

Nomads Parent Panel Help Button

Many of the stzCMD windows have a [Help] button that is defined to perform logic under the TOOLHELP line label. In addition, the [F1] key is set in user-defined CTL values to perform the code found under TOOLHELP…



Ctl value: 1 F1

Process: Perform ";ToolHelp"

```
1 (F1)          Perform    ";ToolHelp"
```

Your application must include logic that instantiates and runs stzHLP, so setting your [Help] button or [F1] key must be defined to execute, that is PERFORM, that logic.

Code

This sample code from the stzCMD **SD** tool shows how the stzHLP class is instantiated and the properties that must be passed into it…

```
TOOLHELP:

   LOCAL stzOBJ, tmpCTL

   stzOBJ = NEW("*stzHLP", %stzCMD$, %stzHeading$, %stzInterpreter$)

   stzOBJ'NomadsLib$ = "*CMD\stzCMD\stzCMDHLP.SW2"

   stzOBJ'WindowCol  = INT(MAX(0,(DEC($00$+MID(OBJ(0),21,2))-%NOMAD_WORKAREA_X))/%NOMAD_XCHAR)  ! column offset of
                                                                                                 this window
   stzOBJ'WindowRow  = INT(MAX(0,(DEC($00$+MID(OBJ(0),23,2))-%NOMAD_WORKAREA_Y))/%NOMAD_YCHAR)  ! row offset of
                                                                                                 this window
   stzOBJ'WindowCols = DEC($00$+MID(OBJ(0),33,2))                                    ! total columns of this window
   stzOBJ'WindowRows = DEC($00$+MID(OBJ(0),35,2))                                    ! total rows of this window

   stzOBJ'AddPanel("hlp"+SCRN_ID$+"-00)                                              ! overview help dialog

   tmpCTL = DB_Filter.ctl
   stzOBJ'AddPanel("hlp"+SCRN_ID$+"-01", tmpCTL'Col, tmpCTL'Line, tmpCTL'Cols, tmpCTL'Lines, "topLeft")
   tmpCTL = DB_Level.ctl
   stzOBJ'AddPanel("hlp"+SCRN_ID$+"-02", tmpCTL'Col, tmpCTL'Line, tmpCTL'Cols, tmpCTL'Lines, "topLeft")
   tmpCTL = CB_FilterValues.ctl
   stzOBJ'AddPanel("hlp"+SCRN_ID$+"-03", tmpCTL'Col, tmpCTL'Line, tmpCTL'Cols, tmpCTL'Lines, "topRight")

   stzOBJ'Go()

   DROP OBJECT stzOBJ

RETURN
```

Breaking it Down

As explained earlier, the TOOLHELP label is used by the stzCMD tools to invoke help. You can use whatever label or method call you want to instantiate and run stzHLP.

Since stzCMD PERFORMs this logic, a couple variables are localized in the first line. Required logic then follows…

1. First, instantiate the *stzHLP class…

   ```
   stzOBJ = NEW("*stzHLP", %stzCMD$, %stzHeading$, %stzInterpreter$)
   ```

   Up to 10 arguments can be passed into the class, which will be available to your help panels using the ARG_1$, ARG_2$ … ARG_10$ Nomads variables. In the example above, variables containing “stzCMD”, “SD / Memory Dump” and the interpreter description are passed in. These three ARG_1$, ARG_2$ and ARG_3$ Nomads variables are used in help panel header titles and fonted-text field expressions.

   Note Arguments ARG_11$ through ARG_18$ are used internally by stzHLP and are not available for use in help panels.

2. After instantiating the class, the name of the Nomads library containing the help panels is set into the NomadsLib$ property…

   ```
   stzOBJ'NomadsLib$ = "*CMD\stzCMD\stzCMDHLP.SW2"
   ```

3. The following four lines set into the object the position and size of the “parent” window, that is, the current Nomads window for which help is being invoked. Setting these variables is what enables stzHLP to position the help panels in relation to controls on the panel.

   Note that these lines must be entered into your code exactly as shown below.

