List.RC

```
/*****************************************************************************
list.rc
produced by IBM Object REXX Resource Workshop
*****************************************************************************/

#define DIALOG_1        1
#define IDC_LIST        10
#define IDC_DIRECTORY   11
#define IDC_REFRESH     12
#define IDC_RADIOICON   15
#define IDC_RADIOSMLICON        16
#define IDC_RADIOREPORT 14
#define IDC_RADIOLIST   13

DIALOG_1 DIALOG -6, -70, 307, 211
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Object REXX List Sample"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "OK", IDOK, 248, 6, 50, 14
 PUSHBUTTON "Cancel", IDCANCEL, 248, 24, 50, 14
 CONTROL "ListView", IDC_LIST, "SysListView32", LVS_REPORT | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP, 9, 27, 226, 174
 LTEXT "Directory:", -1, 6, 8, 60, 8
 EDITTEXT IDC_DIRECTORY, 51, 6, 183, 14, WS_BORDER | WS_TABSTOP
 PUSHBUTTON "Refresh", IDC_REFRESH, 248, 50, 50, 14
 AUTORADIOBUTTON "List", IDC_RADIOLIST, 247, 74, 44, 12, BS_AUTORADIOBUTTON | WS_GROUP
 AUTORADIOBUTTON "Report", IDC_RADIOREPORT, 247, 87, 45, 11
 AUTORADIOBUTTON "Icon", IDC_RADIOICON, 247, 99, 35, 12
 AUTORADIOBUTTON "Small Icon", IDC_RADIOSMLICON, 247, 112, 53, 12
}
```

List.REX

```
/*****************************************************************************/
/* Name: List.rex                                                            */
/* Type: Object REXX Script using OODialog                                   */
/* Author: Christian Michel                                                  */
/* Resource: List.rc                                                         */
/*                                                                           */
/* Description:                                                              */
/* This file has been created by the Object  REXX Workbench OODIALOG         */
/* template generator.                                                       */
/*                                                                           */
/* Copyright (C) _____, 1999. All Rights Reserved.           */
/*                                                                           */
/*****************************************************************************/

MyListDialog = .MyListDialogClass~new
If MyListDialog~InitCode = 0 Then Do
  rc = MyListDialog~Execute("SHOWTOP")
End

/* Add program code here */

Exit   /* leave program */

::REQUIRES "OODWIN32.CLS"    /* This file contains the OODIALOG classes */

/* ---------------------------- Directives --------------------------------*/

::CLASS MyListDialogClass SUBCLASS UserDialog INHERIT AdvancedControls MessageExtensions

::METHOD Init
  Expose curStyle
  Forward Class (super) Continue /* call parent constructor */
  InitRet = Result

  If self~Load("List.rc", ) \= 0 Then Do
     self~InitCode = 1
     Return 1
  End

  /* Connect dialog control items to class methods */
  self~ConnectButton(1,"Ok")
  self~ConnectButton(2,"Cancel")
  self~ConnectListNotify("IDC_LIST","Activate","OnActivate_IDC_LIST")
```

```
  self~ConnectButton("IDC_REFRESH","IDC_REFRESH")
  self~ConnectControl("IDC_RADIOLIST","ShowList")
  self~ConnectControl("IDC_RADIOREPORT","ShowReport")
  self~ConnectControl("IDC_RADIOICON","ShowIcon")
  self~ConnectControl("IDC_RADIOSMLICON","ShowSmallIcon")

  /* Initial values that are assigned to the object attributes  */
  self~IDC_LIST= ''                                  /* List Control */
  self~IDC_DIRECTORY= Directory()                    /* Entry Line */
  self~IDC_RADIOLIST=0                               /* Radio Button */
  self~IDC_RADIOREPORT=1                             /* Radio Button */
  self~IDC_RADIOICON=0                               /* Radio Button */
  self~IDC_RADIOSMLICON=0                            /* Radio Button */

  /* Add your initialization code here */
  curStyle = "REPORT"
  Return InitRet


::METHOD InitDialog
  Expose curList
  InitDlgRet = self~InitDialog:super

  /* Set image list for List control IDC_LIST */
  curList = self~GetListControl("IDC_LIST")
