Powersoft

Open Tools from Sybase, Inc.

PowerBuilder Foundation Class Library

Object Reference: Volume 1

Version 6

PowerBuilder®

AA0523

October 1997

Contents

VOLUME 2

About This Book

Subject

This book describes the objects in the PowerBuilder Foundation Class Library (PFC). Each object's discussion includes:

- A diagram providing a quick overview of the object's capabilities
- Instance variables
- Events and user events
- Object functions

Online Help

The PFC online Help (pbpfc60.hlp) contains all the objects, events, and functions explained in this book. This Help file includes customized buttons and links to help you navigate through inheritance hierarchies and related objects.

FOR INFO For PFC usage information, see the *PowerBuilder Foundation Class Library User's Guide*.

Audience

This book is for experienced PowerBuilder developers. It assumes that:

- You are currently developing applications using PowerBuilder and understand the concepts and techniques described in the PowerBuilder *Application Techniques* book
- You understand SQL and how to use your site-specific DBMS

CHAPTER 1

Object Reference Conventions

About this chapter

This chapter describes the conventions used in this manual and the PFC online Help file.

Naming conventions

Object naming conventions

PFC uses the following prefix standard for object names:

pfcobject_type_objectname

where:

- *pfcobject* indicates whether the object is part of the PFC level or the extension level. Objects that are part of the PFC level have the prefix PFC_
- *type* indicates the object type

This table describes the object types.

Prefix	Description
m_	Menu
n_	Standard class user object
n_cst	Custom class user object
s_	Global structure
u_	Visual user object
w_	Window

For example:

- Pfc_w_master is the master window and is in the PFC level
- Pfc_u_tvs is the TreeView visual user object and is in the PFC level
- U_dw is the DataWindow visual user object and is in the extension level
- N_cst_dwsrv is the custom class user object for DataWindow services and is in the extension level
- N_tr is the transaction standard class user object and is in the extension level

Variable naming conventions

PFC uses the following standard for variable names:

<scope><datatype>_variablename

Scope is one of the following:

Prefix	Description
a	Argument to an event or function

Prefix	Description
g	Global variable
i	Instance variable
l	Local variable
s	Shared variable

For standard data types, *datatype* is one of the following:

Prefix	Description
a	Any
blb	Blob
b	Boolean
ch	Character
d	Date
dtm	DateTime
dc	Decimal
dbl	Double
e	Enumerated
i	Integer
l	Long
r	Real
s	String
tm	Time
ui	UnsignedInteger
ul	UnsignedLong

For reference variables, *datatype* is one of the following:

Prefix	Description
app	Application
ab	ArrayBounds
cbx	CheckBox

Prefix	Description
cb	CommandButton
cd	ClassDefinition
cdo	ClassDefinitionObject
cn	Connection
cninfo	ConnectionInfo
cno	ConnectObject
cxk	ContextKeyword
cxinfo	ContextInformation
cpp	cplusplus
ds	DataStore
dw	DataWindow
dwc	DataWindowChild
drg	DragObject
drw	DrawObject
ddplb	DropDownPictureListBox
ddlb	DropDownListBox
dwo	DWobject
dda	DynamicDescriptionArea
dsa	DynamicStagingArea
ed	EnumerationDefinition
eid	EnumerationItemDefinition
em	EditMask
env	Environment
err	Error
ext	ExtObject
gr	Graph
go	GraphicObject

Prefix	Description
grax	GrAxis
grda	GrDispAttr
gb	GroupBox
hsb	HorizontalScrollBar
inet	Inet
ir	InternetResult
ln	Line
lb	ListBox
lv	ListView
lvi	ListViewItem
mfd	MailFileDescription
mm	MailMessage
mr	MailRecipient
ms	MailSession
mdi	MDIClient
m	Menu
mc	MenuCascade
msg	Message
mle	MultiLineEdit
nv	NonVisualObject
oc	OleControl
oo	OleObject
ostg	OleStorage
omc	OmControl
omcc	OmCustomControl
omec	OmEmbeddedControl
omo	OmObject

Prefix	Description
omstm	OmStream
omstg	OmStorage
oval	Oval
p	Picture
pb	PictureButton
pbcpp	PBToCPPObject
plb	PictureListBox
pl	Pipeline
po	PowerObject
procall	ProfileCall
proclass	ProfileClass
proln	ProfileLine
prort	ProfileRoutine
pro	Profiling
rb	RadioButton
rec	Rectangle
rem	RemoteObject
rte	RichTextEdit
rrec	RoundRectangle
rteo	RteObject
scrd	ScriptDefinition
sle	SingleLineEdit
srv	Service
st	StaticText
std	SimpleTypeDefinition
str	Structure
tab	Tab

Prefix	Description
tabpg	TabPage
tcan	TraceActivityNode
tcbe	TraceBeginEnd
tcerr	TraceError
tcf	TraceFile
tcln	TraceLine
tcgc	TraceGarbageCollect
tco	TraceObject
tcrt	TraceRoutine
tcsql	TraceSQL
tct	TraceTree
tctn	TraceTreeNode
tcterr	TraceTreeError
tctsql	TraceTreeSQL
tctgc	TraceTreeGarbageCollect
tctln	TraceTreeLine
tcto	TraceTreeObject
tctrt	TraceTreeRoutine
tctu	TraceTreeUser
tcu	TraceUser
td	TypeDefinition
tr	Transaction
trp	Transport
tv	TreeView
tvi	TreeViewItem
uo	UserObject
vrcd	VariableCardinalityDefinition

Prefix	Description
vrd	VariableDefinition
vsb	VerticalScrollBar
wo	WindowObject
w	Window

Function naming conventions

Global functions use the f_ prefix and object functions use the of_ prefix.

Online Help

Accessing Help

The PFC online Help file contains all information found in this manual, as well as hypertext links among related topics. There are many ways to access this file:

- **From the PowerBuilder Help Contents tab (Windows 95 and Windows NT only)** The PowerBuilder Help screen includes a jump to PFC Help.
- **Double-click the filename (pbpfc60.hlp) from the Explorer or File Manager** This displays the Contents tab, which you can use to view lists of objects.
- **Using SHIFT+F 1 from the PowerScript editor (Windows 95 and Windows NT only)** This displays Help for the selected object function or event.

Navigating through the Help file

The PFC Help file provides many helpful links among objects, functions, and events:

- **From an object** You can display a dialog box with jumps to instance variables, events, and functions. You can also:
  - Click the See Also button to jump to related objects and the Powersoft Online Books, if available to your workstation.
  - Click the Descendants button to display a list of descendants, if any
  - Click the Ancestor button to jump to the ancestor object
- **From an event or function** You can display a code example, if appropriate, for the event or function. You can also:
  - Click the See Also button to jump to related objects and the Powersoft Online Books, if available to your workstation.
  - Click the Object button to jump to the parent object

Using the Powersoft Online Books

Most online Help topics include a jump to related topics in the Powersoft Online Books. To use the Powersoft Online Books, you must have installed software on your workstation, and have access to the files containing the Online Books (typically either on the PowerBuilder Online Documentation CD or a network drive).

Sample application and code examples

Sample application

The PEAT sample application shows you how to use PFC for a simple project estimation and tracking system. Its includes examples of windows, DataWindows, tabs, TreeViews, ListViews, and custom class user objects.

Code examples

The EXAMPFC code examples application shows you how to use selected PFC objects and services. Its includes examples of application services, windows, window services, DataWindows, DataWindow services, tabs, TreeViews, ListViews, and custom class user objects.

By default, the code examples are installed in the PFC\Examples directory.

CHAPTER 2

Window Objects

About this chapter

This chapter describes the base-class windows in the PowerBuilder Foundation Class Library (PFC). Each discussion includes information on usage, instance variables, events, and functions.

Contents

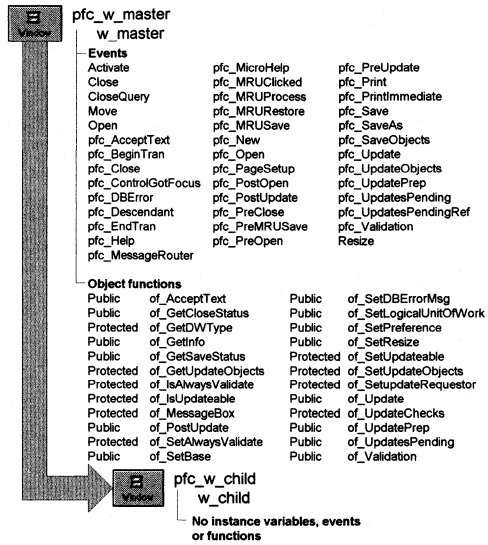
The windows are listed in alphabetical order. Each window's discussion includes alphabetical listings of instance variables, events, and object functions.

w_child

Description

Ancestor for all PFC child windows.

Ancestry



Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use this window as the ancestor for all of your application's child windows.

See also

w_master

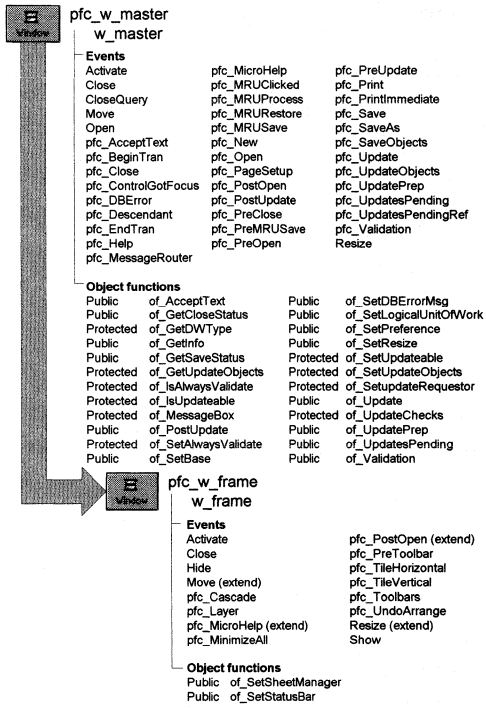
w_frame

Description

Ancestor frame window for all MDI applications.

It includes MicroHelp automatically. To create a frame window without MicroHelp, either modify or rename your application's frame window.

Ancestry



Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

m_frame
n_cst_toolbarattrib
w_statusbar
w_toolbars

Usage

Use this window or a descendant as the frame window for all MDI applications that use PFC.

See also

m_frame
w_master
w_statusbar

w_toolbars

Instance variables

W_frame includes instance variables:

Instance variable	Description	Data type	Access	Usage
inv_sheetmanager	Reference variable for sheet management service	n_cst_winsrv_sheetmanager	Public	Set with of_SetSheetManager
inv_statusbar	Reference variable for status bar service	n_cst_winsrv_statusbar	Public	Set with of_SetStatusBar

Events

W_frame includes precoded events:

Activate	pfc_PostOpen
Close	pfc_PreToolbar
Hide	pfc_TileHorizontal
Move	pfc_TileVertical
pfc_Cascade	pfc_Toolbars
pfc_Layer	pfc_UndoArrange
pfc_MicroHelp	Resize
pfc_MinimizeAll	Show

Activate

Description

Calls the n_cst_appmanager of_SetFrame function to establish this window as the active frame.

This event extends the ancestor Activate event.

Usage

This event executes when the window becomes active.

Close

Description

Destroys any instantiated services.

Usage — This event executes when the window closes.

Hide

Description — Calls the n_cst_winsrv_statusbar pfc_Hide event.

Usage — This event executes when the window is hidden.

Move

Description — Positions the status bar when the user moves the window.

This event extends the ancestor Move event.

Usage — This event executes when the user moves the frame window.

pfc_Cascade

Description — Calls the n_cst_winsrv_sheetmanager pfc_Cascade event.

Return value — Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage — This event executes when the user selects Window>Cascade from the menu bar of a menu descended from the PFC m_master menu.

pfc_Layer

Description — Calls the n_cst_winsrv_sheetmanager pfc_Layer event.

Return value — Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage — This event executes when the user selects Window>Layer from the menu bar of a menu descended from the PFC m_master menu.

pfc_MicroHelp

Description — Displays the passed string in the status bar.

This event extends the w_master pfc_MicroHelp event.

Syntax — *instancename*.EVENT **pfc_MicroHelp** (*microhelp*)

Argument	Description
instancename	Instance name of w_frame
microhelp	String to be displayed in the status bar

Return value

None

Usage

This event is called by u_dw ItemFocusChanged event. It is also extended in w_sheet.

pfc_MinimizeAll

Description

Minimizes all open sheets.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event executes when the user selects Window>Minimize All from the menu bar of a menu descended from the PFC m_master menu.

pfc_PostOpen

Description

Opens the w_statusbar window if this capability has been requested through the of_SetStatusBar function.

This event extends the w_master pfc_PostOpen event.

Usage

This event executes after the window opens.

pfc_PreToolbar

Description

Populates toolbar attributes.

Syntax

instancename.EVENT **pfc_PreToolbar** (*attributes*)

Argument	Description
instancename	Instance name of w_frame
attributes	N_cst_toolbarattrib variable into which the event places toolbar attributes (passed by reference)

Usage

The pfc_Toolbars event calls this event before displaying the w_toolbars dialog box.

pfc_TileHorizontal

Description	Tiles sheets horizontally.
Return value	Integer. Returns 1 if the event succeeds and -1 if an error occurs.
Usage	This event executes when the user selects Window >Tile Horizontal from the menu bar of a menu descended from the PFC m_master menu.

pfc_TileVertical

Description	Tiles open sheets vertically.
Return value	Integer. Returns 1 if the event succeeds and -1 if an error occurs.
Usage	This event executes when the user selects Window>Tile Vertical from the menu bar of a menu descended from the PFC m_master menu.

pfc_Toolbars

Description	Displays the w_toolbars dialog box.
Return value	Integer. Returns 1 if the event succeeds, 0 if the user cancels out of the w_toolbars dialog box, and -1 if an error occurs.
Usage	This event executes when the user selects View>Toolbars from the menu bar of a menu that descends from the PFC m_master menu.

pfc_UndoArrange

Description	Calls the n_cst_winsrv_sheetmanager pfc_UndoArrange event.
Return value	Integer. Returns the number of sheets undone if the event succeeds and -1 if an error occurs.
Usage	This event executes when the user selects Window>Undo Arrange from the menu bar of a menu descended from the PFC m_master menu.

Resize

Description	Positions the status bar when the user resizes the window.

Usage

This event executes when the user resizes the frame window.

Show

Description

Calls the n_cst_winsrv_statusbar pfc_Show event.

Usage

This event executes when the window displays after being hidden.

Functions

W_frame includes precoded object functions:

of_SetSheetManager
of_SetStatusBar

of_SetSheetManager

Description

Creates or destroys an instance of n_cst_winsrv_sheetmanager, which provides sheet management services.

Access

Public

Syntax

windowname.**of_SetSheetManager** (*boolean*)

Argument	Description
windowname	Instance name of w_frame
boolean	Boolean specifying whether to create (TRUE) or destroy (FALSE) an instance of the n_cst_winsrv_sheetmanager object

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetSheetManager function:

```
of_SetSheetManager(TRUE)
of_SetStatusBar(TRUE)
```

of_SetStatusBar

Description

Creates or destroys an instance of n_cst_winsrv_statusbar, which provides status bar services and opens the w_statusbar at the bottom of the frame.

Access

Public

Syntax

windowname.**of_SetStatusBar** (*boolean*)

Argument	Description
windowname	Instance name of w_frame
boolean	Boolean specifying whether to create (TRUE) or destroy (FALSE) an instance of the n_cst_winsrv_statusbar object

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

On UNIX
The status bar service is not available on UNIX.

Examples

This example calls the of_SetStatusBar function:

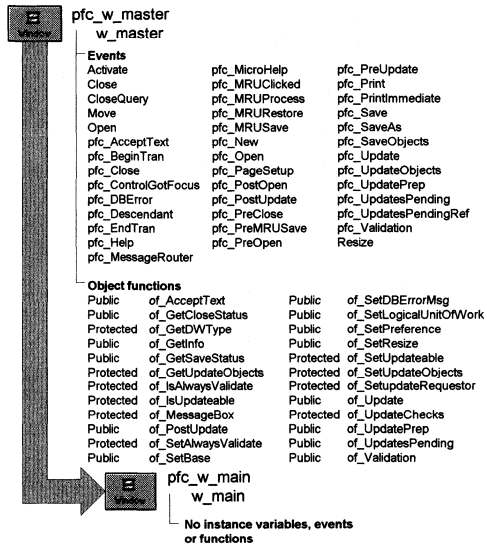
```
of_SetStatusBar (TRUE)
```

w_main

Description

Ancestor main window for all SDI applications.

Ancestry



Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use w_main as the ancestor window for all single document interface (SDI) applications.

See also

w_master

w_master

Description

W_master is the ancestor for all PFC windows. It includes instance variables, events, and functions that are accessed by many PFC objects.

W_master is a self-updating object.

Ancestry

pfc_w_master
w_master

Events

Activate	pfc_MicroHelp	pfc_PreUpdate
Close	pfc_MRUClicked	pfc_Print
CloseQuery	pfc_MRUProcess	pfc_PrintImmediate
Move	pfc_MRURestore	pfc_Save
Open	pfc_MRUSave	pfc_SaveAs
pfc_AcceptText	pfc_New	pfc_SaveObjects
pfc_BeginTran	pfc_Open	pfc_Update
pfc_Close	pfc_PageSetup	pfc_UpdateObjects
pfc_ControlGotFocus	pfc_PostOpen	pfc_UpdatePrep
pfc_DBError	pfc_PostUpdate	pfc_UpdatesPending
pfc_Descendant	pfc_PreClose	pfc_UpdatesPendingRef
pfc_EndTran	pfc_PreMRUSave	pfc_Validation
pfc_Help	pfc_PreOpen	Resize
pfc_MessageRouter		

Object functions

Public	of_AcceptText	Public	of_SetDBErrorMsg
Public	of_GetCloseStatus	Public	of_SetLogicalUnitOfWork
Protected	of_GetDWType	Public	of_SetPreference
Public	of_GetInfo	Public	of_SetResize
Public	of_GetSaveStatus	Protected	of_SetUpdateable
Protected	of_GetUpdateObjects	Protected	of_SetUpdateObjects
Protected	of_IsAlwaysValidate	Protected	of_SetupdateRequestor
Protected	of_IsUpdateable	Public	of_Update
Protected	of_MessageBox	Protected	of_UpdateChecks
Public	of_PostUpdate	Public	of_UpdatePrep
Protected	of_SetAlwaysValidate	Public	of_UpdatesPending
Public	of_SetBase	Public	of_Validation

Library

PFCWNSRV.PBL
PFEWNSRV.PBL

Object relationships

n_cst_appmanager
n_cst_dberrorattrib
n_cst_dwsrv_linkage
n_cst_luw
n_cst_winsrv
n_cst_winsrv_preference
n_cst_resize
u_dw

Usage

When writing a PFC-based application, all windows must descend from w_master.

W_master is the ancestor of all PFC windows. When you add instance variables, events, and functions to w_master, they are immediately available in all PFC windows.