   ```
   stzOBJ'WindowCol  = INT(MAX(0,(DEC($00$+MID(OBJ(0),21,2))-%NOMAD_WORKAREA_X))/%NOMAD_XCHAR)
   stzOBJ'WindowRow  = INT(MAX(0,(DEC($00$+MID(OBJ(0),23,2))-%NOMAD_WORKAREA_Y))/%NOMAD_YCHAR)
   stzOBJ'WindowCols = DEC($00$+MID(OBJ(0),33,2))
   stzOBJ'WindowRows = DEC($00$+MID(OBJ(0),35,2))
   ```

4. Next, each of the help panels is associated with a control from the parent window using the ‘AddPanel() method…

   ```
   stzOBJ'AddPanel("hlp"+SCRN_ID$+"-00")
   tmpCTL = DB_Filter.ctl
   stzOBJ'AddPanel("hlp"+SCRN_ID$+"-01", tmpCTL'Col, tmpCTL'Line, tmpCTL'Cols, tmpCTL'Lines,
                   "topLeft")
   tmpCTL = DB_Level.ctl
   stzOBJ'AddPanel("hlp"+SCRN_ID$+"-02", tmpCTL'Col, tmpCTL'Line, tmpCTL'Cols, tmpCTL'Lines,
                   "topLeft")
   tmpCTL = CB_FilterValues.ctl
   stzOBJ'AddPanel("hlp"+SCRN_ID$+"-03", tmpCTL'Col, tmpCTL'Line, tmpCTL'Cols, tmpCTL'Lines,
                   "topRight")
   ```

   Syntax

   ```
   ‘AddPanel ( screenID$ )
   ‘AddPanel ( screenID$, controlColumn, controlLine, controlWidth, controlHeight,
                 helpPanelPosition )
   ‘AddPanel ( screenID$, controlColumn, controlLine, controlWidth, controlHeight,
                 helpPanelPosition, ofsColumn, ofsRow )
   ```

   Note ‘AddPanel can be invoked with a single argument, 6 arguments or 8 arguments.

   Where **screenId$** The name of the help panel. In the SD tool, the help panels are named “hlpSD-00” through “hlpSD-03”, so the SCRN_ID$ variable (which contains “SD”) is used to specify the help panel.

If a help panel is not attached to a control, calling ‘AddPanel with only the screenID$ argument centers the help panel on the parent window. For example, see the first stzOBJ’AddPanel call above.

Note In the above sample code the variable tmpCTL is used to identify each control for setting the four following values, but explicitly using control variables such as DB_Filter.ctl’Col, DB_Filter.ctl’Line, DB_Filter.ctl’Cols and DB_Filter.ctl’Lines in an ‘AddPanel call is fine.

controlColumn	The column location of the control to which the help panel is attached.
controlLine	The line where the control is located on the parent window.
controlWidth	The control width, in columns.
controlHeight	The control height, in rows.
helpPanelPosition	Tells stzHLP which corner of the help panel should be anchored to the control. Note that these five options are not case-sensitive. topLeft topRight bottomLeft bottomRight center
ofsColumn	The rules stzHLP uses for horizontal positioning a help panel relative to a control on the parent window can be tweaked with this optional argument. For example, for “topLeft” the help panel column is calculated as the column position of the parent control plus 4. By specifying a positive *or* negative value in this argument you can adjust to the left or right where the help panel gets located.
ofsRow	The rules stzHLP uses for vertical positioning a help panel relative to a control on the parent window can be tweaked with this optional argument. For example, for “topLeft” the help panel row is calculated as the row position of the parent control plus 4. By specifying a positive *or* negative value in this argument you can adjust up/down the help panel position.

5. After setting the required object properties and making the necessary ‘AddPanel calls, use the ‘Go() method to pass control to the object to display each of the help panels…

```
stzOBJ'Go()
```

6. Finally, drop the object upon its close…

```
DROP OBJECT stzOBJ
```

Note You **must** drop the object and instantiate it each time help is invoked – you cannot leave the object open to reuse later.