  curList~SetSmallImages("ListSml.bmp",17,17)
  curList~SetImages("ListIco.bmp",34,34)


  /* Add items (and columns) to the list control here using Add or  AddRow (and InsertColumn) */
  curList~InsertColumn(0, "Filename", 50)
  curList~InsertColumn(1, "Size", 25, "RIGHT")
  curList~InsertColumn(2, "Date", 35)
  curList~InsertColumn(3, "Time", 30)
  curList~InsertColumn(4, "Attribs", 25)


  /* Initialization Code (e.g. fill list and combo boxes)  */
  Return InitDlgRet

/* --------------------- message handler ----------------------------------*/

  /* Method Ok is connected to item 1 */
::METHOD Ok
  resOK = self~OK:super  /* make sure self~Validate is called and self~InitCode is set to 1 */
  self~Finished = resOK  /* 1 means close dialog, 0 means keep open  */
  Return resOK

  /* Method Cancel is connected to item 2  */
::METHOD Cancel
  resCancel = self~Cancel:super  /* make sure self~InitCode is set to 2 */
  self~Finished = resCancel      /* 1 means close dialog, 0 means keep open  */
  Return resCancel


  /* Method OnActivate_IDC_LIST handles notification 'Activate' for item  IDC_LIST */
::METHOD OnActivate_IDC_LIST
  Expose curList
  info. = curList~ItemInfo(curList~Selected)
  If info.!Image = 0 Then Do
    /* change to the selected directory  */
    curDir = self~GetValue("IDC_DIRECTORY")
    If curDir~Right(1) = "\" Then
      curDir = curDir~Left(curDir~Length - 1)

    If info.!Text = ".." Then
      /* remove one directory from current path  */
      newDir = curDir~Left(curDir~LastPos("\") - 1) || "\"
    Else
      newDir = curDir || "\" || info.!Text || "\"

    self~SetValue("IDC_DIRECTORY",  newDir)
    self~IDC_REFRESH
  End


  /* Method IDC_REFRESH is connected to item IDC_REFRESH */
::METHOD IDC_REFRESH
  Expose curList

  curList~DeleteAll

  /* get a list of all files in specified directory  */
  curDir = self~GetValue("IDC_DIRECTORY")
```

```
  If curDir~Right(1) \= "\" Then
    curDir = curDir || "\"

  If curDir~Length > 3 Then
    curList~AddRow(,0, "..")

  pattern = curDir || "*"
  Call SysFileTree pattern, "Files.", "BL"
  Do i = 1 To Files.0
    FileName = Files.i~SubStr(Files.i~WordIndex(5))
    FileName = FileSpec("NAME", FileName)
    FileSize = Files.i~Word(3)
    FileDate = Files.i~Word(1)
    FileTime = Files.i~Word(2)
    FileAttribs = Files.i~Word(4)
    If FileAttribs~Pos("D") \= 0 Then
      FileIcon = 0
    Else
      FileIcon = 1
    curList~AddRow(,FileIcon, FileName, FileSize,,
                   FileDate, FileTime, FileAttribs)
  End

  /* Method ShowList is connected to item IDC_RADIOLIST */
::METHOD ShowList
  Expose curList curStyle
  curList~ReplaceStyle(curStyle, "LIST")
  curStyle = "LIST"

  /* Method ShowReport is connected to item IDC_RADIOREPORT */
::METHOD ShowReport
  Expose curList curStyle
  curList~ReplaceStyle(curStyle, "REPORT")
  curStyle = "REPORT"

  /* Method ShowIcon is connected to item IDC_RADIOICON */
::METHOD ShowIcon
  Expose curList curStyle
  curList~ReplaceStyle(curStyle, "ICON")
  curStyle = "ICON"

  /* Method ShowSmallIcon is connected to item IDC_RADIOSMLICON */
::METHOD ShowSmallIcon
  Expose curList curStyle
  curList~ReplaceStyle(curStyle, "SMALLICON")
  curStyle = "SMALLICON"
```

Tree.RC

```
/*****************************************************************************
tree.rc
produced by IBM Object REXX Resource Workshop
*****************************************************************************/

#define DIALOG_1        1
#define IDC_TREE        10
#define IDC_NEWFOLDER   11
#define IDC_NEWITEM     12
#define IDC_DELETE      13

DIALOG_1 DIALOG 6, 15, 310, 177
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Object REXX Tree Control Sample"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "OK", IDOK, 248, 6, 50, 14
 PUSHBUTTON "Cancel", IDCANCEL, 248, 24, 50, 14
 CONTROL "Tree", IDC_TREE, "SysTreeView32", TVS_HASBUTTONS | TVS_HASLINES | TVS_LINESATROOT | WS_CHILD |
WS_VISIBLE | WS_BORDER | WS_TABSTOP, 9, 22, 225, 144
 LTEXT "Create your own tree structure:", -1, 7, 9, 124, 8
 PUSHBUTTON "New Folder", IDC_NEWFOLDER, 248, 53, 50, 14
 PUSHBUTTON "New Item", IDC_NEWITEM, 248, 73, 50, 14