Descendants

w_child
w_frame
w_main
w_popup

w_response
w_sheet

See also

n_cst_appmanager
n_cst_winsrv
n_cst_winsrv_preference
n_cst_resize

Instance variables

W_master includes instance variables:

Instance variable	Description	Data type	Access	Usage
CONTINUE_ACTION	Constant set to 1	Integer	Public	Internal
FAILURE	Constant set to -1	Integer	Public	Internal
ia_helptypeid	Controls whether PFC calls the ShowHelp function by topic, keyword, or index	Any	Protected	Set programmatically
ib_alwaysvalidate	Controls whether the save process includes all objects in the validation process	Boolean	Protected	Set with of_SetAlwaysValidate (default is FALSE)
ib_closestatus	Indicates whether the window is closing	Boolean	Protected	Internal
ib_disableclosequery	Indicates whether CloseQuery processing is enabled	Boolean	Protected	Internal
ib_isupdateable	Indicates whether the window can be updated	Boolean	Protected	Set with of_SetUpdateable (default is TRUE)
ib_savestatus	Controls whether message display during the save process	Boolean	Protected	Internal
idw_active	Current DataWindow	U_dw	Protected	Internal
inv_base	Reference variable for basic window services	n_cst_winsrv	Public	Set with of_SetBase
inv_dberrorattrib	Contains database error information	n_cst_ dberrorattrib	Protected	Set with of_SetDBErrorMsg

Instance variable	Description	Data type	Access	Usage
inv_luw	Reference variable for logical unit of work service	n_cst_luw	Protected	Set with of_ SetLogicalUnitOfWork
inv_preference	Reference variable for window preference service	n_cst_winsrv_ preference	Public	Set with of_SetPreference
inv_resize	Reference variable for resize service	n_cst_resize	Public	Set with of_SetResize
ipo_pendingupdates[ ]	Default list of objects to be updated	PowerObject	Protected	Internal
ipo_updateobjects[ ]	Customized list of objects to be updated	PowerObject	Protected	Internal
ipo_ tempupdateobjects[ ]	One-time list of objects to be updated	PowerObject	Protected	Internal
ipo_updaterequestor	Owner of the save process	PowerObject	Protected	Internal
NO_ACTION	Constant set to 0	Integer	Public	Internal
PREVENT_ACTION	Constant set to 0	Integer	Public	Internal
SUCCESS	Constant set to 1	Integer	Public	Internal

Events

W_master includes precoded events:

Activate	pfc_Open
Close	pfc_PageSetup
CloseQuery	pfc_PostOpen
Move	pfc_PostUpdate
Open	pfc_PreClose
pfc_AcceptText	pfc_PreMRUSave
pfc_BeginTran	pfc_PreOpen
pfc_Close	pfc_PreUpdate
pfc_ControlGotFocus	pfc_Print
pfc_DBError	pfc_PrintImmediate
pfc_Descendant	pfc_Save
pfc_EndTran	pfc_SaveAs
pfc_Help	pfc_SaveObjects
pfc_MessageRouter	pfc_Update
pfc_MicroHelp	pfc_UpdateObjects

pfc_MRUClicked	pfc_UpdatePrep
pfc_MRUProcess	pfc_UpdatesPending
pfc_MRURestore	pfc_UpdatesPendingRef
pfc_MRUSave	pfc_Validation
pfc_New	Resize

Activate

Description

Calls the n_cst_mru pfc_MRURestore event to restore changed menu items as needed.

Usage

This event executes when the window becomes active.

Close

Description

Stores preference information and destroys any existing window service objects.

Usage

This event executes when a window closes.

CloseQuery

Description

Accesses all DataWindows, checks for validation errors and, if changes are pending, prompts the user to save the changes.

Usage

This event executes when a window closes, just before the Close event. It calls one or more of the following:

- Pfc_PreClose
- Of_UpdateChecks
- Pfc_Save

Move

Description

Stores the window position and size.

Usage

This event executes when the user moves a window.

Open

Description

Triggers the pfc_PreOpen event and posts the pfc_PostOpen event. If the preference and MRU services are enabled, this event also restores window settings and MRU information.

Usage

This event executes when the window opens. It's best to add application-specific code to either the pfc_PreOpen or pfc_PostOpen events.

pfc_AcceptText

Description

Calls the n_cst_luw of_AcceptText function, which in turn calls the of_AcceptText function for all controls that implement it.

Syntax

instancename.EVENT **pfc_AcceptText** (*controlarray, focusonerror*)

Argument	Description
instancename	Instance name of the w_master-based descendant
controlarray	PowerObject array containing the controls for which to accept text. From within the pfc_AcceptText event, access this value through the *apo_control* argument
focusonerror	Boolean specifying whether to set focus on a DataWindow column with errors. From within the pfc_AcceptText event, access this value through the *ab_focusonerror* argument

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is called by the of_UpdateChecks function.

pfc_BeginTran

Description

Empty user event that you extend to perform start-of-transaction processing.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is called by the pfc_Save event.

pfc_Close

Description

Closes the window.

Usage

Call this event to close a window.

pfc_ControlGotFocus

Description

Triggered when a PFC visual control gets focus. This event keeps track of the current control.

Syntax

instancename.EVENT **pfc_ControlGotFocus** (*control*)

Argument	Description
instancename	Instance name of the w_master-based descendant
control	DragObject variable containing the control that just received focus

Usage

This event is called by PFC visual controls when they get focus.

Use this event to add application-specific functionality to be performed when a visual control gets focus.

If the control getting focus is a DataWindow inherited from u_dw, this event sets the idw_active instance variable.

pfc_DBError

Description

Displays a database error encountered during the pfc_Save process.

Usage

This event is called by pfc_Save if an update error occurs.

pfc_Descendant

Description

PFC events and functions call this event to determine if the window is inherited from w_master.

Return value

Boolean. Always returns TRUE.

Usage

Internal.

pfc_EndTran

Description

Empty user event that you extend to initiate end-of-transaction processing, such as COMMIT or ROLLBACK.

Syntax

instancename.EVENT **pfc_EndTran** (*savecode*)

Argument	Description
instancename	Instance name of the w_master-based descendant
savecode	Integer specifying the return code from the pfc_Update function. From within the pfc_EndTran event, access this value through the *ai_update_results* argument

Return value

Integer. Return 1 if the end transaction process succeeds and other application-specific values as appropriate.

Usage

This event is called by pfc_Save. The value passed in *savecode* indicates whether previous updates succeeded or failed. Based on *savecode*, you extend this event to add COMMIT, ROLLBACK, or other end-of-transaction processing.

Examples

This example extends the pfc_EndTran event to COMMIT all open transactions. This example returns 0 if it succeeds and -1 if a COMMIT fails:

```
n_tr  ltr_trans[ ]
Integer  li_trans, li_count
Long  ll_return

li_trans = &
  gnv_app.inv_trregistration.of_GetRegistered &
    (ltr_trans)
IF li_trans = 0 THEN Return 0
For li_count = 1 to li_trans
  ll_return = ltr_trans[li_count].of_Commit()
  IF ll_return < 0 THEN Return -1
Next
Return 0
```

pfc_Help

Description

Calls the ShowHelp function by index, topic or keyword, based on the ia_helptypeid instance variable.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is triggered when the window has focus and the user selects Help>Help Topics from the menu bar of a menu that descends from the PFC m_master menu. You can also trigger it directly (from a Help CommandButton, for example).

Call the n_cst_appmanager of_SetHelpfile function to specify the application's Help filename.

pfc_MessageRouter

Description

Triggers the passed user event. This event first attempts to trigger the passed user event on the window, then the current control. If the user event is not found, it then attempts to trigger the passed user event on the last active DataWindow control.

Syntax

instancename.EVENT **pfc_MessageRouter** (*userevent*)

Argument	Description
instancename	Instance name of the w_master-based descendant
userevent	String specifying the event to trigger. From within the pfc_MessageRouter event, access this value through the *as_message* argument

Return value

Integer. Returns 1 if the event succeeds, 0 if no objects contain the user event, and -1 if an error occurs.

Usage

This event is triggered when the user selects a menu choice that calls the of_SendMessage menu function.

All PFC visual controls, including u_dw, include events that work with the message router.

pfc_MicroHelp

Description

Ancestor event to which descendants add code that displays MicroHelp.

Syntax

*instancename.Event***pfc_MicroHelp** (*microhelp*)

Argument	Description
instancename	Instance name of the w_master-based descendant
microhelp	String to be displayed in the status bar

Usage

This event is called by u_dw ItemFocusChanged event. It is extended in w_frame and w_sheet.

pfc_MRUClicked

Description

Checks for the existence of the specified menu item in the MRU DataStore and calls the pfc_MRUProcess event if the menu item is found.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

Internal.

pfc_MRUProcess

Description

Empty user event that you extend to open the appropriate window based on the clicked MRU menu item.

Syntax

instancename.EVENT **pfc_MRUProcess** (*row*)

Argument	Description
instancename	Instance name of the w_master-based descendant
row	Integer specifying the MRU DataStore row that corresponds to the clicked MRU menu item. From within the pfc_MRUProcess event, access this value through the *ai_row* argument

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

You extend this event to open windows (SDI) or sheets (MDI) with or without passing parameters.

The pfc_w_master ancestor contains sample code for this event.

Examples

This example shows code you might add to the pfc_MRUProcess event:

```
window  lw_frame, lw_window
n_cst_menu  lnv_menu
n_cst_mruattrib  lnv_mruattrib

// check parameters
if isnull(ai_row) then
   return -1
end if
if not isvalid(gnv_app.inv_mru) then
   return -1
end if
// retrieve row from datastore and do work with it
gnv_app.inv_mru.of_getmruitem(ai_row, lnv_mruattrib)
```

```
// get the mdi frame if necessary
lnv_menu.of_getmdiframe(this.menuid, lw_frame)
//example opens
// SDI Process
//open(lw_window, lnv_mruattrib.is_classname)
//openwithparm(lw_window,  &
//   lnv_mruattrib.is_menuitemkey,  &
//     lnv_mruattrib.is_classname)
// MDI Process
//opensheet(lw_window, lnv_mruattrib.is_classname,  &
//   lw_frame)
//opensheet(lw_window, lnv_mruattrib.is_classname,  &
//  lw_frame, 0, original!)
Return opensheetwithparm(lw_window,  &
  lnv_mruattrib.is_menuitemkey, &
    lnv_mruattrib.is_classname, lw_frame, 0,  &
      original! )
```

pfc_MRURestore

Description Restores MRU settings.

Usage The Open event calls this event.

pfc_MRUSave

Description Saves MRU key information.

Return value Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage This event is called by the Close event.

pfc_New

Description Empty user event into which you place code that performs processing to add a new entity to the current window, window control, or DataWindow.

Usage Use this event for application- and window-specific new entity processing.

This event is triggered when the user selects File>New from the menu bar of a menu that descends from the PFC m_master menu.

Examples

This example, which might be found in a descendent sheet or main window, inserts a new row into the active DataWindow when the user selects File>New from the menu bar:

```
idw_active.InsertRow(0)
```

pfc_Open

Description

Empty user event into which you place code that performs processing to open a new entity (typically a file) in the current window, window control, or DataWindow.

Usage

Use this event for application- and window-specific open processing.

This event is triggered when the user selects File>Open from the menu bar of a menu that descends from the PFC m_master menu.

Examples

This example, which might be found in a descendent sheet or main window, displays the Open dialog box when the user selects File>Open from the menu bar. Is_fullname and is_filename are instance variables:

```
Integer  li_fileid

SetPointer (HourGlass!)
IF GetFileOpenName &
  ("Open", is_fullname, is_filename &
    "txt", "Text Files (*.txt),*.txt) < 1 THEN
      Return
END IF

// Open the new file and put results into the MLE.
li_fileid = FileOpen(is_fullname, StreamMode!)
FileRead (li_fileid, mle_notepad.text)
FileClose (li_fileid)
...
```

pfc_PageSetup

Description

Empty user allowing you to add code to display a Page Setup dialog box and print multiple DataWindows.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

Call this event to synch page setup properties among multiple DataWindows.

Examples

This example synchs the paper source for three DataWindows:

```
Integer  li_rc
s_pagesetupattrib  lstr_pagesetup

li_rc = dw_1.event pfc_PageSetupDlg(lstr_pagesetup)
IF li_rc = 1 THEN
  dw_2.object.datawindow.print.paper.source = &
    lstr_pagesetup.i_papersource
  dw_3.object.datawindow.print.paper.source = &
    lstr_pagesetup.i_papersource
END IF
...
```

pfc_PostOpen

Description

Empty user event into which you place code that performs processing just after a window opens.

Usage

Use this event for application- and window-specific post-open processing. This event executes just after a window opens.

Examples

This example calls the Retrieve function in a window's pfc_PostOpen event:

```
Long  ll_return

ll_return = dw_employee.EVENT pfc_Retrieve()
IF ll_return = -1 THEN
  MessageBox("Retrieve", "Retrieval error")
ELSE
  gnv_app.of_getframe().SetMicroHelp &
    (String(ll_return) + " rows retrieved")
END IF
```

pfc_PostUpdate

Description

Calls the n_cst_luw of_PostUpdate function, which in turn calls the of_PostUpdate function for all controls that implement it.

Syntax

instancename.EVENT **pfc_PostUpdate** (*controlarray*)

Argument	Description
instancename	Instance name of the w_master-based descendant
controlarray	PowerObject array containing the controls for which to perform post update processing. From within the pfc_AcceptText event, access this value through the *apo_control* argument

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

The pfc_Save user event calls this event after calling the pfc_DBError user event.

You can extend this event to perform additional post-update processing.

pfc_PreClose

Description

Empty user event that you extend to perform processing just before a window closes.

Return value

Integer. Return 1 if the pre-close process succeeds and -1 to prevent the window from closing.

Usage

Use this event for application- and window-specific pre-close processing.

This event executes just before CloseQuery processing.

Examples

This example uses the pfc_PreClose function to write to a log file:

```
IF inv_filesrv.of_FileWrite &
  (is_logfile, "Window: " + this.title &
    + " closing at " + String(Today()) &
    + String(Now()),TRUE) = 1 THEN
  Return 1
ELSE
  Return -1
END IF
```

pfc_PreMRUSave

Description

Empty user event that you extend to create a key that you can later access when opening windows.

Syntax

instancename.EVENT **pfc_PreMRUSave** (*mruattrib*)

Argument	Description
instancename	Instance name of the w_master-based descendant
mruattrib	N_cst_mruattrib into which the event places window information

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

You must populate all three values in the n_cst_mruattrib structure object.

Examples

This example shows code you might add to the pfc_PreMRUSave event:

```
anv_mruattrib.is_classname = this.classname()
anv_mruattrib.is_menuitemname = this.title
anv_mruattrib.is_menuitemkey = this.classname()
```

pfc_PreOpen

Description

Empty user event into which you place code that performs processing just before a window opens.

Usage

Use this event for application- and window-specific pre-open processing.

This event executes just before a window opens.

Examples

This example connects to the database in the frame window's pfc_PreOpen event:

```
String  ls_file, ls_section
Long  ll_return

ls_file = "c:\pbapp\pbapp.ini"
ls_section = "PBAppDB"

itr_pbapp.of_Init(ls_file, ls_section)
ll_return = itr_pbapp.of_Connect()
IF ll_return <> 0 THEN
  itr_pbapp.of_Rollback()
  MessageBox("PBApp", "Connect error")
END IF
```

pfc_PreUpdate

Description

Empty user event into which you place pre-save processing.

Return value

Integer. Return 1 if the pre-save processing succeeds and -1 if it does not.

Usage

Extend this event to perform application-specific pre-save processing.

The pfc_Save user event calls this event before performing any save processing. If you return -1, save processing terminates.

Examples

This example adds code to the pfc_PreUpdate event to check for user ID:

```
String  ls_user

ls_user = gnv_app.of_GetUserID()
IF ls_user = "RNielsen" THEN
  Return -1
ELSE
  Return 1
END IF
```

pfc_Print

Description

Empty user event into which you place code that prints windows or DataWindows, first displaying the Print dialog box.

Usage

Extend this event to print application- and window-specific information. For example, you might print a single DataWindow. Alternatively, you might call the DataWindow report service's of_CreateComposite function to create a composite report of all the window's DataWindows.

This event is triggered when the user selects File>Print from the menu bar of a menu that descends from the PFC m_master menu.

Examples

This example synchs the copies property of all DataWindows on the window:

```
Integer  li_rc
s_printdlgattrib  lstr_printdlg

li_rc = dw_1.Event pfc_PrintDlg(lstr_printdlg)
IF li_rc = 1 THEN
  dw_2.object.datawindow.print.copies = &
    lstr_printdlg.l_copies
  dw_3.object.datawindow.print.copies = &
    lstr_printdlg.l_copies
END IF
```

pfc_PrintImmediate

Description

Empty user event into which you place code that prints windows or DataWindows without displaying the Print dialog box.

Usage

Extend this event to print application- and window-specific information.

This event is triggered when the user selects File>Print Immediate from the menu bar of a menu that descends from the PFC m_master menu.

Examples

This example prints the dw_employee DataWindow:

```
IF dw_employee.Event pfc_PrintImmediate( ) <> 1 THEN
  Return -1
END IF
```

pfc_Save

Description

Calls w_master user events and functions to save changes for all self-updating objects in the window.

Return value

Integer. Returns values as follows:

- **1** All save processes succeeded
- **0** No pending updates
- **-1** Accept text error
- **-2** Updates pending error
- **-3** Validation error
- **-4** Pre-update process failed
- **-5** Pfc_BeginTran failed
- **-6** Pfc_Update failed
- **-7** Pfc_EndTran failed
- **-8** Post-update process failed
- **-9** Update prep process failed

Usage

The CloseQuery event calls this user event automatically to check for pending updates and initiate the save process. You can call it from your application to save changes, as needed.

FOR INFO For more on the save process, see n_cst_luw on page 1139.

pfc_SaveAs

Description

Empty user event allowing you to save all or part of a window.

Usage

This event is triggered when the user selects File>Save As from the menu bar of a menu that descends from the PFC m_master menu.

Examples

This example displays the Save As dialog box:

```
String is_docname, is_named
Integer li_value

li_value = GetFileSaveName("Save As", &
  is_docname, is_named, "TXT", &
    "Text Files (*.TXT),*.TXT," + &
      " Doc Files (*.DOC), *.DOC")
...
```

pfc_SaveObjects

Description

Performs a save on the specified controls.

Syntax

instancename.EVENT **pfc_SaveObjects** (*controlarray*)

Argument	Description
instancename	Instance name of the w_master-based descendant
controlarray	PowerObject array containing the controls for which to perform save processing. From within the pfc_SaveObjects event, access this value through the *apo_control* argument

Return value

Integer. Returns values as follows:

- **1** All save processes succeeded
- **0** No pending updates
- **-1** Accept text error
- **-2** Updates pending error
- **-3** Validation error
- **-4** Pre-update process failed
- **-5** Pfc_BeginTran failed
- **-6** Pfc_Update failed

- **-7** Pfc_EndTran failed
- **-8** Post-update process failed
- **-9** Update prep process failed

Usage

Call the of_SetUpdateObjects function to establish the array of controls to be updated.

Examples

This example calls the pfc_SaveObjects event:

```
PowerObject  lpo_save[ ]
Integer  li_return

lpo_save[1] = dw_1
this.of_SetUpdateObjects(lpo_save)
li_return = this.EVENT pfc_SaveObjects(lpo_save)
...
```

pfc_Update

Description

Calls the pfc_UpdateObjects event.

Syntax

instancename.EVENT **pfc_Update** (*controlarray*)

Argument	Description
instancename	Instance name of the w_master-based descendant
controlarray	PowerObject array containing the controls to be updated. From within the pfc_Update event, access this value through the *apo_control* argument

Return value

Integer. Returns 1 if the event succeeds and -1 if one or more DataWindow update errors occur.

Usage

The pfc_Save event calls this event.

pfc_UpdateObjects

Description

Calls the n_cst_luw of_Update function, which in turn calls the of_Update function for all controls that implement it.