Shared Calls to stzHLP

If your application has multiple windows from which help can be called, you can simplify the logic to set up and call stzHLP by sharing all but the 'AddPanel method calls. For example, the stzCMD SF command has help for its SF, SF2, SF3, SF4, SF5 and SF9 windows. In its code, shown below, the call to instantiate the stzHLP object, set the NomadsLib$ property and the position and size of the parent window is always the same, with the 'AddPanel calls conditioned with a SWITCH statement…

```
TOOLHELP:
LOCAL stzOBJ, tmpCTL
stzOBJ = NEW("*stzHLP", %stzCMD$, %stzHeading$, %stzHelp01$, %stzHelp02$)
stzOBJ'NomadsLib$ = %stzNomadsLibHelp$
stzOBJ'WindowCol  = INT(MAX(0,(DEC($00$+MID(OBJ(0),21,2))-%NOMAD_WORKAREA_X))/%NOMAD_XCHAR)
stzOBJ'WindowRow  = INT(MAX(0,(DEC($00$+MID(OBJ(0),23,2))-%NOMAD_WORKAREA_Y))/%NOMAD_YCHAR)
stzOBJ'WindowCols = DEC($00$+MID(OBJ(0),33,2))
stzOBJ'WindowRows = DEC($00$+MID(OBJ(0),35,2))
stzOBJ'AddPanel("hlp"+SCRN_ID$+"-00"
SWITCH SCRN_ID$
       CASE "SF"
              tmpCTL = DB_Filter.ctl    ; stzOBJ'AddPanel("hlp"+SCRN_ID$+"-01", tmpCTL'Col…
              tmpCTL = LB_Main.ctl      ; stzOBJ'AddPanel("hlp"+SCRN_ID$+"-02", tmpCTL'Col…
              tmpCTL = BT_Clipboard.ctl ; stzOBJ'AddPanel("hlp"+SCRN_ID$+"-03", tmpCTL'Col…
       BREAK
       CASE "SF2"
              tmpCTL = ML_Key.ctl             ; stzOBJ'AddPanel("hlp"+SCRN_ID$+"-01", tmpCTL'Col…
              tmpCTL = BT_KeyList.ctl         ; stzOBJ'AddPanel("hlp"+SCRN_ID$+"-02", tmpCTL'Col…
              RB_DataViewMode.ctl'ID = 1    ! used first radio-button
              tmpCTL = RB_DataViewMode.ctl    ; stzOBJ'AddPanel("hlp"+SCRN_ID$+"-03", tmpCTL'Col…
              tmpCTL = LB_Contents.ctl        ; stzOBJ'AddPanel("hlp"+SCRN_ID$+"-04", tmpCTL'Col…
              tmpCTL = BT_Dataview_Delete.ctl ; stzOBJ'AddPanel("hlp"+SCRN_ID$+"-05", tmpCTL'Col…
       BREAK
       CASE "SF3"
              tmpCTL = ML_KeyListKey.ctl    ; stzOBJ'AddPanel("hlp"+SCRN_ID$+"-01", tmpCTL'Col…
              tmpCTL = DB_KeyListFilter.ctl ; stzOBJ'AddPanel("hlp"+SCRN_ID$+"-02", tmpCTL'Col…
              tmpCTL = LB_ContentsKeys.ctl  ; stzOBJ'AddPanel("hlp"+SCRN_ID$+"-03", tmpCTL'Col…
       BREAK
       CASE "SF4"
              tmpCTL = DB_OpenFile.ctl ; stzOBJ'AddPanel("hlp"+SCRN_ID$+"-01", tmpCTL'Col…
       BREAK
       CASE "SF5"
              tmpCTL = ML_EditVal.ctl ; stzOBJ'AddPanel("hlp"+SCRN_ID$+"-01", tmpCTL'Col…
       BREAK
       CASE "SF9"
              tmpCTL = ML_CopyFolder.ctl ; stzOBJ'AddPanel("hlp"+SCRN_ID$+"-01", tmpCTL'Col…
       BREAK
END SWITCH
stzOBJ'Go()
DROP OBJECT stzOBJ
RETURN
```