 PUSHBUTTON "Delete", IDC_DELETE, 248, 93, 50, 14
}
```

Tree.REX

```
/**************************************************************************/
/* Name: Tree.rex                                                         */
/* Type: Object REXX Script using OODialog                                */
/* Author: Christian Michel                                               */
/* Resource: Tree.rc                                                      */
/*                                                                        */
/* Description:                                                           */
/* This file has been created by the Object  REXX Workbench OODIALOG      */
/* template generator.                                                    */
/*                                                                        */
/* Copyright (C) _____, 1999. All Rights Reserved.                        */
/*                                                                        */
/**************************************************************************/

MyTreeDialog = .MyTreeDialogClass~new
If MyTreeDialog~InitCode = 0 Then Do
  rc = MyTreeDialog~Execute("SHOWTOP")
End

/* Add program code here */

Exit   /* leave program */

::REQUIRES "OODWIN32.CLS"    /* This file contains the OODIALOG classes */
::REQUIRES "WINSYSTM.CLS"    /* This file contains the VirtualKeyCodes class */

/* ---------------------------- Directives --------------------------------*/

::CLASS MyTreeDialogClass SUBCLASS UserDialog,
        INHERIT AdvancedControls MessageExtensions VirtualKeyCodes

::METHOD Init
  Forward Class (super) Continue /* call parent constructor */
  InitRet = Result

  If self~Load("Tree.rc", ) \= 0 Then Do
     self~InitCode = 1
     Return 1
  End

  /* Connect dialog control items to class methods */
  self~ConnectButton(1,"Ok")
  self~ConnectButton(2,"Cancel")
  self~ConnectTreeNotify("IDC_TREE","SelChanged","OnSelChanged_IDC_TREE")
  self~ConnectTreeNotify("IDC_TREE","KeyDown","OnKeyDown_IDC_TREE")
  self~ConnectButton("IDC_NEWFOLDER","IDC_NEWFOLDER")
  self~ConnectButton("IDC_NEWITEM","IDC_NEWITEM")
  self~ConnectButton("IDC_DELETE","IDC_DELETE")

  /* Initial values that are assigned to the object attributes */
  self~IDC_TREE= ''                                  /* Tree Control */

  /* Add your initialization code here */
  Return InitRet


::METHOD InitDialog
  Expose curTree selItem
  InitDlgRet = self~InitDialog:super

  /* Set image list for Tree control IDC_TREE */
  curTree = self~GetTreeControl("IDC_TREE")
  If curTree \= .Nil Then curTree~SetImages("Tree.bmp",16,12)

  /* Add items to the tree control here using Add or Insert */
  curTree~Add("root", 0, 1)
  curTree~Add(,"Folder 1", 0, 1)
  curTree~Add(,,"Item 1", 2, 3)
  curTree~Add(,,"Item 2", 2, 3)
  curTree~Add(,"Folder 2", 0, 1)
  curTree~Add(,,"Item 3", 2, 3)
  curTree~Add(,,"Item 4", 2, 3)
  selItem = 0

  /* Initialization Code (e.g. fill list and combo boxes) */
  Return InitDlgRet

/* --------------------- message handler ---------------------------------*/

  /* Method Ok is connected to item 1 */
```

```
::METHOD Ok
  Expose curTree

  /* list items in tree, then terminate */
  Say "The following items are in the tree:"
  Call ListItems curTree, curTree~Root, 0

  resOK = self~OK:super  /* make sure self~Validate is called and self~InitCode is set to 1 */
  self~Finished = resOK  /* 1 means close dialog, 0 means keep open */
  Return resOK

  /* Method Cancel is connected to item 2 */
::METHOD Cancel
  resCancel = self~Cancel:super  /* make sure self~InitCode is set to 2 */
  self~Finished = resCancel      /* 1 means close dialog, 0 means keep open */
  Return resCancel


    /* Method OnSelChanged_IDC_TREE handles notification 'SelChanged' for item IDC_TREE */
::METHOD OnSelChanged_IDC_TREE
  Expose curTree selItem
  curTree~Modify(selItem,,,,"NOTBOLD")
  selItem = curTree~Selected
  curTree~Modify(selItem,,,,"BOLD")


  /* Method OnKeyDown_IDC_TREE handles notification 'KeyDown' for item IDC_TREE */
::METHOD OnKeyDown_IDC_TREE
  Expose curTree selItem