Syntax

instancename.EVENT **pfc_UpdateObjects** (*controls, accepttext, resetflags*)

Argument	Description
instancename	Instance name of the w_master-based descendant
controls	PowerObject array containing the objects to update. This argument is accessed through the *apo_control* argument
accepttext	Boolean specifying whether n_cst_luw should automatically perform an AcceptText before performing the Update (TRUE) or not (FALSE)
resetflags	Boolean specifying whether n_cst_luw should automatically reset DataWindow update flags (TRUE) or not (FALSE)

Return value

Integer. Returns 1 if the event succeeds, 0 if no action was taken, and -1 if one or more update errors occur.

Usage

The of_Update function and the pfc_Update event call this event.

pfc_UpdatePrep

Description

Empty user event to which you can add code that prepares for update.

Syntax

instancename.EVENT **pfc_UpdatePrep** (*controls*)

Argument	Description
instancename	Instance name of the w_master-based descendant
controls	PowerObject array containing the objects to update. This argument is accessed through the *apo_control* argument

Return value

Long. Return 1 if the update preparation succeeds and -1 to halt the update process.

Usage

The of_UpdatePrep function calls this event.

pfc_UpdatesPending

Description

Calls the pfc_UpdatesPendingRef event.

When CloseQuery calls this user event, validation messages are suppressed.

Syntax

instancename.EVENT **pfc_UpdatesPending** (*controlarray*)

Argument	Description
instancename	Instance name of the w_master-based descendant

Argument	Description
controlarray	PowerObject array containing controls to be tested for pending updates

Return value

Integer. Returns values as follows:

- 1 Pending updates were found
- 0 No pending updates
- -1 AcceptText failed

Usage

The CloseQuery event calls this event to determine if there are pending updates.

pfc_UpdatesPendingRef

Description

Calls the n_cst_luw of_UpdatesPending function, which in turn calls the of_UpdatesPending function for all controls that implement it.

Syntax

*instancename.Event***pfc_UpdatesPendingRef** (*controls, pending*)

Argument	Description
instancename	Instance name of the w_master-based descendant
controls	PowerObject array containing the objects to update. This argument is accessed through the *apo_control* argument
pending	PowerObject array to contain objects with pending updates. This argument is accessed through the *apo_pending* argument (passed by reference)

Return value

Integer. Returns values, as follows:

- 1 Pending updates were found
- 0 No pending updates
- -1 AcceptText failed

Usage

The pfc_UpdatesPending event calls this event to determine if there are pending updates.

pfc_Validation

Description

Calls the n_cst_luw of_Validation function, which in turn calls the of_Validation function for all controls that implement it.

Syntax

instancename.EVENT **pfc_Validation** (*controlarray*)

Argument	Description
instancename	Instance name of the w_master-based descendant
controlarray	PowerObject control array containing controls to be validate

Return value

Integer. Returns 1 if there are no validation errors and -1 if a validation error occurs.

Usage

The of_Validation and of_UpdateChecks functions call this event to determine if there are validation errors.

Resize

Description

Triggers automatic resize processing, if enabled in n_cst_resize.

Usage

This event executes when the user resizes the window.

Functions

W_master includes precoded object functions:

of_AcceptText	of_SetDBErrorMsg
of_GetCloseStatus	of_SetLogicalUnitOfWork
of_GetDWType	of_SetPreference
of_GetInfo	of_SetResize
of_GetSaveStatus	of_SetUpdateable
of_GetUpdateObjects	of_SetUpdateObjects
of_IsAlwaysValidate	of_SetUpdateRequestor
of_IsUpdateable	of_Update
of_MessageBox	of_UpdateChecks
of_PostUpdate	of_UpdatePrep
of_SetAlwaysValidate	of_UpdatesPending
of_SetBase	of_Validation

of_AcceptText

Description

Calls the pfc_AcceptText event.

Access

Public

Syntax

instancename.**of_AcceptText** (*focusonerror*)

Argument	Description
instancename	Instance name of w_master
focusonerror	Boolean indicating whether PFC sets focus to the first item in error when an error occurs

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize accept text processing, extend the pfc_AcceptText event.

Examples

This example is from the n_cst_luw of_AcceptText function:

```
...
If lb_defined Then
   li_rc =  &
      lpo_tocheck.Function Dynamic of_AcceptText &
         (ab_focusonerror)
   If li_rc < 0 Then Return -1

...
```

of_GetCloseStatus

Description

Reports the window's close status (whether the window is closing.)

Access

Public

Syntax

windowname.**of_GetCloseStatus** ( )

Argument	Description
windowname	Instance name of w_master

Return value

Boolean. Returns TRUE if the window is closing and FALSE if it is not.

Usage

PFC does not report validation errors if the window is closing.

Examples

This example calls the of_GetCloseStatus function:

```
IF al_row <> 0 AND &
  (Not lw_parent. of_GetCloseStatus ( )) THEN
  IF IsValid(gnv_app) THEN
    MessageBox(gnv_app.iapp_object.Appname, &
      "Required Value Missing for " &
        + as_colname + " on row " + &
        String(al_row) + &
        '. Please enter a value.', StopSign!)
  ELSE
...
```

of_GetDWType

Description

Indicates whether a DataWindow is a descendant of u_dw and whether it uses the linkage service.

Access

Protected

Syntax

instancename.**of_GetDWType** (*datawindow*)

Argument	Description
instancename	Instance name of w_master
datawindow	DataWindow variable containing the DataWindow to be tested

Return value

Integer. Returns values as follows:

- 0 *Datawindow* is not a descendant of u_dw
- 1 *Datawindow* is a descendant of u_dw
- 2 *Datawindow* is a descendant of u_dw and it is linked
- -1 An error occurred

Usage

Internal.

Examples

This example calls the of_GetDWType function:

```
...
li_dwtype = of_GetDWType (ldw_dw)
If li_dwtype < 0 Then Return -1
...
```

of_GetInfo

Description

Retrieves object information.

Access

Public

Syntax

instancename.**of_GetInfo** (*infoobject*)

Argument	Description
instancename	Instance name of w_master
infoobject	N_cst_infoattrib instance into which the function places information (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_GetInfo function:

```
n_cst_infoattrib  lnv_info

w_emplist.of_GetInfo(lnv_info)
MessageBox("Info",  &
  "Description: " + lnv_info.is_description  &
    + ". Name: " + lnv_info.is_name)
```

of_GetSaveStatus

Description

Reports the window's save status (whether the window is in the process of updating its DataWindows).

Access

Public

Syntax

windowname.**of_GetSaveStatus** ( )

Argument	Description
windowname	Instance name of w_master

Return value

Boolean. Returns TRUE if the window is updating and FALSE if it is not.

Usage

Call this function to determine whether a window is updating its DataWindows.

Examples

This example calls the of_GetSaveStatus function:

```
...
IF lw_parent.TriggerEvent &
```

```
    ("pfc_Descendant") = 1 THEN
      lb_pfcsaveprocess = &
        lw_parent.of_GetSaveStatus()
END IF
...
```

of_GetUpdateObjects

Description — Retrieves the current default array of objects affected by the update process.

Access — Protected

Syntax — *instancename*.**of_GetUpdateObjects** (*objects*)

Argument	Description
instancename	Instance name of w_master
objects	PowerObject array into which the function places objects to be updated (passed by reference)

Return value — Integer. Returns the number of elements in the *objects* array if the function succeeds and -1 if an error occurs.

Examples — This example calls the of_GetUpdateObjects function:

```
PowerObject  lpo_objs[ ]
Integer  li_return, li_count

li_return = this.of_GetUpdateObjects(lpo_objs)
FOR li_count = 1 to li_return
   IF lpo_objs[li_count] = ids_data THEN
      Return 1
   END IF
NEXT
li_return++
lpo_objs[li_return] = ids_data
Return this.of_SetUpdateObjects(lpo_objs)
```

of_IsAlwaysValidate

Description — Reports whether the default save process always performs validation.

Access — Protected

Syntax

instancename.**of_IsAlwaysValidate ( )**

Argument	Description
instancename	Instance name of w_master

Return value

Boolean. Returns TRUE if the default save process always performs validation and FALSE if it does not.

Examples

This example calls the of_IsAlwaysValidate function:

```
IF this.of_IsAlwaysValidate() = TRUE THEN
   MessageBox("Window", "Always validate")
ELSE
   MessageBox("Window", "Sometimes validate")
END IF
```

of_IsUpdateable

Description

Reports whether the window is updatable and should be included in save processing.

Access

Protected

Syntax

instancename.**of_IsUpdateable ( )**

Argument	Description
instancename	Instance name of w_master

Return value

Boolean. Returns TRUE if the window is updatable and FALSE if it is not.

Usage

Internal.

Examples

This example is from the pfc_UpdatesPendingRef event:

```
...
If Not of_IsUpdateable() Then Return NO_UPDATESPENDING
...
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of w_master
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in windows.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('win_error','Save', &
   as_error, StopSign!, Ok!, 1)
```

of_PostUpdate

Description

Calls the pfc_PostUpdate event.

Access

Public

Syntax

instancename.**of_PostUpdate** ( )

Argument	Description
instancename	Instance name of w_master

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize post-update processing, extend the pfc_PostUpdate event.

Examples

This example is from the n_cst_luw of_PostUpdate function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_PostUpdate()
...
```

of_SetAlwaysValidate

Description

Specifies whether the default save process always performs validation.

Access

Protected

Syntax

instancename.**of_SetAlwaysValidate** (*boolean*)

Argument	Description
instancename	Instance name of w_master
boolean	Boolean specifying whether the default save process always perform validation (TRUE) or only performs validation if a control has pending updates (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetAlwaysValidate function:

```
this.of_SetAlwaysValidate(TRUE)
```

of_SetBase

Description

Enables or disables n_cst_winsrv, which provides basic window services.

Access

Public

Syntax

instancename.**of_SetBase** (*boolean*)

Argument	Description
instancename	Instance name of w_master
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) an instance of the n_cst_winsrv object

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_winsrv. This instance is named inv_base.

Because all window services are descendants of n_cst_winsrv (and have n_cst_winsrv functions available to them), use this object when you require basic window services only.

Examples

This example calls the of_SetBase function in a window Open event to enable basic window services:

```
this.of_SetBase (TRUE)
```

of_SetDBErrorMsg

Description

Saves a database update error message for display after save processing completes.

Access

Public

Syntax

windowname.**of_SetDBErrorMsg** (*message*)

Argument	Description
windowname	Instance name of w_master
message	String or n_cst_dberrorattrib containing the error message

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function to save a database error message for display after ROLLBACK processing completes.

The u_dw DBError event populates this instance variable, suppressing the display of a message dialog box. This allows you to roll back changes in the pfc_EndTran event. After calling pfc_EndTran, the pfc_Save process displays this message.

Examples

This example is from the u_dw DBError event:

```
...
If IsValid(lpo_updaterequestor) Then
  lpo_updaterequestor.Dynamic Function  &
    of_SetDBErrorMsg(lnv_dberrorattrib)
Else
...
```

of_SetLogicalUnitOfWork

Description

Enables or disables n_cst_luw, which provides the logical unit of work service.

Access

Public

Syntax

instancename.**of_SetLogicalUnitOfWork** (*boolean*)

Argument	Description
instancename	Instance name of w_master
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) n_cst_luw

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_luw. This instance is named inv_luw. If you do not enable n_cst_luw, w_master enables it automatically.

Examples

This example calls the of_SetLogicalUnitOfWork function:

```
this.of_SetLogicalUnitOfWork(TRUE)
```

of_SetPreference

Description

Enables or disables n_cst_winsrv_preference, the window preference service.

Access

Public

Syntax

instancename.**of_SetPreference** (*boolean*)

Argument	Description
instancename	Instance name of w_master
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) an instance of the n_cst_winsrv_preference object

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_winsrv_preference. This instance is named inv_preference.

Examples

This example calls the of_SetPreference function:

```
this.of_SetPreference(TRUE)
```

of_SetResize

Description

Enables or disables n_cst_resize, the resize service.

Access

Public

Syntax

instancename.**of_SetResize** (*boolean*)

Argument	Description
instancename	Instance name of w_master
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) an instance of the n_cst_resize object

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_resize. This instance is named inv_resize.

Examples

This example calls the of_SetResize function:

```
this.of_SetResize(TRUE)
```

of_SetUpdateable

Description

Specifies whether the window is updatable and should be included in a default save process.

Access

Protected

Syntax

instancename.**of_SetUpdateable** (*boolean*)

Argument	Description
instancename	Instance name of w_master
boolean	Boolean indicating whether the window is updatable

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function to disable default save processing for windows that are not updatable.

Examples

This example calls the of_SetUpdateable function:

```
this.of_SetUpdateable(FALSE)
```

of_SetUpdateObjects

Description

Sets a new default array containing objects for which updates are attempted.

Access

Protected

Syntax

instancename.**of_SetUpdateObjects** (*requestor*)

Argument	Description
instancename	Instance name of u_dw
requestor	PowerObject array containing the object to be updated

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function to customize the objects updated by the save process. You can even add other windows to the save process.

Examples

This example calls the of_SetUpdateObjects function:

```
PowerObject  lpo_objs[ ]
Integer  li_count

lpo_objs = this.control
```

```
li_count = UpperBound(lpo_objs)
li_count++
// Update w_other as well as this window
lpo_objs[li_count] = w_other
Return this.of_SetUpdateObjects(lpo_objs)
```

of_SetUpdateRequestor

Description

Creates a reference to the object requesting an update within a logical unit of work.

Access

Protected

Syntax

instancename.**of_SetUpdateRequestor** (*requestor*)

Argument	Description
instancename	Instance name of w_master
requestor	PowerObject containing the object requesting the update

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the of_SetLogicalUnitOfWork function:

```
...
inv_luw = create n_cst_luw
inv_luw.of_SetRequestor (this)
inv_luw.of_SetUpdateRequestor (this)
...
```

of_Update

Description

Calls the pfc_UpdateObjects event.

Access

Public

Syntax

instancename.**of_Update** (*accept*, *resetflag* {, *requestor* })

Argument	Description
instancename	Instance name of w_master
accept	Boolean indicating whether the Update function performs an AcceptText before saving rows to the database

Argument	Description
resetflag	Boolean indicating whether the Update function resets the update flags
requestor (optional)	PowerObject containing the requestor object

Return value

Integer. Returns 1 if the update succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize update processing, extend the pfc_Update event.

You typically call the pfc_Save event instead of this function.

Examples

This example is from the n_cst_luw of_Update function:

```
...
If lb_defined Then
  li_rc = lpo_tocheck.Function Dynamic of_Update &
    (ab_accepttext,  ab_resetflag, &
      lpo_updaterequestor)
  If li_rc < 0 Then Return -1
  Continue
End If
...
```

of_UpdateChecks

Description

Determines whether edits are pending and whether there are any validation errors.

Access

Protected

Syntax

instancename.**of_UpdateChecks** ({ *controls* })

Argument	Description
instancename	Instance name of w_master
controls (optional)	PowerObject array containing the objects to be checked

Return value

Integer. Returns values as follows:

- **1** Updates are pending and there are no validation errors

- **0** No updates pending
- **-1** AcceptText error
- **-2** UpdatesPending error
- **-3** Validation error

Usage

Internal.

Examples

This example is from the pfc_Save event:

```
...
li_rc = of_UpdateChecks(lpo_updatearray)
   If li_rc <= 0 Then
...
```

of_UpdatePrep

Description

Calls the pfc_UpdatePrep event, which allows you to code additional update preparation logic.

Access

Public

Syntax

instancename.**of_UpdatePrep ( )**

Argument	Description
instancename	Instance name of w_master

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize update preparation processing, extend the pfc_UpdatePrep event.

Examples

This example is from the n_cst_luw of_UpdatePrep function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_UpdatePrep()
  If li_rc < 0 Then Return -1
  Continue
End If
...
```

of_UpdatesPending

Description

Calls the pfc_UpdatesPendingRef event.

Access

Public

Syntax

instancename.**of_UpdatesPending** ( )

Argument	Description
instancename	Instance name of w_master

Return value

Integer. Returns values as follows:

- **1** Updates are pending
- **0** No updates pending
- **-1** An error occurred

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize pending updates processing, extend the pfc_UpdatesPending event.

Examples

This example calls the of_UpdatesPending function:

```
...
If lb_defined Then
   la_rc = lpo_tocheck.Dynamic of_UpdatesPending()
...
```

of_Validation

Description

Calls the pfc_Validation event.

Access

Public

Syntax

instancename.**of_Validation** ( )

Argument	Description
instancename	Instance name of w_master

Return value

Integer. Returns 1 if the function succeeds and -1 if a validation error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize validation processing, extend the pfc_Validation event.

Examples

This example calls the of_Validation function:

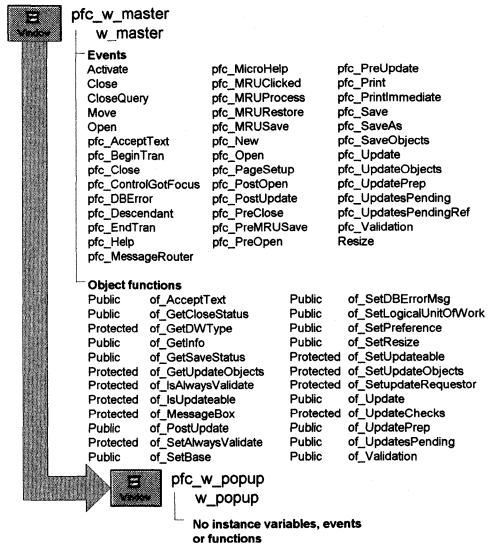
```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_Validation()
...
```

w_popup

Description

Ancestor for all PowerBuilder Foundation Class Library popup windows.

Ancestry



Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use w_popup as the ancestor window for all popup windows.

See also

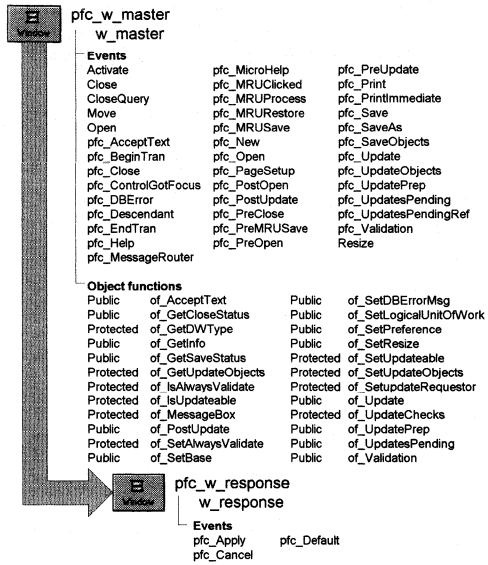
w_master

w_response

Description

Ancestor for all PFC response windows.

Ancestry



Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use w_response as the ancestor window for all response windows.

Add OK, Cancel, and window-specific CommandButtons to your w_response descendant. Then add code to the CommandButtons' Clicked events to trigger the appropriate w_response user event. Typically, Cancel triggers pfc_Cancel, Apply triggers pfc_Apply, and OK triggers pfc_Default.

See also

w_master

Events

W_response includes predefined events:

pfc_Apply	pfc_Default
pfc_Cancel	

pfc_Apply

Description

Empty user event to which you add code that applies dialog box specifications to the associated sheet or window.

Usage

Add code to this event to apply specifications to the associated sheet or window. Trigger this user event from a cb_apply.Clicked event script.

Examples

This example shows code you might add to pfc_Apply:

```
Integer  li_cnt, li_arraysize, li_rc
Boolean  lb
ToolbarAlignment  lt
String  ls

// Set toolbar settings
li_arraysize = UpperBound(istr_toolbar[])
FOR li_cnt = 1 to li_arraysize
  istr_toolbar[li_cnt].w_owner.SetToolbar &
    (istr_toolbar[li_cnt].i_barindex, &
    istr_toolbar[li_cnt].b_visible, &
    istr_toolbar[li_cnt].e_alignment, &
    istr_toolbar[li_cnt].s_title)
  istr_toolbar[li_cnt].w_owner.GetToolbar &
    (istr_toolbar[li_cnt].i_barindex, lb, lt, ls)
NEXT
// Set application toolbar settings
iapp_object.ToolbarText = cbx_text.checked
iapp_object.ToolbarTips = cbx_tips.checked
// Disable Apply button. Make OK the default
cb_apply.Enabled = False
cb_apply.Default = False
cb_cancel.Default = False
cb_ok.Default = True
```

pfc_Cancel

Description

Empty user event to which you add code that closes the window without accepting changes.

Usage

Add code to this user event to close the window without accepting changes. Trigger this user event from the cb_cancel.Clicked event script.

Examples

This example closes the window and passes a return value of 0:

```
CloseWithReturn(This, 0)
```

pfc_Default

Description

Event to apply changes and close a response window.

Usage

Add code to this event to apply changes and close a response window. It is meant to be triggered by the default CommandButton (typically cb_ok), but it could be triggered by Cancel, Close, or some other CommandButton.

Examples

This example shows code you might add to the pfc_Default:

```
SetPointer(HourGlass!)

SQLCA.dbms = "ODBC"
SQLCA.database = "expense"
SQLCA.logid = sle_userid.text
SQLCA.logpass = sle_password.text
SQLCA.dbparm = "Connectstring='DSN=" + &
SQLCA.database + ";UID=" + &
SQLCA.logid + ";PWD=" + SQLCA.logpass + "'"
IF SQLCA.of_Connect() <> 0 THEN
  MessageBox("Login Error", SQLCA.SQLErrText)
  CloseWithReturn(This, 0)
ELSE
  CloseWithReturn(This, 1)
END IF
```

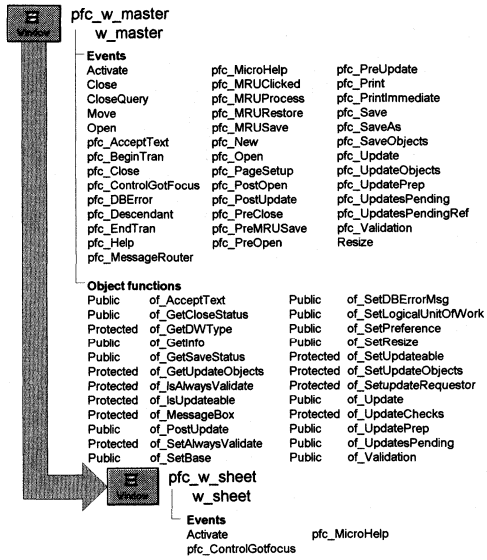
w_sheet

Description

Ancestor for all PowerBuilder Foundation Class Library sheet windows.

This window contains automatic MicroHelp display capability.

Ancestry



Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

n_cst_string

Usage

Use this window as the ancestor for all of your application's sheet windows.

If you want to display MicroHelp for controls within a sheet, associate the MicroHelp text with each control's tag value. The control's tag value must specify MICROHELP=*microhelptext*. When the control gets focus, the pfc_ControlGotFocus event displays the tag text in the status bar.

See also

w_frame
w_master

Events

W_sheet includes predefined events:

Activate
pfc_MicroHelp
pfc_ControlGotFocus

Activate

Description

Sets MicroHelp to the control with focus.

Usage

This event executes whenever the window receives focus.

pfc_ControlGotFocus

Description

Displays the MicroHelp stored in the control's tag value.

This event extends the ancestor pfc_ControlGotFocus event.

Syntax

instancename.EVENT **pfc_ControlGotFocus** (*control*)

Argument	Description
instancename	Instance name of w_sheet-based descendant
control	DragObject variable containing the control that just got focus

Usage

This event is called by PFC visual controls when they get focus.

The control's tag value must specify MICROHELP=*microhelptext*.

pfc_MicroHelp

Description

Updates MicroHelp by calling the frame's pfc_MicroHelp event.

This event extends the ancestor pfc_MicroHelp event.

Syntax

instancename.EVENT **pfc_MicroHelp** (*microhelp*)

Argument	Description
instancename	Instance name of w_sheet-based descendant
microhelp	String to be displayed in the status bar

Usage

This event executes whenever the sheet or a control on the sheet receives focus.

CHAPTER 3

Menus

About this chapter

This chapter describes the menus in the PowerBuilder Foundation Class Library (PFC).

Contents

The menus are listed in alphabetical order. Each object's discussion includes alphabetical listings of instance variables, events, and object functions.

m_dw

Description

Popup menu displayed when the user releases the right mouse button over a u_dw based DataWindow control.

Ancestry

Library

PFCMAIN.PBL
PFEMAIN.PBL

Menu bar items

M_dw is a popup menu and has a single menu bar item:

Table

Table menu

Menu items

Menu item	Description
Cut	Calls the pfc_Cut event
Copy	Calls the pfc_Copy
Paste	Calls the pfc_Paste event
Select All	Calls the pfc_SelectAll event
Insert	Calls the pfc_InsertRow event
Add	Calls the pfc_AddRow event
Delete	Calls the pfc_DeleteRow event
Restore	Calls the pfc_RestoreRow event
Columns	Calls the pfc_Columns event
Functions	Calls the pfc_Functions event
Operators	Calls the pfc_Operators event

Menu item	Description
Values	Calls the pfc_Values event
DataWindow Properties	Calls the pfc_Debug event
Properties	Empty menu item. Add calls to functions or events that display properties for a DataWindow column

See also

u_dw

Instance variables

M_dw includes one instance variable:

Instance variable	Description	Data type	Access	Usage
idw_parent	Owning DataWindow control.	DataWindow	Protected	Used to access the current DataWindow control

Functions

M_dw includes one precoded object function:

of_SetParent

of_SetParent

Description

Establishes the current DataWindow control.

Access

Public

Syntax

instancename.**of_SetParent** (*currentdw*)

Argument	Description
instancename	Instance name of m_dw
currentdw	DataWindow variable identifying the DataWindow that opened this instance of m_dw

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

U_dw calls this function to establish a relationship with m_dw.

Examples

This example is from the u_dw RButtonUp event:

```
...
lm_dw = create m_dw
lm_dw.of_SetParent(this)
...
```

m_edit

Description

Popup menu displayed when the user releases the right mouse button over certain PFC visual controls.

Ancestry

Library

PFCMAIN.PBL
PFEMAIN.PBL

Menu bar items

M_edit is a popup menu and has a single menu bar item:

Edit

Edit menu

Menu items

Menu item	Description
Cut	Calls the pfc_Cut event for the current window or window control
Copy	Calls the pfc_Copy event for the current window or window control
Paste	Calls the pfc_Paste event for the current window or window control
Select All	Calls the pfc_SelectAll event for the current window or window control

See also

m_view
u_mle
u_rte
u_sle

Instance variables

M_edit includes one instance variable:

Instance variable	Description	Data type	Access	Usage
idrg_current	Current window or window control	DragObject	Protected	Used to access the current control

Functions

M_edit includes one precoded object function:

of_SetParent

of_SetParent

Description

Establishes the current control.

Access

Public

Syntax

instancename.**of_SetParent** (*currentcontrol*)

Argument	Description
instancename	Instance name of m_edit
currentcontrol	DragObject containing the control that opened this instance of m_edit

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

PFC visual objects call this function to establish a relationship with m_edit.

Examples

This example is from the u_rte RButtonUp event:

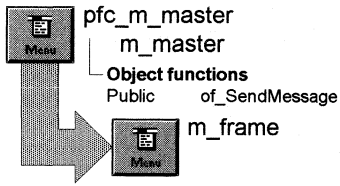
```
Integer     li_rc
m_edit      lm_edit
Window      lw_parent
String      ls_selectedtext
...
lm_edit = create m_edit
lm_edit.of_SetParent(this)
...
```

m_frame

Description

The m_frame menu is a descendant of m_master with most items disabled and hidden. Use this menu as the frame menu for your application, modifying and adding menu items as necessary.

Ancestry



Library

PFEWNSRV.PBL

Menu bar items

The m_frame menu has the following visible menu bar items:

File
Help

File menu

Menu items

Menu item	Description
New	Calls the pfc_New event for the frame window
Open	Calls the pfc_Open event for the frame window
Exit	Calls the pfc_Exit event for the frame window

Help menu

Menu items

Menu item	Description
Help Topics	Calls the pfc_Help event for the frame window
About	Calls the n_cst_appmanager of_About function

See also

m_master

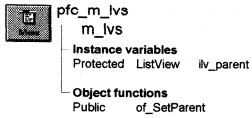
w_frame

m_lvs

Description

Popup menu displayed when the user releases the right mouse button over a ListView based on u_lvs.

Ancestry



Library

PFCMAIN.PBL
PFEMAIN.PBL

Menu bar items

M_lvs is a popup menu and has a single menu bar item:

View

View menu

Menu items

Menu item	Description
View	Changes the ListView display style. Options are: ♦ Large icon ♦ Small icon ♦ List ♦ Report
Arrange Icons	Sorts the display by the specified column
Select All	Calls the pfc_SelectAll event
Invert Selection	Calls the pfc_InvertSelection event
Cut	Calls the pfc_Cut event
Copy	Calls the pfc_Copy event
Paste	Calls the pfc_Paste event
Clear	Calls the pfc_Clear event

Menu item	Description
New	Calls the pfc_New event
Delete	Calls the pfc_Delete event
Rename	Calls the pfc_Rename event
Properties	Calls the pfc_Properties event

See also

u_lvs

Instance variables

M_lvs includes one instance variable:

Instance variable	Description	Data type	Access	Usage
ilv_parent	ListView control displaying the popup menu.	ListView	Protected	Used to access the associated ListView control

Functions

M_lvs includes one precoded object function:

of_SetParent

of_SetParent

Description

Establishes the ListView control associated with an instance of the m_lvs popup menu.

Access

Public

Syntax

instancename.**of_SetParent** (*listview*)

Argument	Description
instancename	Instance name of m_lvs
listview	ListView variable containing the ListView that opened this instance of m_lvs

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

U_lvs calls this function to establish a relationship with m_lvs.

Examples

This example calls the of_SetParent function:

```
...
if IsNull(lm_view) Or not IsValid(lm_view) then
   lm_view = create m_lvs
   lm_view.of_SetParent(this)
end if
...
```

m_master

Description

Master menu for all PFC applications. Use this menu as the ancestor for your application's menus.

Ancestry

Library

PFCWNSRV.PBL
PFEWNSRV.PBL

Menu bar items

The m_master menu has the following menu bar items:

File
Edit
View
Insert
Tools
Window
Help

File menu

Menu items

Menu item	Description
New	Calls the pfc_New event for the current window or window control
Open	Calls the pfc_Open event for the current window or window control
Close	Calls the pfc_Close event for the current window or window control
Save	Calls the pfc_Save event for the current window or window control
Save As	Calls the pfc_SaveAs event for the current window or window control

Menu item	Description
Print	Calls the pfc_Print event for the current window or window control
Print Preview	Calls the pfc_PrintPreview event for the current window or window control
Page Setup	Calls the pfc_PageSetup event for the current window or window control
Print Immediate	Calls the pfc_PrintImmediate event for the current window or window control
Delete	Empty menu item. Add calls to function or events that perform deletion
Properties	Empty menu item. Add calls to function or events that display properties
Exit	Calls the pfc_Exit event for the current window or window control
PFCMRU1 through PFCMRU5	Calls the pfc_MRUClicked event for the current window or window control

Edit menu

Menu items

Menu item	Description
Undo	Calls the pfc_Undo event for the current window or window control
Cut	Calls the pfc_Cut event for the current window or window control
Copy	Calls the pfc_Copy event for the current window or window control
Paste	Calls the pfc_Paste event for the current window or window control
Paste Special	Calls the pfc_PasteSpecial event for the current window or window control
Clear	Calls the pfc_Clear event for the current window or window control
Select All	Calls the pfc_SelectAll event for the current window or window control

Menu item	Description
Find	Calls the pfc_FindDlg event for the current window or window control
Replace	Calls the pfc_ReplaceDlg event for the current window or window control
Update Links	Calls the pfc_UpdateLinks event for the current window or window control
Object>Edit	Calls the pfc_EditObject event for the current window or window control
Object>Open	Calls the pfc_OpenObject event for the current window or window control

View menu

Menu items

Menu item	Description
Ruler	Calls the pfc_Ruler event for the current window or window control
Large Icon	Empty menu item. Add calls to function or events that change ListView display
Small Icon	Empty menu item. Add calls to function or events that change ListView display
List	Empty menu item. Add calls to function or events that change ListView display
Details	Empty menu item. Add calls to function or events that change ListView display
Arrange Icons>By	Empty menu item. Add calls to function or events that arrange icons
Arrange Icons>Auto Arrange	Empty menu item. Add calls to function or events that arrange icons
First	Calls the pfc_FirstPage event for the current window or window control
Next	Calls the pfc_NextPage event for the current window or window control
Prior	Calls the pfc_PreviousPage event for the current window or window control

Menu item	Description
Last	Calls the pfc_LastPage event for the current window or window control
Sort	Calls the pfc_SortDlg event for the current window or window control
Filter	Calls the pfc_FilterDlg event for the current window or window control
Zoom	Calls the pfc_Zoom event for the current window or window control

Insert menu

Menu items

Menu item	Description
File	Calls the pfc_InsertFile event for the current window or window control
Picture	Calls the pfc_InsertPicture event for the current window or window control
Object	Calls the pfc_InsertObject event for the current window or window control

Tools menu

Menu items

Menu item	Description
Customize Toolbars	Calls the pfc_Toolbars event for the current window or window control

Window menu

Menu items

Menu item	Description
Cascade	Calls the pfc_Cascade event for the frame window
Tile Horizontal	Calls the pfc_TileHorizontal event for the frame window
Tile Vertical	Calls the pfc_TileVertical event for the frame window
Layer	Calls the pfc_Layer event for the frame window
Minimize All	Calls the pfc_MinimizeAll event for the frame window

Menu item	Description
Undo Arrange	Calls the pfc_UndoArrange event for the frame window

Help menu

Menu items

Menu item	Description
Help Topics	Calls the pfc_Help event for the current window or window control
About	Calls the n_cst_appmanager of_About function

See also

m_frame
w_frame
w_master

Functions

The m_master menu includes one precoded object function:

of_SendMessage

of_SendMessage

Description

Calls the n_cst_menu of_SendMessage function, which sends the passed message to the current window through the pfc_MessageRouter event.

Access

Public

Syntax

instancename.**of_SendMessage** (*message*)

Argument	Description
instancename	Instance name of m_master
message	String specifying the user event to be triggered by the pfc_MessageRouter event

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

PFC menu items call this function to trigger events on the associated window and window controls.

Examples

This example is from the m_master File>Open menu item's Clicked event.

```
SetPointer (hourglass!)
of_SendMessage("pfc_open")
```

m_oc

Description

Popup menu displayed when the user releases the right mouse button over an OLE control based on u_oc.

Ancestry

Library

PFCMAIN.PBL
PFEMAIN.PBL

Menu bar items

M_oc is a popup menu and has a single menu bar item:

Object

Object menu

Menu items

Menu item	Description
Edit	Calls the pfc_EditObject event for the OLE control
Open	Calls the pfc_OpenObject event for the OLE control
Cut	Calls the pfc_Cut event for the OLE control
Copy	Calls the pfc_Copy event for the OLE control
Paste	Calls the pfc_Paste event for the OLE control

See also

u_oc

Instance variables

M_oc includes one instance variable:

Instance variable	Description	Data type	Access	Usage
ioc_parent	OLE control displaying the popup menu.	OLEControl	Protected	Used to access the associated OLE control.

Functions

M_oc includes one precoded object function:

of_SetParent

of_SetParent

Description

Establishes the OLE control associated with an instance of the m_oc popup menu.

Access

Public

Syntax

instancename.**of_SetParent** (*olecontrol*)

Argument	Description
instancename	Instance name of m_oc
olecontrol	OLEControl variable containing the OLE control that opened this instance of m_oc

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

U_oc calls this function to establish a relationship with m_oc.

Examples

This example calls the of_SetParent function:

```
m_oc      lm_oc
String    ls_selectedtext
...
lm_view = create m_oc
lm_view.of_SetParent(this)
...
```

m_tvs

Description

Popup menu displayed when the user releases the right mouse button over a TreeView based on u_tvs.

Ancestry

Library

PFCMAIN.PBL
PFEMAIN.PBL

Menu bar items

M_tvs is a popup menu and has a single menu bar item:

View

View menu

Menu items

Menu item	Description
Open	Calls the pfc_Open event
Find	Calls the pfc_FindDlg event
Cut	Calls the pfc_Cut event
Copy	Calls the pfc_Copy event
Paste	Calls the pfc_Paste event
Clear	Calls the pfc_Clear event
New	Calls the pfc_New event
Delete	Calls the pfc_Delete event
Rename	Calls the pfc_Rename event
Properties	Calls the pfc_Properties event

See also

u_tvs

Instance variables

M_tvs includes one instance variable.

Instance variable	Description	Data type	Access	Usage
itv_parent	TreeView control displaying the popup menu.	TreeView	Protected	Used to access the associated TreeView control

Functions

M_tvs includes one precoded object function:

of_SetParent

of_SetParent

Description

Establishes the TreeView control associated with an instance of the m_tvs popup menu.

Access

Public

Syntax

instancename.**of_SetParent** (*listview*)

Argument	Description
instancename	Instance name of m_tvs
listview	TreeView variable containing the TreeView that opened this instance of m_tvs

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

U_tvs calls this function to establish a relationship with m_tvs.

Examples

This example calls the of_SetParent function:

```
...
im_view = CREATE m_tvs
im_view.of_SetParent(this)
...
```

CHAPTER 4

Global Functions

About this chapter

This chapter describes the global functions in the Powersoft Foundation Class Library (PFC).

f_SetFilesrv

Description

Enables or disables the platform-specific file-handling service. The file service is implemented through the n_cst_filesrv user object and its descendants.

This function creates an operating system-specific descendant of n_cst_filesrv:

- **Windows 3.1** n_cst_filesrvwin16
- **Windows 95 and Windows NT** n_cst_filesrvwin32
- **Macintosh** n_cst_filesrvmac
- **UNIX** n_cst_filesrvsol2, n_cst_filesrvaix, or n_cst_filesrvhpux as appropriate

Access

Public

Syntax

f_SetFilesrv (*fileservice, boolean*)

Argument	Description
fileservice	N_cst_filesrv variable into which the function creates the platform-specific n_cst_filesrv descendant
boolean	Boolean specifying whether to create an instance of n_cst_filesrv (TRUE) or destroy an instance of n_cst_filesrv (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function to enable and disable the file service.

These objects contain functions that make external function calls to perform operating system-specific file processing.

Examples

This example enables the file service. It assumes an inv_filesrv instance variable:

```
f_SetFileSrv(inv_filesrv, TRUE)
```

See also

f_SetPlatform
n_cst_filesrv
n_cst_platform

f_SetPlatform

Description

Enables or disables platform-specific services. The platform service is implemented through the n_cst_platform user object and its descendants.