  Use Arg treeId, Key

  /* if DELETE key is pressed delete the selected item */
  If self~KeyName(Key) = "DELETE" Then
    self~IDC_DELETE


  /* Method IDC_NEWFOLDER is connected to item IDC_NEWFOLDER */
::METHOD IDC_NEWFOLDER
  Expose curTree selItem

  /* get new folder name */
  Input = .InputBox~New("Enter new folder name:", "Folder Name")
  newFldr = Input~Execute
  If newFldr \= "" Then Do
    /* look at selected item */
    info. = curTree~ItemInfo(selItem)
    If info.!Image = 0 Then
      /* this is a folder, create new folder as a child */
      newPos = selItem
    Else
      /* this is an itme, create new folder as a sibling */
      newPos = curTree~Parent(selItem)
    curTree~Insert(newPos, "LAST", newFldr, 0, 1)
    curTree~Expand(newPos)
  End


  /* Method IDC_NEWITEM is connected to item IDC_NEWITEM */
::METHOD IDC_NEWITEM
  Expose curTree selItem

  /* get new item name */
  Input = .InputBox~New("Enter new item name:", "Item Name")
  newItem = Input~Execute
  If newItem \= "" Then Do
    /* look at selected item */
    info. = curTree~ItemInfo(selItem)
    If info.!Image = 0 Then
      /* this is a folder, create new item as a child */
      newPos = selItem
    Else
      /* this is an itme, create new folder as a sibling */
      newPos = curTree~Parent(selItem)
    curTree~Insert(newPos, "LAST", newItem, 2, 3)
    curTree~Expand(newPos)
  End


  /* Method IDC_DELETE is connected to item IDC_DELETE */
::METHOD IDC_DELETE
  Expose curTree selItem

  /* look at selected item */
  info. = curTree~ItemInfo(selItem)
```

```
  If info.!Image = 0 Then
    itemDesc = "Folder"
  Else
    itemDesc = "Item"

  query = "Are you sure you want to delete"  itemDesc,
          "'" || info.!Text || "'?"

  If RxMessageBox(query, "Confirm Delete", "YESNO") = 6 Then Do
    curTree~Delete(selItem)
    selItem = 0
  End

::ROUTINE ListItems
  Use Arg treeCtrl, Item, Indent

  Do While Item \= 0
    info. = treeCtrl~ItemInfo(Item)
    Say Copies(" ", Indent) || info.!Text

    Child = treeCtrl~Child(Item)
    If Child \= 0 Then
      Call ListItems treeCtrl, Child, Indent + 2

    Item = treeCtrl~Next(Item)
  End
```

Progress.RC

```
/*****************************************************************************
progress.rc
produced by IBM Object REXX Resource Workshop
*****************************************************************************/

#define DIALOG_1        1
#define IDC_1   101
#define IDC_MSG 102

DIALOG_1 DIALOG 117, 17, 123, 63
STYLE WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Progress Title"
FONT 8, "MS Sans Serif"
{
 PUSHBUTTON "Cancel", IDCANCEL, 36, 45, 50, 14
 LTEXT "Progress Message", IDC_MSG, 7, 8, 109, 8
 CONTROL "Progress", IDC_1, "msctls_progress32", 0 | WS_CHILD | WS_VISIBLE, 12, 25, 100, 10
}
```

Progress.REX

```
/*****************************************************************************/
/* Name: Progress.rex                                                        */
/* Type: Object REXX Script using OODialog                                   */
/* Author: Christian Michel                                                  */
/* Resource: Progress.rc                                                     */
/*                                                                           */
/* Description:                                                              */
/* This file has been created by the Object REXX Workbench OODIALOG          */
/* template generator.                                                       */
/*                                                                           */
/* Copyright (C) _____, 1999. All Rights Reserved.                           */
/*                                                                           */
/*****************************************************************************/

MyProgressDialog = .MyProgressClass~new("Copying Files...", "")
If MyProgressDialog~InitCode = 0 Then Do
  rc = MyProgressDialog~ExecuteAsync(100, "SHOWTOP")
End

/* Add program code here */
Do i = 0 To 100 By 10
  If MyProgressDialog~Finished \= 0 Then Do
    Say "Cancel was pressed!"