This function creates an operating system-specific descendant of n_cst_platform:

- **Windows 3.1** n_cst_platformwin16
- **Windows 95 and Windows NT** n_cst_platformwin32
- **Macintosh** n_cst_platformmac
- **UNIX** n_cst_platformsol2, n_cst_platformaix, or n_cst_platformhpux as appropriate

Access

Public

Syntax

f_SetPlatform (*platformservice, boolean*)

Argument	Description
platformservice	N_cst_platform variable into which the function creates the platform-specific n_cst_platform descendant
boolean	Boolean specifying whether to create an instance of n_cst_platform (TRUE) or destroy an instance of n_cst_platform (FALSE).

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function to enable and disable the platform service.

These objects contain functions that make external function calls to return operating system-specific information and play sounds.

Examples

This example enables the platform service. It assumes an inv_platform instance variable:

```
f_SetPlatform(inv_platform, TRUE)
```

See also

f_SetFilesrv
n_cst_filesrv
n_cst_platform

CHAPTER 5

Global Structures and Structure Objects

About this chapter

This chapter describes the global structures and structure objects in the PowerBuilder Foundation Class Library (PFC). Structure objects are autoinstantiated custom class user objects that consist solely of instance variables. These objects provide all the functionality of global structures and they also allow you to extend them by adding instance variables, events, and object functions.

Contents

The global structures and structure objects are listed alphabetically.

n_cst_aboutattrib

Description

Structure object containing information displayed in w_about.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
is_application	String
is_copyright	String
is_logo	String
is_version	String

See also

n_cst_appmanager

n_cst_baseattrib

Description	Ancestor of all structure objects. This object contains no properties. To define a global property or function for structure objects, define it in this object.
Library	PFCMAIN.PBL PFEMAIN.PBL
Properties	No properties
See also	All other structure objects

n_cst_calculatorattrib

Description

Structure object containing information used by u_calculator.

Library

PFCMAIN.PBL
PFEMAIN.PBL

Properties

Field	Data type
ib_dropdown	Boolean

See also

u_calculator

n_cst_calendarattrib

Description

Structure object containing information used by u_calendar.

Library

PFCMAIN.PBL
PFEMAIN.PBL

Properties

Field	Data type
ib_dropdown	Boolean

See also

u_calendar

n_cst_columnattrib

Description

Structure object containing column information.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
is_colformat	String
is_coltype	String
is_columns	String

See also

n_cst_lvsrv_datasource
n_cst_lvsrv_sort

n_cst_dberrorattrib

Description

Structure object containing database error information.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
idwb_buffer	DWBuffer
il_row	Long
il_sqldbcode	Long
ipo_inerror	PowerObject
is_errormsg	String
is_sqlerrtext	String
is_sqlsyntax	String

See also

n_cst_luw
w_master

n_cst_dirattrib

Description

Structure object containing information used in n_cst_filesrv and its descendants.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
ib_archive	Boolean
ib_drive	Boolean
ib_hidden	Boolean
ib_readonly	Boolean
ib_subdirectory	Boolean
ib_system	Boolean
id_creationdate	Date
id_lastaccessdate	Date
id_lastwritedate	Date
idb_filesize	Date
is_altfilename	String
is_filename	String
it_creationtime	Time
it_lastwritetime	Time

See also

n_cst_filesrv
n_cst_filesrvwin16
n_cst_filesrvwin32

n_cst_dwobjectattrib

Description

Structure object containing information used in n_cst_dwcache, n_cst_dssrv, n_cst_dwsrv, and its descendants.

Library

PFCDWSRV.PBL
PFEDWSRV.PBL

Properties

Field	Data type
is_column	String
is_datatype	String
is_value	String

See also

n_cst_dwcache
n_cst_dssrv
n_cst_dwsrv

n_cst_dwpropertyattrib

Description

Structure object containing DataWindow property information.

Library

PFCUTIL.PBL
PFEUTIL.PBL

Properties

Field	Data type
idw_requestor	DataWindow
is_dataobjectbuffer	String
is_dataobjectstatus	String
is_objectname	String
is_tabpages	String
iw_mail	Window

See also

w_dwproperty

n_cst_errorattrib

Description

Structure object containing information used in n_cst_error and w_message.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
Boolean	ib_print
Boolean	ib_userinput
Datetime	idt_date
Button	ie_buttonstyle
Icon	ie_icon
Integer	ii_buttonclicked
Integer	ii_default
Integer	ii_rc
Integer	ii_severity
Integer	ii_timeout
String	is_text
String	is_title
String	is_user
String	is_usertext

See also

n_cst_error

n_cst_filterattrib

Description

Structure object containing information used in n_cst_dwsrv_filter and Filter dialog boxes.

Library

PFCDWSRV.PBL
PFEDWSRV.PBL

Properties

Field	Data type
idw_dw	u_dw
is_colnamedisplay[ ]	String
is_columns[ ]	String
is_dbnames[ ]	String
is_filter	String

See also

n_cst_dwsrv_filter

n_cst_findattrib

Description

Structure object containing information used in n_cst_dwsrv_find, u_rte, w_find, and w_replace.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
ib_directionenabled	Boolean
ib_directionvisible	Boolean
ib_lookenabled	Boolean
ib_lookvisible	Boolean
ib_matchcase	Boolean
ib_matchcaseenabled	Boolean
ib_matchcasevisible	Boolean
ib_wholeword	Boolean
ib_wholewordenabled	Boolean
ib_wholewordvisible	Boolean
ii_lookindex	Integer
ipo_requestor	PowerObject
is_direction	String
is_find	String
is_lookdata[]	String
is_lookdisplay[]	String
is_replacewith	String

See also

n_cst_dwsrv_find
u_rte

n_cst_infoattrib

Description

Structure object containing information displayed by the DataWindow Properties dialog box.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
is_description	String
is_name	String

See also

All objects that contain an of_GetInfo function.

n_cst_itemattrib

Description

Structure object containing information used in u_lb, u_plb, and u_tv.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
ii_index	Integer
is_itemtext	String

See also

u_lb
u_plb
u_tvs

n_cst_linkageattrib

Description

Structure object containing information used in n_cst_dwsrv_linkage.

Library

PFCDWSRV.PBL
PFEDWSRV.PBL

Properties

Field	Data type
is_detailcolarg[ ]	String
is_mastercolarg[ ]	String

See also

n_cst_dwsrv_linkage

n_cst_logonattrib

Description

Structure object containing information used in n_cst_appmanager.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
ii_rc	Integer
ipo_source	PowerObject
is_appname	String
is_logo	String
is_password	String
is_userid	String

See also

n_cst_appmanager

n_cst_lvsrvattrib

Description

Structure object containing information used by ListView services.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
ids_source	n_ds
itr_object	n_tr
is_dataobject	String
is_keycolumns[ ]	String
is_labelcolumn	String
is_method	String
is_overlaycolumn	String
is_picturecolumn	String
is_statecolumn	String
is_xposcolumn	String
is_yposcolumn	String

See also

n_cst_lvsrv_datasource

n_cst_propertyattrib

Description

Structure object containing information used by the DataWindow Properties dialog box.

Library

PFCUTIL.PBL
PFEUTIL.PBL

Properties

Field	Data type
ib_switchbuttons	Boolean
is_description	String
is_name	String
is_propertypage[ ]	String
is_propertytabtext	String

See also

All objects that contain an of_GetPropertyInfo function.

n_cst_restorerowattrib

Description

Structure object containing information used by n_cst_dwsrv_rowmanager when restoring deleted rows.

Library

PFCDWSRV.PBL
PFEDWSRV.PBL

Properties

Field	Data type
idw_active	DataWindow
is_filter	String
is_sort	String

See also

n_cst_dwsrv_rowmanager

n_cst_returnattrib

Description

Structure object containing return code and associated text.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
ii_rc	Integer
is_rs	String

See also

n_cst_dwsrv_sort

n_cst_selectionattrib

Description

Structure object containing information used in n_cst_selection and w_selection.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
ia_argument[20]	Any
ia_returnval[ ]	Any
ipo_data[ ]	PowerObject
is_columnreturn[ ]	String
is_dataobject	String
is_title	String
itr_object	n_tr

See also

n_cst_selection

n_cst_sortattrib

Description

Structure object containing information used in n_cst_dwsrv_sort and the Sort dialog boxes.

Library

PFCDWSRV.PBL
PFEDWSRV.PBL

Properties

Field	Data type
ib_usedisplay[ ]	Boolean
is_colnamedisplay[ ]	String
is_origcolumns[ ]	String
is_origorder[ ]	String
is_sort	String
is_sortcolumns[ ]	String

See also

n_cst_dwsrv_sort

n_cst_splashattrib

Description

Structure object containing information used in n_cst_appmanager and w_splash.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
ii_secondsvisible	Integer
is_application	String
is_copyright	String
is_logo	String
is_version	String

See also

n_cst_appmanager

n_cst_sqlattrib

Description

Structure object containing information used in n_cst_sql.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
s_columns	String
s_group	String
s_having	String
s_order	String
s_tables	String
s_values	String
s_verb	String
s_where	String

See also

n_cst_sql

n_cst_textstyleattrib

Description

Structure object used by u_rte-based RichTextEdit controls to access text style information.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
ib_bold	Boolean
ib_italic	Boolean
ib_strikeout	Boolean
ib_subscript	Boolean
ib_superscript	Boolean
ib_underlined	Boolean

See also

u_rte

n_cst_tmgregisterattrib

Description

Structure object containing information used by n_cst_tmgmultiple.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
ii_notifystyle	Integer
il_notifyinterval	Long
ipo_notify	PowerObject
is_notifyevent	String

See also

n_cst_tmgmultiple

n_cst_toolbarattrib

Description

Structure object containing information used in w_frame and w_toolbars.

Library

PFCWNSRV.PBL
PFEWNSRV.PBL

Properties

Field	Data type
ib_largebuttonsenabled	Boolean
ib_positionenabled	Boolean
ib_tooltipsenabled	Boolean
ib_visibleenabled	Boolean
iw_owner	Window

See also

w_frame

n_cst_tvsrvattrib

Description

Structure object containing information used by n_cst_tvsrv_levelsource.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
ib_appenddatastore	Boolean
ib_recursive	Boolean
ii_deletestyle	Integer
ids_obj	n_ds
itr_obj	n_tr
is_dataobject	String
is_keycolumns	String
is_labelcolumn	String
is_method	String
is_overlaycolumn	String
is_picturecolumn	String
is_retrieveargs	String
is_selectedcolumn	String
is_statecolumn	String

See also

n_cst_tvsrv_levelsource

n_cst_zoomattrib

Description

Structure object containing information used in n_cst_dwsrv_printpreview and w_zoom.

Library

PFCDWSRV.PBL
PFEDWSRV.PBL

Properties

Field	Data type
idw_obj	DataWindow
ii_zoom	Integer

See also

n_cst_dwsrv_printpreview

s_pagesetupattrib

Description

Global structure containing information used in n_cst_platform and w_pagesetup.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
b_disablemargins	Boolean
b_disableorientation	Boolean
b_disablepaper	Boolean
i_minmarginleft	Integer
i_minmarginright	Integer
i_minmargintop	Integer
i_minmarginbottom	Integer
i_marginleft	Integer
i_marginright	Integer
i_margintop	Integer
i_marginbottom	Integer
i_papersize	Integer
i_papersource	Integer
b_portraitorientation	Boolean
str_papersize[ ]	s_paperattrib
str_papersource[ ]	s_paperattrib
i_units	Integer
b_actiontaken	Boolean

See also

n_cst_platform

s_paperattrib

Description

Global structure containing information used in s_pagestupattrib.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
i_val	Integer
s_type	String

See also

n_cst_platform
s_pagesetupattrib

s_printdlgattrib

Description

Structure object containing information used by n_cst_platform for printing.

Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Properties

Field	Data type
b_allpages	Boolean
b_pagenums	Boolean
b_selection	Boolean
b_disablepagenums	Boolean
b_disableselection	Boolean
b_collate	Boolean
l_copies	Long
b_printtofile	Boolean
b_disableprinttofile	Boolean
b_hideprinttofile	Boolean
l_frompage	Long
l_topage	Long
l_minpage	Long
l_maxpage	Long

See also

n_cst_platform

s_svalue

Description

Global structure containing information used in n_cst_dwsrv_querymode.

Library

PFCDWSRV.PBL
PFEDWSRV.PBL

Properties

Field	Data type
s_value	String

See also

n_cst_platform
s_pagesetupattrib

CHAPTER 6

Standard Visual User Objects

About this chapter

This chapter describes the standard visual user objects in PFC.

Contents

The standard visual user objects are listed in alphabetical order. Each object's discussion includes alphabetical listings of instance variables, events, and object functions.

u_cb

Description

Command button visual user object.

Ancestry

pfc_u_cb
u_cb
Events
GetFocus

Functions
Public of_GetParentWindow
Protected of_MessageBox

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

PFC visual user objects are designed to be used with windows that are descendants of w_master.

Usage

Use this visual object in windows instead of the standard PowerBuilder CommandButton. U_cb event scripts provide integration with PFC menus.

See also

m_master
w_master

Events

U_cb includes a precoded event script:

GetFocus

GetFocus

Description

Updates the parent window so it can set MicroHelp.

Usage

This event calls the pfc_ControlGotFocus event in the parent window.

Functions

U_cb includes precoded object functions:

of_GetParentWindow
of_MessageBox

of_GetParentWindow

Description

Retrieves a reference to the parent window.

Access

Public

Syntax

instancename.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_cb
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if there is no parent window. If no parent window is found, *window* returns NULL.

Usage

The u_cb GetFocus event calls this function.

Examples

This example is from the u_cb GetFocus event:

```
Window   lw_parent

//Check for MicroHelp requirements.
IF gnv_app.of_GetMicrohelp() THEN
   //Notify the parent.
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
      lw_parent.Dynamic Event &
      pfc_ControlGotFocus (this)
   END IF
END IF
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of u_cb
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in CommandButtons.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('cb_error','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

u_cbx

Description

CheckBox visual user object.

Ancestry

pfc_u_cbx
u_cbx
Events
GetFocus
Functions
Public of_GetParentWindow
Protected of_MessageBox

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

PFC visual user objects are designed to be used with windows that are descendants of w_master.

Usage

Use this visual object in windows instead of the standard PowerBuilder CheckBox. U_cbx event scripts provide integration with PFC menus.

See also

m_master
w_master

Events

U_cbx includes a precoded event script:

GetFocus

GetFocus

Description

Updates the parent window so it can set MicroHelp.

Usage

This event calls the pfc_ControlGotFocus event in the parent.

Functions

U_cbx includes precoded object functions:

of_GetParentWindow
of_MessageBox

of_GetParentWindow

Description — Retrieves a reference to the parent window.

Access — Public

Syntax — *instancename*.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_cbx
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value — Integer. Returns 1 if the function succeeds and -1 if there is no parent window. If no parent window is found, *window* returns NULL.

Usage — The u_cbx GetFocus event calls this function.

Examples — This example is from the u_cbx GetFocus event:

```
Window lw_parent

//Check for MicroHelp requirements.
IF gnv_app.of_GetMicrohelp() THEN
   //Notify the parent.
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
     lw_parent.Dynamic Event &
     pfc_ControlGotFocus (this)
   END IF
END IF
```

of_MessageBox

Description — Displays a MessageBox.

Access — Protected

Syntax — *instancename*.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of u_cbx
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in CheckBoxes.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('cbx_error','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

u_ddlb

Description

DropDownListBox visual user object ancestor. This object provides:

- Integration with PFC menus for the cut, copy, paste, clear, and select all Edit menu choices
- A function to display a popup Edit menu when the user points at the DropDownListBox and right-clicks
- An event you can use to scroll to the matching letters in the list as the user types

Ancestry

pfc_u_ddlb
u_ddlb
Events

CbnEditChange	pfc_Paste
GetFocus	pfc_PreRMBMenu
pfc_Clear	pfc_SelectAll
pfc_Copy	RBbuttonUp
pfc_Cut	

Object functions

Public	of_GetParentWindow
Protected	of_MessageBox

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

PFC visual user objects are designed to be used with windows that are descendants of w_master.

Usage

Use this visual object in windows instead of the standard PowerBuilder DropDownListBox. U_ddlb event scripts provide integration with PFC menus. Additionally, u_ddlb supports:

- **Cut, copy, paste, clear, and select all** These are triggered automatically by the message router.
- **Right mouse button support** The RButtonUp event enables you to use the right mouse button to perform editing actions. To disable right mouse button support, set ib_rmbmenu to FALSE in the dropdown listbox's Constructor event.
- **Autoselect** This means that text is selected when a user tabs to the dropdown listbox. To disable autoselect, set ib_autoselect to FALSE in the dropdown listbox's Constructor event.

- **Autoscroll** This means that PFC scrolls to matching dropdown listbox entries as you type. To enable autoscroll, set ib_search to TRUE in the dropdown listbox's Constructor event.

See also

m_master
m_edit
u_ddplb

Instance variables

U_ddlb includes instance variables:

Instance variable	Description	Data type	Access	Usage
ib_autoselect	Indicates whether PFC selects text automatically when the control receives focus	Boolean	Protected	Set this to TRUE to enable autoselect (default is FALSE)
ib_rmbmenu	Indicates whether the m_edit menu displays when the user presses the right mouse button	Boolean	Protected	Set this to FALSE to disable right mouse button support (default is TRUE)
ib_search	Controls whether PFC automatically scrolls to matching entries as the user types	Boolean	Protected	Set this to TRUE to enable autoscroll (default is FALSE)

Events

U_ddlb includes precoded event scripts:

CbnEditChange	pfc_Paste
GetFocus	pfc_PreRmbMenu
pfc_Clear	pfc_SelectAll
pfc_Copy	RButtonUp
pfc_Cut	

CbnEditChange

Description

Scrolls the DropDownListBox, based on the typed characters. For example, if you press *r*, the event scrolls to the first entry beginning with r; if you then press *i*, the event scrolls to the first entry beginning with ri.

This event maps to the pbm_cbneditchange event.

This capability differs from the default DropDownListBox behavior, which scrolls based on the first letter only. For example, if you press *r*, the list scrolls to the first entry beginning with r; if you then press *i*, the list scrolls to the first entry beginning with i.

Usage

To use this functionality, you must enable the Allow Edit property for the DropDownListBox. To disable this functionality, set ib_search to FALSE.

GetFocus

Description

Updates the parent window so it can set MicroHelp.

Usage

This event calls the pfc_ControlGotFocus event in the parent.

pfc_Clear

Description

Deletes selected text.

Return value

Integer. Returns the number of characters deleted if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a DropDownListBox based on u_ddlb has focus and the user selects Edit>Clear.

pfc_Copy

Description

Copies selected text to the clipboard.

Return value

Integer. Returns the number of characters copied if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a DropDownListBox based on u_ddlb has focus and the user selects Edit>Copy.

This event is also called by the m_edit popup menu.

pfc_Cut

Description

Deletes selected text and stores it on the clipboard.

Return value

Integer. Returns the number of characters removed if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a DropDownListBox based on u_ddlb has focus and the user selects Edit>Cut.

This event is also called by the m_edit popup menu.

pfc_Paste

Description

Inserts (pastes) the contents of the clipboard at the insertion point.

Return value

Integer. Returns the number of characters that were pasted if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a DropDownListBox based on u_ddlb has focus and the user selects Edit>Paste.

This event is also called by the m_edit popup menu.

pfc_PreRmbMenu

Description

User event allowing you to modify m_edit contents before display.

Syntax

instancename.Event **pfc_PreRmbMenu** (*editmenu*)

Argument	Description
instancename	Instance name of u_ddlb
editmenu	M_edit variable containing the popup menu to be displayed (passed by reference)

Return value

None

Usage

Optionally add logic to this event to selectively enable and disable m_edit menu items.

pfc_SelectAll

Description

Selects all text in the DropDownListBox.

Return value

Integer. Returns the number of characters selected if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a DropDownListBox based on u_ddlb has focus and the user selects Edit>Select All.

This event is also called by the m_edit popup menu.

RButtonUp

Description

Displays the m_edit popup menu.

Usage

This event executes when the user releases the right mouse button over a control based on u_ddlb.

Functions

U_ddlb includes precoded object functions:

of_GetParentWindow
of_MessageBox

of_GetParentWindow

Description

Retrieves a reference to the parent window.

Access

Public

Syntax

instancename.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_ddlb
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if there is no parent window. If no parent window is found, *window* returns NULL.

Usage

The u_ddlb GetFocus event calls this function.

Examples

This example is from the u_ddlb GetFocus event:

```
Window  lw_parent

//Check for MicroHelp requirements.
IF gnv_app.of_GetMicrohelp() THEN
   //Notify the parent.
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
      lw_parent.Dynamic Event &
      pfc_ControlGotFocus (this)
   END IF
END IF
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id*, *title*, *message*, *icon*, *button*, *default*)

Argument	Description
instancename	Instance name of u_ddlb
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!