    Exit
  End
  CopyFilename = "FILE" || Right(i, 3, "0") || ".DAT"
  MyProgressDialog~SetMessage("Copying" CopyFilename)
  MyProgressDialog~SetPos(i)
```

```
  Call SysSleep 1
End

MyProgressDialog~SetMessage("All files copied successfully!")
Call SysSleep 3
MyProgressDialog~Close
MyProgressDialog~EndAsyncExecution
Exit   /* leave program */

::REQUIRES "OODWIN32.CLS"    /* This file contains the OODIALOG classes */

/* ---------------------------- Directives --------------------------------*/

::CLASS MyProgressClass SUBCLASS UserDialog INHERIT AdvancedControls

::METHOD Init
  Expose TitleText MessageText
  Use Arg TitleText, MessageText
  Forward Class (super) Continue /* call parent constructor */
  InitRet = Result

  If self~Load("Progress.rc", "DIALOG_1") \= 0 Then Do
     self~InitCode = 1
     Return 1
  End

  /* Connect dialog control items to class methods */
  self~ConnectButton(2,"Cancel")

  /* Initial values that are assigned to the object attributes */

  /* Add your initialization code here */
  Return InitRet


::METHOD InitDialog
  Expose curPB TitleText MessageText
  InitDlgRet = self~InitDialog:super

  self~Title = TitleText
  self~SetStaticText("IDC_MSG", MessageText)

  /* Initialize progress bar IDC_1 */
  curPB = self~GetProgressBar("IDC_1")
  If curPB \= .Nil Then Do
     curPB~SetRange(0,100)
     curPB~SetStep(1)
     curPB~SetPos(0)
  End

  /* Initialization Code (e.g. fill list and combo boxes) */
  Return InitDlgRet


/* --------------------- message handler ----------------------------------*/

  /* Method Cancel is connected to item 2 */
::METHOD Cancel
  resCancel = self~Cancel:super  /* make sure self~InitCode is set to 2 */
  self~Finished = resCancel      /* 1 means close dialog, 0 means keep open */
  self~EndAsyncExecution
  Return resCancel

::METHOD Close UNGUARDED
  self~Finished = 1

::METHOD SetPos UNGUARDED
  Expose curPB
  Use Arg newPos
  curPB~SetPos(newPos)

::METHOD SetMessage UNGUARDED
  Use Arg newMsg
  self~SetStaticText("IDC_MSG", newMsg)
```

Slider.RC

```
/*****************************************************************************
slider.rc
produced by IBM Object REXX Resource Workshop
*****************************************************************************/

#define DIALOG_1        1
#define IDC_SLDR2       102
#define IDC_SLDR1_VAL   103
#define IDC_SLDR2_VAL   104
#define IDC_SLDR1       101

DIALOG_1 DIALOG 13, 27, 216, 100
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Object REXX Slider Sample"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "OK", IDOK, 159, 9, 50, 14
 PUSHBUTTON "Cancel", IDCANCEL, 159, 27, 50, 14
 CONTROL "Slider", IDC_SLDR1, "msctls_trackbar32", TBS_AUTOTICKS | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 11,
29, 139, 20
 CONTROL "Slider", IDC_SLDR2, "msctls_trackbar32", TBS_AUTOTICKS | TBS_ENABLESELRANGE | WS_CHILD |
WS_VISIBLE | WS_TABSTOP, 11, 68, 139, 20
 LTEXT "Slider 1 Value:", -1, 15, 10, 60, 8
 LTEXT "Slider 2 Value:", -1, 16, 55, 60, 8
 LTEXT "0", IDC_SLDR1_VAL, 84, 10, 60, 8
 LTEXT "0", IDC_SLDR2_VAL, 81, 55, 60, 8
}
```

Slider.REX

```
/*****************************************************************************/
/* Name: slider.rex                                                          */
/* Type: Object REXX Script using OODialog                                   */
/* Author: Christian Michel                                                  */
/* Resource: Slider.rc                                                       */
/*                                                                           */
/* Description:                                                              */
/* This file has been created by the Object  REXX Workbench OODIALOG         */
/* template generator.                                                       */
/*                                                                           */
/* Copyright (C) _____, 1999. All Rights Reserved.           */
/*                                                                           */
/*****************************************************************************/

MySlidersDialog = .MySlidersClass~new
If MySlidersDialog~InitCode = 0 Then Do
  rc = MySlidersDialog~Execute("SHOWTOP")
  Say MySlidersDialog~Slider1Pos  MySlidersDialog~Slider2Pos
End

/* Add program code here */

Exit   /* leave program */

::REQUIRES "OODWIN32.CLS"    /* This file contains the OODIALOG classes */

/* ---------------------------- Directives --------------------------------*/

::CLASS MySlidersClass subclass UserDialog inherit AdvancedControls MessageExtensions

::METHOD Init
  Forward Class (super) Continue /* call parent constructor */
  InitRet = Result

  If self~Load("Slider.rc", ) \= 0 Then Do
     self~InitCode = 1
     Return 1
  End

  /* Connect dialog control items to class methods */
  self~ConnectButton(1,"Ok")
  self~ConnectButton(2,"Cancel")

  /* Initial values that are assigned to the object attributes */
  self~IDC_SLDR1=0                                  /* Slider Control */
  self~IDC_SLDR2=0                                  /* Slider Control */
```

```
  /* Add your initialization code here */
  Return InitRet


::METHOD InitDialog
  Expose curSL1 curSL2
  InitDlgRet = self~InitDialog:super
  self~ConnectSliderNotify("IDC_SLDR1","EndTrack","OnEndTrack_IDC_SLDR1")
  self~ConnectSliderNotify("IDC_SLDR2","EndTrack","OnEndTrack_IDC_SLDR2")

  /* Initialize slider IDC_SLDR1 */
  curSL1 = self~GetSliderControl("IDC_SLDR1")
  If curSL1 \= .Nil Then Do
    curSL1~InitRange(0,100)
    curSL1~SetLineStep(1)
    curSL1~SetPageStep(10)
    curSL1~SetTickFrequency(10)
  End

  /* Initialize slider IDC_SLDR2 */
  curSL2 = self~GetSliderControl("IDC_SLDR2")
  If curSL2 \= .Nil Then Do
    curSL2~InitRange(0,100)
    curSL2~SetLineStep(1)
    curSL2~SetPageStep(10)
    curSL2~SetTickFrequency(10)
  End

  /* Initialization Code (e.g. fill list and combo boxes)  */
  Return InitDlgRet

/* --------------------- message handler ----------------------------------*/

  /* Method Ok is connected to item 1 */
::METHOD Ok
  resOK = self~OK:super  /* make sure self~Validate is called and self~InitCode is set to 1 */
  self~Finished = resOK  /* 1 means close dialog, 0 means keep open */
  Return resOK

  /* Method Cancel is connected to item 2 */
::METHOD Cancel
  resCancel = self~Cancel:super  /* make sure self~InitCode is set to 2 */
  self~Finished = resCancel      /* 1 means close dialog, 0 means keep open */
  Return resCancel


  /* Method OnEndTrack_IDC_SLDR1 handles notification 'EndTrack' for slider IDC_SLDR1 */
::METHOD OnEndTrack_IDC_SLDR1
  Expose curSL1
  Pos = curSL1~Pos
  self~SetStaticText("IDC_SLDR1_VAL", Pos)
  self~Slider1Pos = Pos

  /* Method OnEndTrack_IDC_SLDR2 handles notification 'EndTrack' for slider IDC_SLDR2 */
::METHOD OnEndTrack_IDC_SLDR2
  Expose curSL2
  Pos = curSL2~Pos
  self~SetStaticText("IDC_SLDR2_VAL", Pos)
  self~Slider2Pos = Pos

::METHOD Slider1Pos ATTRIBUTE
::METHOD Slider2Pos ATTRIBUTE
```