Argument	Description
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in DropDownListBoxes.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('ddlb_error','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

u_ddplb

Description

DropDownPictureListBox user object ancestor. This object provides:

- Integration with PFC menus for the cut, copy, paste, clear, and select all Edit menu choices
- A function to display a popup Edit menu when the user points at the DropDownPictureListBox and right-clicks
- An event you can use to scroll to the matching letters in the list as the user types

Ancestry

pfc_u_ddplb
u_ddplb

Events

CbnEditChange	pfc_Paste
GetFocus	pfc_PreRMBMenu
pfc_Clear	pfc_SelectAll
pfc_Copy	RButtonUp
pfc_Cut	

Object functions

Public	of_GetParentWindow
Protected	of_MessageBox

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

PFC visual user objects are designed to be used with windows that are descendants of w_master.

Usage

Use this visual object in windows instead of the standard PowerBuilder DropDownPictureListBox. U_ddplb event scripts provide integration with PFC menus. Additionally, u_ddplb supports:

- **Cut, copy, paste, clear, and select all** These are triggered automatically by the message router.
- **Right mouse button support** The RButtonUp event enables you to use the right mouse button to perform editing actions. To disable right mouse button support, set ib_rmbmenu to FALSE in the dropdown picturelistbox's Constructor event.
- **Autoselect** This means that text is selected when a user tabs to the dropdown picturelistbox. To disable autoselect, set ib_autoselect to FALSE in the dropdown picturelistbox's Constructor event.

- **Autoscroll** This means that PFC scrolls to matching dropdown picturelistbox entries as you type. To enable autoscroll, set ib_search to TRUE in the dropdown picturelistbox's Constructor event.

See also

m_edit
u_ddlb

Instance variables

U_ddplb includes instance variables:

Instance variable	Description	Data type	Access	Usage
ib_autoselect	Indicates whether PFC selects text automatically when the control receives focus	Boolean	Protected	Set this to TRUE to enable autoselect (default is FALSE)
ib_rmbmenu	Controls whether the m_edit menu displays when the user presses the right mouse button	Boolean	Protected	Set this to FALSE to disable right mouse button support (default is TRUE)
ib_search	Controls whether PFC automatically scrolls to matching entries as the user types	Boolean	Protected	Set this to TRUE to enable autoscroll (default is FALSE)

Events

U_ddplb includes precoded event scripts:

CbnEditChange	pfc_Paste
GetFocus	pfc_PreRmbMenu
pfc_Clear	pfc_SelectAll
pfc_Copy	RButtonUp
pfc_Cut	

CbnEditChange

Description

Scrolls the DropDownPictureListBox, based on the typed characters. For example, if you press *r*, the event scrolls to the first entry beginning with r; if you then press *i*, the event scrolls to the first entry beginning with ri.

This event maps to the pbm_cbneditchange event.

This capability differs from the default DropDownPictureListBox behavior, which scrolls based on the first letter only. For example, if you press *r*, the list scrolls to the first entry beginning with r; if you then press *i*, the list scrolls to the first entry beginning with i.

Usage

To use this functionality, you must enable the Allow Edit property for the DropDownPictureListBox. To disable this functionality, set ib_search to FALSE.

GetFocus

Description

Updates the parent window so it can set MicroHelp.

Usage

This event calls the pfc_ControlGotFocus event in the parent.

pfc_Clear

Description

Deletes selected text.

Return value

Integer. Returns the number of characters deleted if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a DropDownPictureListBox based on u_ddplb has focus and the user selects Edit>Clear.

pfc_Copy

Description

Copies selected text to the clipboard.

Return value

Integer. Returns the number of characters copied if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a DropDownPictureListBox based on u_ddplb has focus and the user selects Edit>Copy.

This event is also called by the m_edit popup menu.

pfc_Cut

Description

Deletes selected text and stores it on the clipboard.

Return value

Integer. Returns the number of characters removed if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a DropDownPictureListBox based on u_ddplb has focus and the user selects Edit>Cut.

This event is also called by the m_edit popup menu.

pfc_Paste

Description

Inserts (pastes) the contents of the clipboard at the insertion point.

Return value

Integer. Returns the number of characters that were pasted if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a DropDownPictureListBox based on u_ddplb has focus and the user selects Edit>Paste.

This event is also called by the m_edit popup menu.

pfc_PreRmbMenu

Description

User event allowing you to modify m_edit contents before display.

Syntax

instancename.Event **pfc_PreRmbMenu** (*editmenu*)

Argument	Description
instancename	Instance name of u_ddplb
editmenu	M_edit variable containing the popup menu to be displayed (passed by reference)

Return value

None

Usage

Optionally add logic to this event to selectively enable and disable m_edit menu items.

pfc_SelectAll

Description: Selects all text in the DropDownPictureListBox.

Return value: Integer. Returns the number of characters selected if the event succeeds and -1 if an error occurs.

Usage: The message router calls this event when a DropDownPictureListBox based on u_ddplb has focus and the user selects Edit>Select All.

This event is also called by the m_edit popup menu.

RButtonUp

Description: Displays the m_edit popup menu.

Usage: This event executes when the user releases the right mouse button over a control based on u_ddplb.

Functions

U_ddplb includes precoded object functions:

of_GetParentWindow
of_MessageBox

of_GetParentWindow

Description: Retrieves a reference to the parent window.

Access: Public

Syntax: *instancename*.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_ddplb
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value: Integer. Returns 1 if the function succeeds and -1 if there is no parent window. If no parent window is found, *window* returns NULL.

Usage

The u_ddplb GetFocus event calls this function.

Examples

This example is from the u_ddplb GetFocus event:

```
Window  lw_parent

//Check for MicroHelp requirements.
IF gnv_app.of_GetMicrohelp() THEN
   //Notify the parent.
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
      lw_parent.Dynamic Event &
         pfc_ControlGotFocus (this)
   END IF
END IF
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id*, *title*, *message*, *icon*, *button*, *default*)

Argument	Description
instancename	Instance name of u_ddplb
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!

Argument	Description
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in DropDownPictureListBoxes.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('ddplb_error','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

u_dw

Description

Ancestor for all DataWindow controls in applications written using PFC. This DataWindow user object includes:

- Functions to enable and disable DataWindow services
- Events that automatically make use of enabled DataWindow services
- Precoded user events that provide basic editing functionality
- Template user events to which you can add application-specific functionality

U_dw is a self-updating object.

Ancestry

pfc_u_dw
u_dw

Events

Clicked	pfc_NextPage	pfc_ResetUpdate
DBError	pfc_Operators	pfc_RestoreRow
Destructor	pfc_PageSetup	pfc_Retrieve
DropDown	pfc_PageSetupDlg	pfc_RetrieveDDDW
GetFocus	pfc_Paste	pfc_RowChanged
ItemChanged	pfc_PopulateDDDW	pfc_RowValidation
ItemError	pfc_PostUpdate	pfc_Ruler
ItemFocusChanged	pfc_PreDeleteRow	pfc_SelectAll
LButtonDown	pfc_PreFindDlg	pfc_SortDlg
LButtonUp	pfc_PreInsertRow	pfc_Undo
pfc_AcceptText	pfc_PrePageSetupDlg	pfc_Update
pfc_AddRow	pfc_PrePrintDlg	pfc_UpdatePrep
pfc_Clear	pfc_PreProperties	pfc_UpdatesPending
pfc_Copy	pfc_PreReplaceDlg	pfc_Validation
pfc_Cut	pfc_PreRestoreRow	pfc_Values
pfc_DDCalculator	pfc_PreRMBMenu	pfc_Zoom
pfc_DDCalendar	pfc_PreUpdate	RButtonDown
pfc_Debug	pfc_PreviousPage	RButtonUp
pfc_DeleteRow	pfc_Print	Resize
pfc_Descendant	pfc_PrintDlg	RetrieveEnd
pfc_FilterDlg	pfc_PrintImmediate	RetrieveStart
pfc_FindDlg	pfc_PrintPreview	RowFocusChanged
pfc_FirstPage	pfc_Properties	RowFocusChanging
pfc_InsertRow	pfc_ReplaceDlg	SQLPreview
pfc_LastPage		

Object functions

Public	of_AcceptText	Public	of_SetMultiTable
Public	of_CheckRequired	Public	of_SetPrintPreview
Public	of_GetParentWindow	Public	of_SetProperty
Public	of_GetUpdateable	Public	of_SetQuerymode
Public	of_IsRoot	Public	of_SetReport
Public	of_IsSharedProperty	Public	of_SetReqColumn
Public	of_IsUpdateable	Public	of_SetResize
Protected	of_MessageBox	Public	of_SetRowManager
Public	of_PostUpdate	Public	of_SetRowSelect
Public	of_Reset	Public	of_SetSharedProperty
Public	of_Retrieve	Public	of_SetSort
Public	of_SetBase	Public	of_SetTransObject
Public	of_SetDropDownCalculator	Public	of_SetUpdateable
Public	of_SetDropDownCalendar	Public	of_SetUpdateRequestor
Public	of_SetDropDownSearch	Public	of_Update
Public	of_SetFilter	Public	of_UpdatePrep
Public	of_SetFind	Public	of_UpdatesPending
Public	of_SetLinkage	Public	of_Validation

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

m_dw
n_cst_conversion
n_cst_dwpropertyattrib
n_cst_findattrib

n_cst_platform
n_cst_restorerowattrib
n_cst_string
s_pagesetupattrib
s_printdlgattrib

Although u_dw contains substantial standalone functionality, the DataWindow services user objects (those objects whose name starts with n_cst_dwsrv) must be available to get the most out of it.

Usage

To use u_dw:

1 Place a u_dw user object in your window.

2 In the window Open event (or some other appropriate place), enable DataWindow services as needed by your application.

3 Extend basic u_dw functionality by adding PowerScript code to events and user events.

4 Add application-specific functionality as needed.

See also

n_cst_dwsrv
n_cst_dwsrv_dropdownsearch
n_cst_dwsrv_filter
n_cst_dwsrv_find
n_cst_dwsrv_linkage
n_cst_dwsrv_multitable
n_cst_dwsrv_printpreview
n_cst_dwsrv_property
n_cst_dwsrv_querymode
n_cst_dwsrv_report
n_cst_dwsrv_reqcolumn
n_cst_dwsrv_resize
n_cst_dwsrv_rowmanager
n_cst_dwsrv_rowselection
n_cst_dwsrv_sort
u_calculator
u_calendar

Instance variables

U_dw contains instance variables and one shared variable:

Instance variable	Description	Data type	Access	Usage
CONTINUE_ACTION	Constant set to 1	Integer	Public	Internal
FAILURE	Constant set to -1	Integer	Public	Internal
ib_isupdateable	Indicates whether the DataWindow can be updated	Boolean	Protected	Access through the of_GetUpdateable and of_SetUpdateable functions. Default is TRUE
ib_rmbfocuschange	Used to track focus change while the right mouse button is down	Boolean	Protected	Internal
ib_rmbmenu	Controls whether the m_dw menu displays when the user presses the right mouse button over the DataWindow control	Boolean	Protected	Set FALSE in the DataWindow's Constructor event to disable right mouse button support
inv_base	Reference variable for basic DataWindow services	n_cst_dwsrv	Public	Use in dot notation to access n_cst_dwsrv functions and attributes
inv_dropdownsearch	Reference variable for the dropdown DataWindow search service	n_cst_dwsrv_ dropdownsearch	Public	Use in dot notation to access dropdown DataWindow search functions and attributes
inv_filter	Reference variable for the DataWindow filter service	n_cst_dwsrv_filter	Public	Use in dot notation to access DataWindow filter service functions and attributes
inv_find	Reference variable for the DataWindow find service	n_cst_dwsrv_find	Public	Use in dot notation to access DataWindow find service functions and attributes

Instance variable	Description	Data type	Access	Usage
inv_linkage	Reference variable for the DataWindow linkage service	n_cst_dwsrv_linkage	Public	Use in dot notation to access n_cst_dwsrv_linkage functions and attributes
inv_multitable	Reference variable for the DataWindow multitable update service	n_cst_dwsrv_ multitable	Public	Use in dot notation to access n_cst_dwsrv_ multitable functions and attributes
inv_printpreview	Reference variable for the DataWindow print preview service	n_cst_dwsrv_ printpreview	Public	Use in dot notation to access n_cst_dwsrv_ printpreview functions and attributes
inv_property	Reference variable for the DataWindow property service	n_cst_dwsrv_property	Public	Internal
inv_querymode	Reference variable for the query mode service	n_cst_dwsrv_ querymode	Public	Use in dot notation to access n_cst_dwsrv_ querymode functions and attributes
inv_report	Reference variable for the reporting service	n_cst_dwsrv_report	Public	Use in dot notation to access n_cst_dwsrv_report functions and attributes
inv_reqcolumn	Reference variable for the required column service	n_cst_dwsrv_ reqcolumn	Public	Use in dot notation to access n_cst_dwsrv_ reqcolumn functions and attributes
inv_resize	Reference variable for the resize service	n_cst_dwsrv_resize	Public	Use in dot notation to access n_cst_dwsrv_resize functions and attributes
inv_rowmanager	Reference variable for the row manager service	n_cst_dwsrv_ rowmanager	Public	Use in dot notation to access n_cst_dwsrv_ rowmanager functions and attributes

Instance variable	Description	Data type	Access	Usage
inv_rowselect	Reference variable for the row selection service	n_cst_dwsrv_ rowselection	Public	Use in dot notation to access n_cst_dwsrv_ rowselection functions and attributes
inv_sort	Reference variable for the sort service	n_cst_dwsrv_sort	Public	Use in dot notation to access n_cst_dwsrv_sort functions and attributes
is_updatesallowed	Specifies allowable update types	String	Protected	Internal
itr_object	Transaction object used by the DataWindow	n_tr	Public	DataWindow services use this instance variable to track the transaction object Set with of_SetTransObject
iuo_calculator	Reference variable for dropdown calculator	u_calculator	Public	Use in dot notation to access u_calculator events, functions, and attributes
iuo_calendar	Reference variable for dropdown calendar	u_calendar	Public	Use in dot notation to access u_calendar events, functions, and attributes
NO_ACTION	Constant set to 0	Integer	Public	Internal
PREVENT_ACTION	Constant set to 0	Integer	Public	Internal
snv_property	Shared reference variable for the DataWindow property service	n_cst_dwsrv_property	Public	Internal
SUCCESS	Constant set to 1	Integer	Public	Internal

Events

U_dw contains two types of events/user events:

- Precoded events and user events that PFC uses to provide functionality
- Template user events to which you optionally add PowerScript code to take advantage of DataWindow services

Clicked	pfc_PrePrintDlg
DBError	pfc_PreProperties
Destructor	pfc_PreReplaceDlg
Dropdown	pfc_PreRestoreRow
GetFocus	pfc_PreRMBMenu
ItemChanged	pfc_PreUpdate
ItemError	pfc_PreviousPage
ItemFocusChanged	pfc_Print
LButtonDown	pfc_PrintDlg
LButtonUp	pfc_PrintImmediate
pfc_AcceptText	pfc_PrintPreview
pfc_AddRow	pfc_ReplaceDlg
pfc_Clear	pfc_ResetUpdate
pfc_Copy	pfc_RestoreRow
pfc_Cut	pfc_Retrieve
pfc_DDCalculator	pfc_RetrieveDDDW
pfc_DDCalendar	pfc_RowChanged
pfc_Debug	pfc_RowValidation
pfc_DeleteRow	pfc_Ruler
pfc_Descendant	pfc_SelectAll
pfc_FilterDlg	pfc_SortDlg
pfc_FindDlg	pfc_Undo
pfc_FirstPage	pfc_Update
pfc_InsertRow	pfc_UpdatePrep
pfc_LastPage	pfc_UpdatesPending
pfc_NextPage	pfc_Validation
pfc_Operators	pfc_Values
pfc_PageSetup	pfc_Zoom
pfc_PageSetupDlg	RButtonDown
pfc_Paste	RButtonUp
pfc_PopulateDDDW	Resize
pfc_PostUpdate	RetrieveEnd
pfc_PreDeleteRow	RetrieveStart
pfc_PreFindDlg	RowFocusChanged
pfc_PreInsertRow	SQLPreview
pfc_PrePageSetupDlg	

Clicked

Description

Notifies DataWindow selection service, linkage service, and sort service functions, if enabled.

Usage

This event executes when the user clicks in the DataWindow. The functions called in the Clicked event automatically handle row selection and sorting, if enabled.

For the selection service, you may need to add logic in other places to access selected rows and perform application specific processing.

DBError

Description

Displays a message box informing the user that a database error occurred.

Usage

This event is called when there is a database error.

Destructor

Description

Destroys all existing DataWindow service objects.

Usage

This event executes when the DataWindow closes.

Dropdown

Description

Indicates whether the current dropdown object is associated with u_calculator or u_calendar.

Usage

This event executes when a DropDownListBox or DropDownDataWindow begins the dropdown process.

GetFocus

Description

Notifies the parent window so it can track the current control.

Usage

This event executes when the DataWindow control gets focus. It triggers the pfc_ControlGotFocus event in the parent.

ItemChanged

Description — Notifies the linkage service that an item has changed.

Usage — This event executes when a field in the DataWindow control has been modified and loses focus.

ItemError

Description — When an error occus, calls the required column service, if enabled, to set the ActionCode to 3, causing the DataWindow to accept the value.

Usage — Use this event to add application-specific error-handling functionality.

ItemFocusChanged

Description — Notifies the linkage service that item focus has changed, Also, if MicroHelp is enabled for the application, this event calls the pfc_Microhelp event on the parent.

Usage — This event executes when the user moves to a different item within a DataWindow. It displays MicroHelp using text defined in the DataWindow column's tag value. The column's tag value must specify MICROHELP=*microhelptext*.

You enable MicroHelp display through the n_cst_appmanager of_SetMicrohelp function.

If you are using the dropdown DataWindow search service (n_cst_dwsrv_dropdownsearch), add code to this event that calls the n_cst_dwsrv_dropdownsearch pfc_ItemFocusChanged event.

Examples — This example shows the code you add to call the n_cst_dropdownsearch pfc_ItemFocusChanged event:

```
inv_dropdownsearch.Event pfc_ItemFocusChanged &
   (row, dwo)
```

LButtonDown

Description — Calls the row selection service's pfc_LButtonDown event.

Usage

This event executes when the user presses the left mouse button over a DataWindow.

LButtonUp

Description

Calls the row selection service's pfc_LButtonUp event.

Usage

This event executes when the user releases the left mouse button over a DataWindow.

pfc_AcceptText

Description

Accepts text for the DataWindow control, optionally setting focus if an error occurs.

Syntax

instancename.Event **pfc_AcceptText** (*focusonerror*)

Argument	Description
instancename	Instance name of u_dw
focusonerror	Boolean indicating whether focus should be set if an error occurs. This argument is accessed through the *ab_focusonerror* argument

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is called by the w_master event, as part of its default save process.

pfc_AddRow

Description

Calls the row manager service to add a row at the end of the DataWindow. Also notifies the linkage service, if enabled.

Return value

Long. Returns the number of the new row if the event succeeds and -1 if an error occurs.

Usage

This event also calls the pfc_PreInsertRow event.

pfc_Clear

Description

Deletes selected text.

Return value

Integer. Returns the number of characters deleted if the event succeeds and -1 if an error occurs.

Usage

This event is triggered when the DataWindow has focus and the user selects Edit>Clear from the menu bar of a menu descended from the PFC m_master menu.

pfc_Copy

Description

Copies selected text to the clipboard.

Return value

Integer. Returns the number of characters copied if the event succeeds and -1 if an error occurs.

Usage

This event is triggered when the DataWindow has focus and the user selects Edit>Copy from the menu bar of a menu descended from the PFC m_master menu. It is also triggered by the m_edit.m_copy clicked event.

pfc_Cut

Description

Deletes selected text and stored it on the clipboard.

Return value

Integer. Returns the number of characters removed if the event succeeds and -1 if an error occurs.

Usage

This event is triggered when the DataWindow has focus and the user selects Edit>Cut from the menu bar of a menu descended from the PFC m_master menu. It is also triggered by the m_edit.m_cut clicked event.

pfc_DDCalculator

Description

Displays the dropdown calculator if the current column has been registered and uses a numeric data type.

Return value

Integer. Returns 1 if the function succeeds, 0 if the current DataWindow column has not been registered, and -1 if an error occurs.

Usage

Call this event to display the dropdown calculator programmatically. If the current DataWindow column has been registered and uses a numeric data type, the dropdown calculator displays.

The Dropdown event displays the dropdown calculator automatically for registered columns.

Examples

This example calls the pfc_DDCalculator event:

```
Integer  li_return

li_return = dw_1.Event pfc_DDCalculator( )
...
```

pfc_DDCalendar

Description

Displays the dropdown calendar if the current column has been registered and uses a Date data type.

Return value

Integer. Returns 1 if the function succeeds, 0 if the current DataWindow column has not been registered, and -1 if an error occurs.

Usage

Call this event to display the dropdown calendar programmatically. If the current DataWindow column has been registered and uses a Date data type, the dropdown calendar displays.

The Dropdown event displays the dropdown calendar automatically for registered columns.

Examples

This example calls the pfc_DDCalendar event:

```
Integer  li_return

li_return = dw_1.Event pfc_DDCalendar()
...
```

pfc_Debug

Description

Calls the pfc_Properties event, which opens the DataWindow Properties windo, using either the inv_properties instance variable or the snv_properties shared variable.

Usage

This event is called when the user selects DataWindow Properties from the m_dw popup menu.

pfc_DeleteRow

Description	Deletes either the current row or, if the row manager service is enabled, all selected rows. If the linkage service is enabled, this event calls that service's pfc_DeleteRow event.
Return value	Integer. Returns 1 if the event succeeds and -1 if an error occurs.
Usage	Call this event to delete one or more rows.

pfc_Descendant

Description	PFC events and functions may trigger this event to determine if the DataWindow control is inherited from u_dw.
Return value	Boolean. Always returns TRUE.
Usage	Internal.

pfc_FilterDlg

Description	Displays a filter dialog box by calling the n_cst_dwsrv_filter service's pfc_FilterDlg event.
Return value	Integer. Returns 1 if the event succeeds and -1 if an error occurs.
Usage	This event is called when the user selects View>Filter from the menu bar of a menu descended from the PFC m_master menu.

pfc_FindDlg

Description	Displays a find dialog box by calling the n_cst_dwsrv_find service's pfc_FindDlg event.
Return value	Integer. Returns 1 if the event succeeds and -1 if an error occurs.
Usage	This event is called when the user selects Edit>Find from the menu bar of a menu descended from the PFC m_master menu.

pfc_FirstPage

Description	Scrolls to the first page of the DataWindow.

Return value	Integer. Returns 1 if the event succeeds and -1 if an error occurs.
Usage	Call this event to scroll the DataWindow to the first page.

pfc_InsertRow

Description	Inserts a row just before the current row and calls the n_cst_dwsrv_linkage pfc_InsertRow event.
Return value	Integer. Returns the number of the inserted row if the event succeeds and -1 if an error occurs.
Usage	Call this event to insert a row.

pfc_LastPage

Description	Scrolls to the last page of the DataWindow.
Return value	Long. Returns the row number displayed at the top of the last page if the event succeeds and -1 if an error occurs.
Usage	Call this event to scroll the DataWindow to the last page.

pfc_NextPage

Description	Scrolls to the next page of the DataWindow.
Return value	Long. Returns the row number at the top of the page if the event succeeds and -1 if an error occurs.
Usage	Call this event to scroll the DataWindow to the next page.

pfc_Operators

Description	Calls the n_cst_dwsrv_querymode pfc_Operators event, which displays a w_selection dialog box containing operators.
Return value	Integer. Returns 1 if the event succeeds, 0 if the user cancels out of the dialog box, and -1 if an error occurs.
Usage	This event is called by the m_dw popup menu.

pfc_PageSetup

Description

Calls the pfc_PageSetupDlg event to display the Page Setup dialog box.

Return value

Integer. Returns 1 if the event succeeds, 0 if the user cancels out of the dialog box, and -1 if an error occurs.

Usage

This event is called when the user selects File>Page Setup from the menu bar of a menu descended from the PFC m_master menu.

pfc_PageSetupDlg

Description

Displays the Page Setup dialog box by calling the n_cst_platform of_PageSetupDlg function, passing the DataWindow's page display properties.

Syntax

instancename.Event **pfc_PageSetupDlg** (*attributes*)

Argument	Description
instancename	Instance name of u_dw
attributes	S_pagesetupattrib variable into which the event places page setup information. This argument is accessed through the *astr_pagesetup* argument (passed by reference)

Return value

Integer. Returns 1 if the event succeeds, 0 if the user cancels out of the dialog box, and -1 if an error occurs.

Usage

This event is called by the pfc_PageSetup event.

pfc_Paste

Description

Inserts (pastes) the contents of the clipboard at the insertion point.

Return value

Integer. Returns the number of characters that were pasted if the event succeeds and -1 if an error occurs.

Usage

This event is triggered when the DataWindow has focus and the user selects Edit>Paste from the menu bar of a menu descended from the PFC m_master menu. It is also triggered by the m_edit.m_paste Clicked event.

pfc_PopulateDDDW

Description

Empty user event that you extend to populate the specified DropDownDataWindow.

Syntax

instancename.Event **pfc_PopulateDDDW** (*column, dddw*)

Argument	Description
instancename	Instance name of u_dw
column	String specifying the column containing the dropdown DataWindow for which rows are retrieved. Access this value through the *as_colname* argument
dddw	DataWindowChild instance referencing the DropDownDataWindow column. This argument is accessed through the *adwc_obj* argument (passed by reference)

Return value

Long. Returns the number of rows populated if the event succeeds and -1 if an error occurs.

Usage

The code you add to this event can populate the DropDownDataWindow in any manner, including database retrieval, DataWindow caching, and retrieving from a file.

Examples

This example shows code you might add to the pfc_PopulateDDDW event:

```
IF adwc_obj.SetTransObject(SQLCA) = -1 THEN
   Return -1
ELSE
   Return adwc_obj.Retrieve()
END IF
```

pfc_PostUpdate

Description

Clears the DataWindow update flags..

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

You can extend this event to code additional post update processing.

Examples

This example is from the of_PostUpdate function:

```
...
If IsValid(inv_linkage) Then
   li_rc = inv_linkage.of_PostUpdate()
```

```
Else
   li_rc = this.Event pfc_postupdate()
End If
...
```

pfc_PreDeleteRow

Description

Calls the n_cst_dwsrv_linkage pfc_PreDeleteRow event to allow for linkage processing. This may include database updates as well as the prevention of deletion processing.

Syntax

instancename.Event **pfc_PreDeleteRow** ( )

Argument	Description
instancename	Instance name of u_dw

Return value

Return 1 to continue deletion, 0 to prevent deletion, and -1 if an error occurs.

Usage

You can extend this event to perform additional predelete processing.

pfc_PreFindDlg

Description

Empty user event allowing you to modify the properties passed to the pfc_FindDlg event.

Syntax

instancename.Event **pfc_PreFindDlg** (*attributes*)

Argument	Description
instancename	Instance name of u_dw
attributes	N_cst_findattrib variable into which the event places additional information. This argument is accessed through the *anv_findattrib* argument (passed by reference)

Usage

Use this event to modify or extend the information passed in the n_cst_findattrib object.

pfc_PreInsertRow

Description

Calls the n_cst_dwsrv_linkage pfc_PreInsertRow event to allow for linkage processing. This may include prevention of insert processing.

Syntax

instancename.Event **pfc_PreInsertRow** ( )

Argument	Description
instancename	Instance name of u_dw

Return value

Return 1 to continue insertion, 0 to prevent insertion, and -1 if an error occurs.

Usage

You can extend this event to perform additional preinsert processing.

pfc_PrePageSetupDlg

Description

Empty user event allowing you to modify the properties passed to the n_cst_platform of_PageSetupDlg function.

Syntax

instancename.Event **pfc_PrePageSetupDlg** (*attributes*)

Argument	Description
instancename	Instance name of u_dw
attributes	S_pagesetupattrib variable into which the event places additional page setup information. This argument is accessed through the *astr_pagesetup* argument (passed by reference)

Usage

This event is called by the pfc_PageSetupDlg event before calling the n_cst_platform of_PageSetupDlg function.

You can use this event to modify or extend the information passed in the s_pagesettupattrib structure.

Examples

This example contains code you might add to the pfc_PrePageSetupDlg event:

```
// Sets page setup to portrait
astr_pagesetup.b_portraitorientation = TRUE
```

pfc_PrePrintDlg

Description

Empty user event allowing you to modify the properties passed to the n_cst_platform of_PrintDlg function.

Syntax

instancename.Event **pfc_PrePrintDlg** (*attributes*)

Argument	Description
instancename	Instance name of u_dw

Argument	Description
attributes	S_printdlgattrib variable into which the event places additional printing information. This argument is accessed through the *astr_printdlg* argument (passed by reference)

Return value

None

Usage

This event is called by pfc_PrintDlg before calling the n_cst_platform of_PrintDlg function.

You can use this event to modify or extend the information passed in the s_printdlgattrib structure.

Examples

This example contains code you might add to the pfc_PrePrintDlg event:

```
// Default copies to 1
astr_printdlg.l_copies = 1
```

pfc_PreProperties

Description

Empty user event allowing you to modify the properties passed to the DataWindow Properties dialog box.

Syntax

instancename.Event **pfc_PreProperties** (*attributes*)

Argument	Description
instancename	Instance name of u_dw
attributes	N_cst_dwpropertyattrib variable into which the event places additional information. This argument is accessed through the *anv_dwpropertyattrib* argument (passed by reference)

Usage

This event is called by the pfc_Debug event before calling the n_cst_property of_OpenProperty function.

You can use this event to modify or extend the information passed in the n_cst_propertyattrib object.

pfc_PreReplaceDlg

Description

Empty user event allowing you to modify the properties passed to the pfc_ReplaceDlg event.

Syntax

instancename.Event **pfc_PreReplaceDlg** (*attributes*)

Argument	Description
instancename	Instance name of u_dw
attributes	N_cst_findattrib variable into which the event places additional information. This argument is accessed through the *anv_findattrib* argument (passed by reference)

Usage

Use this event to modify or extend the information passed in the n_cst_findattrib object.

pfc_PreRestoreRow

Description

Prepares the w_restorerow dialog box for display.

Syntax

instancename.Event **pfc_PreRestoreRow** (*attributes*)

Argument	Description
instancename	Instance name of u_dw
attributes	N_cst_restorerowattrib variable into which the event places additional information. This argument is accessed through the *anv_restorerowattrib* argument (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

The n_cst_rowmanager of_UnDelete function calls this event before opening the w_restorerow window. It adds filter and sort information to the n_cst_restorerowattrib structure.

You can extend this event to modify the information passed to the w_restorerow window.

Examples

This example is from the n_cst_dwsrv_rowmanager of_UnDelete function:

```
n_cst_restorerowattrib   lnv_restorerowattrib

lnv_restorerowattrib.idw_active = idw_requestor
idw_requestor.Event pfc_PreRestoreRow &
   (lnv_restorerowattrib)
OpenWithParm(w_restorerow, lnv_restorerowattrib)
...
```

pfc_PreRMBMenu

Description

Empty user event allowing you to modify m_dw contents before display.

Syntax

instancename.Event **pfc_PreRMBMenu** (*editmenu*)

Argument	Description
instancename	Instance name of u_dw
editmenu	M_dw variable containing the popup menu to be displayed. This argument, which is accessed through the *am_dw* argument, is passed by reference

Usage

Optionally add logic to this event to selectively enable and disable m_edit menu items.

Examples

This example contains code you might add to the pfc_PreRMBMenu event:

```
// Always disable delete
am_dw.m_delete.Enabled = FALSE
```

pfc_PreUpdate

Description

Empty user event called by pfc_Update before updating rows.

Syntax

instancename.Event **pfc_PreUpdate** ( )

Argument	Description
instancename	Instance name of u_dw

Return value

Return 1 if the event succeeds and -1 to terminate update processing.

Usage

Extend this event to add site-specific preupdate processing.

Examples

This example contains code you might add to the pfc_PreUpdate event:

```
// Sets page setup to portrait
astr_pagesetup.b_portraitorientation = TRUE
```

pfc_PreviousPage

Description

Scrolls to the previous page of the DataWindow.

Return value

Long. Returns the row number at the top of the page if the event succeeds and -1 if an error occurs.

Usage

Call this event to scroll the DataWindow to the previous page.

pfc_Print

Description

Calls the pfc_PrintDlg function and prints the DataWindow, as specified in the Print dialog box.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is triggered when the DataWindow has focus and the user selects File>Print from the menu bar of a menu descended from the PFC m_master menu.

pfc_PrintDlg

Description

Initializes the s_printdlgattrib structure with the DataWindow's current settings, displays the Print dialog box by calling the n_cst_platform of_PrintDlg function, and resets the DataWindow's settings, as specified by the user.

Syntax

instancename.Event **pfc_PrintDlg** (*attributes*)

Argument	Description
instancename	Instance name of u_dw
attributes	S_printdlgattrib variable into which the event places printing information. This argument, which is accessed through the *astr_printdlg* argument, is passed by reference

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is called by the pfc_Print event.

pfc_PrintImmediate

Description

Prints the current DataWindow without displaying the Print dialog box.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is called when the DataWindow has focus and the user selects File>Print Immediate from the menu bar of a menu descended from the PFC m_master menu.

Hiding Print Immediate
Consider hiding the Print Immediate menu item but enabling its toolbar button. This provides functionality equivalent to many current software packages.

pfc_PrintPreview

Description — Toggles the DataWindow between preview and edit modes.

Return value — Boolean. Returns TRUE if the DataWindow is placed in preview mode and FALSE if the DataWindow is placed in edit mode.

Usage — To use this event, you must enable the print preview service by calling the of_SetPrintPreview function.

This event is called when the DataWindow has focus and the user selects File>Print Preview from a menu that descends from the PFC m_master menu.

pfc_ReplaceDlg

Description — Displays a Replace dialog box by calling the n_cst_dwsrv_find service's pfc_ReplaceDlg event.

Return value — Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage — This event is called when the user selects View>Replace from the menu bar of a menu descended from the PFC m_master menu.

pfc_ResetUpdate

Description — Clears the DataWindow's update flags.

Return value — Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage — This event is called as part of the default pfc_Save process.

pfc_RestoreRow

Description — Calls the row manager pfc_RestoreRow event, which displays the w_restorerow dialog box.

Return value: Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage: This event is called by the m_dw popup menu.

pfc_Retrieve

Description: Empty user event to contain all database retrieve logic. PFC calls this event for all database retrieves.

Return value: Long. Returns the result of the Retrieve function.

Usage: Add code to this event to retrieve rows for the DataWindow.

This event is called by the u_dw of_Retrieve function and by the n_cst_dwsrv_linkage of_Retrieve function.

Examples: This example shows the code you add to this event:

```
Return this.Retrieve()
```

pfc_RetrieveDDDW

Description: Obsolete. Use pfc_PopulateDDDW instead.

Syntax: *instancename.Event* **pfc_RetrieveDDDW** (*column*)

Argument	Description
instancename	Instance name of u_dw
column	String specifying the column containing the dropdown DataWindow for which rows are retrieved. Access this value through the *as_column* argument

Return value: Long. Returns the result of the Retrieve function.

pfc_RowChanged

Description: Calls the linkage service's pfc_RowFocusChanged function.

Usage: This event is called whenever the DataWindow buffer has been sorted, filtered, or otherwise modified such that the current row has not changed but the actual row at that location is different.

pfc_RowValidation

Description

Empty user event that you extend to perform site-specific row validation.

Syntax

instancename.Event **pfc_RowValidation** (*row*)

Argument	Description
instancename	Instance name of u_dw
row	Long specifying the row to validate. Access this value through the *al_row* argument

Return value

Integer. Returns 1 if validation succeeds and -1 if an error occurs.

Usage

Extend this event to perform additional row validation.

The n_cst_dwsrv_linkage of_Save function calls this event.

pfc_Ruler

Description

Toggles the DataWindow between displaying and hiding rulers in print preview mode.

Return value

Boolean. Returns TRUE if print preview rulers are displayed and FALSE if they are hidden.

Usage

To use this event, you must enable the print preview service by calling the of_SetPrintPreview function.

This event is called when the DataWindow has focus and the user selects View>Ruler from the menu bar of a menu descended from the PFC m_master menu.

pfc_SelectAll

Description

Selects all text in the current DataWindow cell.

Return value

Integer. Returns the number of characters selected if the event succeeds and -1 if an error occurs.

Usage

This event is triggered when the DataWindow has focus and the user selects Edit>Select All from the menu bar of a menu descended from the PFC m_master menu.

pfc_SortDlg

Description

Displays a sort dialog box by calling the n_cst_dwsrv_sort service's pfc_SortDlg event.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is called when the user selects View>Sort from the menu bar of a menu descended from the PFC m_master menu.

pfc_Undo

Description

Cancels the last edit to the DataWindow.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs or there is nothing to undo.

Usage

This event is triggered when the DataWindow has focus and the user selects Edit>Undo from the menu bar of a menu descended from the PFC m_master menu.

pfc_Update

Description

Updates the DataWindow. If the multitable update service is enabled, this event updates all specified tables.

Syntax

instancename.Event **pfc_Update** (*accepttext, resetflags*)

Argument	Description
instancename	Instance name of u_dw
accepttext	Boolean specifying whether the DataWindow control should automatically perform an AcceptText before performing the update (TRUE) or not (FALSE)
resetflags	Boolean specifying whether the DataWindow control should automatically reset the update flags (TRUE) or not (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

This event calls the pfc_PreUpdate event before updating rows.

pfc_UpdatePrep

Description	Empty user event to which you can add code that prepares for update.
Return value	Long. Return 1 if the update preparation succeeds and -1 to halt the update process.
Usage	The of_UpdatePrep function calls this function.

pfc_UpdatesPending

Description	Determines if there are pending updates for the DataWindow.
Return value	Integer. Returns 1 if there are pending updates for the DataWindow and 0 if there are no pending updates.
Usage	This event is called by the of_UpdatesPending function and by the n_cst_dwsrv_linkage of_GetUpdatesPending function.

pfc_Validation

Description	Checks for required fields.
Return value	Integer. Returns 1 if the event succeeds and -1 if an error occurs.
Usage	This event is called by the of_Validation function. You can extend this event to perform additional validation.

pfc_Values

Description	Calls the n_cst_dwsrv_querymode pfc_Values event, which displays a w_selection dialog box containing a list of values.
Return value	Integer. Returns 1 if the event succeeds, 0 if the user cancels out of the dialog box, and -1 if an error occurs.
Usage	This event is called by the m_dw popup menu.

pfc_Zoom

Description	Displays the w_zoom dialog box, allowing the user to control DataWindow display while in print preview mode.

Return value

Integer. Returns the zoom level chosen by the user if the event succeeds, 0 if the user cancels out of the w_zoom dialog box, and -1 if an error occurs.

Usage

To use this event, you must enable the print preview service by calling the of_SetPrintPreview function.

This event is called when the DataWindow has focus and the user selects View>Zoom from the menu bar of a menu descended from the PFC m_master menu.

RButtonDown

Description

Calls the linkage service and row selection service's pfc_RButtonDown event.

Usage

This event executes when the user presses the right mouse button over a DataWindow. The n_cst_dwsrv_linkage pfc_RButtonDown event can cancel right-click processing.

RButtonUp

Description

Displays the m_dw popup menu.

Usage

This event executes when the user releases the right mouse button over the DataWindow control.

Resize

Description

Calls the DataWindow resize service's pfc_Resize event, if the service is enabled.

Usage

This event executes when the DataWindow control resizes, either because it is itself resizable or because it is registered as resizable with the window resize service.

RetrieveEnd

Description

Calls the linkage service's pfc_RetrieveEnd event, if the service is enabled.

Usage

This event executes when a retrieval process completes.

RetrieveStart

Description — Calls the linkage service's pfc_RetrieveStart event, if the service is enabled.

Usage — This event executes when a retrieval process begins.

RowFocusChanged

Description — Used by the linkage service, if enabled, to coordinate scrolling among the linkage chain.

Usage — This event executes when the DataWindow first displays and again when the user changes rows.

This event calls the n_cst_dwsrv_linkage pfc_RowFocusChanged event.

SQLPreview

Description — Calls SQL Spy functions, if enabled for the application.

Usage — This event executes just before accessing the database.

Functions

U_dw includes precoded object functions to control DataWindow services:

of_AcceptText	of_SetMultiTable
of_CheckRequired	of_SetPrintPreview
of_GetParentWindow	of_SetProperty
of_GetUpdateable	of_SetQuerymode
of_IsRoot	of_SetReport
of_IsSharedProperty	of_SetReqColumn
of_IsUpdateable	of_SetResize
of_MessageBox	of_SetRowManager
of_PostUpdate	of_SetRowSelect
of_Reset	of_SetSharedProperty
of_Retrieve	of_SetSort
of_SetBase	of_SetTransObject
of_SetDropDownCalculator	of_SetUpdateable
of_SetDropDownCalendar	of_SetUpdateRequestor
of_SetDropDownSearch	of_Update
of_SetFilter	of_UpdatePrep

of_SetFind	of_UpdatesPending
of_SetLinkage	of_Validation

of_AcceptText

Description

Performs an AcceptText function for the DataWindow.

Access

Public

Syntax

instancename.**of_AcceptText** (*focusonerror*)

Argument	Description
instancename	Instance name of u_dw
focusonerror	Boolean indicating whether PFC sets focus to the DataWindow when an error occurs (TRUE) or not (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize accept text processing, extend the pfc_AcceptText event.

Examples

This example is from the n_cst_luw of_AcceptText function:

```
...
If lb_defined Then
   li_rc =  &
      lpo_tocheck.Function Dynamic of_AcceptText &
         (ab_focusonerror)
   If li_rc < 0 Then Return -1

...
```

of_CheckRequired

Description

Determines if any required columns contain NULL values.

Access

Public

Syntax

instancename.**of_CheckRequired** (*buffer, row, column, colname, updateonly*)

Argument	Description
instancename	Instance name of u_dw
buffer	DWBuffer enumerated data type specifying the DataWindow buffer to check
row	Long specifying the first row to check and into which the function places the number of the row in error (passed by reference)
column	Integer specifying the first column to check and into which the function places the number of the column in error (passed by reference)
colname	String specifying the first column to check and into which the function places the name of the column in error (passed by reference)
updateonly	Boolean indicating whether to validate only those rows and columns that have changed (TRUE) or validate all rows and columns

Return value

Integer. Returns 1 if the function succeeds and -1 if there is no parent window.

Usage

The u_dw pfc_Validation event calls this function.

Examples

This example is from the u_dw pfc_Validation event:

```
...
li_rc = &
   of_CheckRequired(Primary!, ll_checkrow, &
      li_checkcolumn, ls_checkcolname, &
         ib_updateonly)
...
```

of_GetParentWindow

Description

Retrieves a reference to the parent window.

Access

Public

Syntax

instancename.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_dw

Argument	Description
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if there is no parent window.

Usage

The u_dw GetFocus event calls this function.

Examples

This example is from the u_dw GetFocus event:

```
...
of_GetParentWindow(lw_parent)
If IsValid(lw_parent) Then
...
```

of_GetUpdateable

Description

Obsolete. Call of_IsUpdateable instead.

Access

Public

Syntax

instancename.**of_GetUpdateable** ( )

Argument	Description
instancename	Instance name of u_dw

Return value

Boolean. Returns TRUE if the DataWindow is updatable and FALSE if it is not.

of_IsRoot

Description

Reports whether the DataWindow is a root DataWindow. A root DataWindow does not have a master DataWindow.

Access

Public

Syntax

instancename.**of_IsRoot** ( )

Argument	Description
instancename	Instance name of u_dw

Return value

Boolean. Returns TRUE if the DataWindow is a root and FALSE if it is not.

Examples

This example calls the of_IsRoot function:

```
IF dw_1.of_IsRoot() THEN
   MessageBox("dw_1", "DW is a root")
ELSE
   MessageBox("dw_1", "DW is not a root")
END IF
```

of_IsSharedProperty

Description

Reports whether the DataWindow shared property service is enabled.

Access

Public

Syntax

instancename.**of_IsSharedProperty ( )**

Argument	Description
instancename	Instance name of u_dw

Return value

Boolean. Returns TRUE if the DataWindow shared property service is enabled and FALSE if it is not.

Examples

This example calls the of_IsSharedProperty function:

```
IF dw_1.of_IsSharedProperty() THEN
   MessageBox("DW", "Shared property enabled")
ELSE
   MessageBox("DW", "Shared property disabled")
END IF
```

of_IsUpdateable

Description

Reports whether the DataWindow is updatable.

Access

Public

Syntax

instancename.**of_IsUpdateable ( )**

Argument	Description
instancename	Instance name of u_dw

Return value

Boolean. Returns TRUE if the DataWindow is updatable and FALSE if it is not.

Usage

The pfc_UpdatesPending event calls this function.

Examples

This example is from the pfc_UpdatesPending event:

```
...
IF NOT of_IsUpdateable( ) THEN
   Return 0
END IF
...
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of u_dw
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!

Argument	Description
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in DataWindow controls.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('dw_dberror','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

of_PostUpdate

Description

Calls the pfc_PostUpdate event, which clears update flags and allows you to code additional post-update processing.

Access

Public

Syntax

instancename.**of_PostUpdate** ( )

Argument	Description
instancename	Instance name of u_dw

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize post update processing, extend the pfc_PostUpdate event.

Examples

This example is from the n_cst_luw of_PostUpdate function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_PostUpdate()
...
```

of_Reset

Description

Resets either a single DataWindow or all DataWindows in a linkage chain.

Access

Public

Syntax

instancename.**of_Reset ( )**

Argument	Description
instancename	Instance name of u_dw

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_Reset function:

```
...
dw_1.of_Reset()
...
```

of_Retrieve

Description

Retrieves rows for either a single DataWindow or all DataWindows in a linkage chain. This function calls the pfc_Retrieve event.

Access

Public

Syntax

instancename.**of_Retrieve ( )**

Argument	Description
instancename	Instance name of u_dw

Return value

Integer. Returns the number of rows retrieved if the function succeeds and -1 if an error occurs.

Usage

Call this function to retrieve rows in a DataWindow. You code the actual Retrieve function in the pfc_Retrieve event.

Examples

This example calls the of_Retrieve function:

```
dw_emplist.of_Retrieve()
```

of_SetBase

Description

Enables or disables n_cst_dwsrv, which provides basic DataWindow services.

Access

Public

Syntax

instancename.**of_SetBase** (*boolean*)

Argument	Description
instancename	Instance name of u_dw
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) basic DataWindow services

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_dwsrv. This instance is named inv_base.

Because all DataWindow services are descendants of n_cst_dwsrv (and have n_cst_dwsrv functions available to them), use this object when you require basic DataWindow services only.

Examples

This example calls the of_SetBase function to enable basic DataWindow services:

```
dw_employee.of_SetBase(TRUE)
```

of_SetDropDownCalculator

Description

Enables or disables u_calculator, which provides a dropdown calculator for registered columns.

Access

Public

Syntax

instancename.**of_SetDropDownCalculator** (*boolean*)

Argument	Description
instancename	Instance name of u_dw
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the dropdown calculator service

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Call this function to enable or disable the dropdown calculator. The u_dw instance variable for this service is iuo_calculator.

Examples

This example calls the of_SetDropDownCalculator function to enable an instance of the dropdown calculator:

```
dw_employee.of_SetDropDownCalculator(TRUE)
```

of_SetDropDownCalendar

Description

Enables or disables u_calendar, which provides a dropdown calendar for registered columns.

Access

Public

Syntax

instancename.**of_SetDropDownCalendar** (*boolean*)

Argument	Description
instancename	Instance name of u_dw
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the dropdown calendar service

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Call this function to enable or disable the dropdown calendar. The u_dw instance variable for this service is iuo_calendar.

Examples

This example calls the of_SetDropDownCalendar function to enable an instance of the dropdown calendar:

```
dw_employee.of_SetDropDownCalendar(TRUE)
```

of_SetDropDownSearch

Description

Enables or disables n_cst_dwsrv_dropdownsearch, which provides dropdown DataWindow search services.

Access

Public

Syntax

instancename.**of_SetDropDownSearch** (*boolean*)

Argument	Description
instancename	Instance name of u_dw
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the dropdown DataWindow search service

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Call this function to enable or disable DropDownDataWindow search services. The u_dw instance variable for DropDownDataWindow search services is inv_dropdownsearch.

Examples

This example calls the of_SetDropDownSearch function to enable the dropdown DataWindow search service:

```
dw_employee.of_SetDropDownSearch(TRUE)
```

of_SetFilter

Description

Controls or destroys an instance of n_cst_dwsrv_filter, which provides filtering services.

Access

Public

Syntax

instancename.**of_SetFilter** (*boolean*)

Argument	Description
instancename	Instance name of u_dw
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the filter service

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_dwsrv_filter. This instance is named inv_filter.

Examples

This example calls the of_SetFilter function to enable the filter service:

```
dw_employee.of_SetFilter(TRUE)
```

of_SetFind

Description

Enables or disables n_cst_dwsrv_find, which provides the DataWindow find service.

Access

Public

Syntax

instancename.**of_SetFind** (*boolean*)

Argument	Description
instancename	Instance name of u_dw
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the DataWindow find service

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Call this function to enable or disable the DataWindow find service. The u_dw instance variable for the DataWindow find service is inv_find.

Examples

This example calls the of_SetFind function to enable the DataWindow find service:

```
dw_employee.of_SetFind(TRUE)
```

of_SetLinkage

Description

Enables or disables an instance of n_cst_dwsrv_linkage, which provides master/detail services.

Access

Public

Syntax

instancename.**of_SetLinkage** (*boolean*)

Argument	Description
instancename	Instance name of u_dw

Argument	Description
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the linkage service

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_dwsrv_linkage. This instance is named inv_linkage.

Examples

This example calls the of_SetLinkage function to enable the linkage service:

```
dw_employee.of_SetLinkage(TRUE)
```

of_SetMultiTable

Description

Enables or disables an instance of n_cst_dwsrv_multitable, which provides multitable update services.

Access

Public

Syntax

instancename.**of_SetMultiTable** (*boolean*)

Argument	Description
instancename	Instance name of u_dw
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the multitable update service

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_dwsrv_multitable. This instance is named inv_multitable.

Examples

This example calls the of_SetMultiTable function to enable the multitable update service:

```
dw_employee.of_SetMultiTable(TRUE)
```

of_SetPrintPreview

Description

Enables or disables an instance of n_cst_dwsrv_printpreview, which provides the print preview service.

Access

Public

Syntax

instancename.**of_SetPrintPreview** (*boolean*)

Argument	Description
instancename	Instance name of u_dw
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the print preview service

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_dwsrv_printpreview. This instance is named inv_printpreview.

Examples

This example calls the of_SetPrintPreview function to enable the print preview service:

```
dw_employee.of_SetPrintPreview(TRUE)
```

of_SetProperty

Description

Enables or disables an instance of n_cst_dwsrv_property, which provides the DataWindow property service.

Access

Public

Syntax

instancename.**of_SetProperty** (*boolean*)

Argument	Description
instancename	Instance name of u_dw
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the DataWindow property service

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_dwsrv_property. This instance is named inv_property.

Examples

This example calls the of_SetProperty function to enable the DataWindow property service:

```
dw_employee.of_SetProperty(TRUE)
```

of_SetQuerymode

Description

Enables or disables an instance of n_cst_dwsrv_querymode, which provides querymode services.

Access

Public

Syntax

instancename.**of_SetQuerymode** (*boolean*)

Argument	Description
instancename	Instance name of u_dw
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the querymode service

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_dwsrv_querymode. This instance is named inv_querymode.

Examples

This example calls the of_SetQuerymode function to enable the querymode service:

```
dw_employee.of_SetQuerymode(TRUE)
```

of_SetReport

Description

Enables or disables an instance of n_cst_dwsrv_report, which provides reporting services.

Access

Public

Syntax

instancename.**of_SetReport** (*boolean*)

Argument	Description
instancename	Instance name of u_dw
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the report service

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_dwsrv_report. This instance is named inv_report.

Examples

This example calls the of_SetReport function to enable the reporting service:

```
dw_employee.of_SetReport(TRUE)
```

of_SetReqColumn

Description

Enables or disables an instance of n_cst_dwsrv_reqcolumn, which provides DataWindow required-field services.

Access

Public

Syntax

instancename.**of_SetReqColumn** (*boolean*)

Argument	Description
instancename	Instance name of u_dw
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the required column service

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_dwsrv_reqcolumn. This instance is named inv_reqcolumn.

Examples

This example calls the of_SetReqColumn function to enable the required column service:

```
dw_employee.of_SetReqColumn(TRUE)
```

of_SetResize

Description

Enables or disables an instance of n_cst_dwsrv_resize, which provides the DataWindow resize service.

Access

Public

Syntax

instancename.**of_SetResize** (*boolean*)

Argument	Description
instancename	Instance name of u_dw
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the DataWindow resize service

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_dwsrv_resize. This instance is named inv_resize.

Examples

This example calls the of_SetResize function to enable the DataWindow resize service:

```
dw_employee.of_SetResize(TRUE)
```

of_SetRowManager

Description

Enables or disables an instance of n_cst_dwsrv_rowmanager, which provides row management services.

Access

Public

Syntax

instancename.**of_SetRowManager** (*boolean*)

Argument	Description
instancename	Instance name of u_dw
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the row management service

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_dwsrv_rowmanager. This instance is named inv_rowmanager.

Examples

This example calls the of_SetRowManager function to enable the row management service:

```
dw_employee.of_SetRowManager(TRUE)
```

of_SetRowSelect

Description

Enables or disables an instance of n_cst_dwsrv_rowselection, which provides row-selection services.

Access

Public

Syntax

instancename.**of_SetRowSelect** (*boolean*)

Argument	Description
instancename	Instance name of u_dw
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the row selection service

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_dwsrv_rowselection. This instance is named inv_rowselect.

Examples

This example calls the of_SetRowSelect function to enable the row selection service:

```
dw_employee.of_SetRowSelect(TRUE)
```

of_SetSharedProperty

Description

Enables or disables a shared instance of n_cst_dwsrv_property, which provides the DataWindow property service.

Access

Public

Syntax

instancename.**of_SetSharedProperty** (*boolean*)

Argument	Description
instancename	Instance name of u_dw
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the DataWindow property service

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy a shared instance of n_cst_dwsrv_property. This instance is named snv_property.

Examples

This example calls the of_SetSharedProperty function to enable the DataWindow property service:

```
dw_employee.of_SetSharedProperty(TRUE)
```

of_SetSort

Description

Enables or disables an instance of n_cst_dwsrv_sort, which provides sorting services.

Access

Public

Syntax

instancename.**of_SetSort** (*boolean*)

Argument	Description
instancename	Instance name of u_dw
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the sort service

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_dwsrv_sort. This instance is named inv_sort.

Examples

This example calls the of_SetSort function to enable the sort service:

```
dw_employee.of_SetSort(TRUE)
```

of_SetTransObject

Description

Sets the transaction object for either a single DataWindow or all DataWindows in a linkage chain. This function also initializes instance variables for use by DataWindow services.

Access

Public

Syntax

instancename.**of_SetTransObject** (*transaction*)

Argument	Description
instancename	Instance name of u_dw
transaction	N_tr variable specifying the Transaction object to use for the DataWindow

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function to establish a Transaction object for a DataWindow that uses DataWindow services.

Examples

This example calls the of_SetTransObject function. It assumes you have associated n_tr with SQLCA in the Application painter:

```
dw_employee.of_SetTransObject(SQLCA)
```

of_SetUpdateable

Description

Specifies whether the DataWindow is updatable.

Access

Public

Syntax

instancename.**of_SetUpdateable** (*boolean*)

Argument	Description
instancename	Instance name of u_dw
boolean	Boolean indicating whether the DataWindow is updatable. All updatable DataWindows are included in w_master's default save processing

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function to disable default save processing for DataWindows that are not updatable.

Shared DataWindows
If your application uses shared data, be sure that only one of the DataWindows using the shared data is declared as updatable.

Examples

This example disables default save processing for the dw_report DataWindow:

```
dw_report.of_SetUpdateable(FALSE)
```

of_SetUpdateRequestor

Description

Creates a reference to the object requesting an update within a logical unit of work.

Access

Public

Syntax

instancename.**of_SetUpdateRequestor** (*requestor*)

Argument	Description
instancename	Instance name of u_dw

Argument	Description
requestor	PowerObject containing the object requesting the update

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the of_Update function:

```
...
Else
  If this.of_SetUpdateRequestor(apo_requestor) <0 Then
    Return FAILURE
  End If
  li_rc = this.of_Update(ab_accepttext, ab_resetflag)
...
```

of_Update

Updates either a single DataWindow or all DataWindows in a linkage chain. There are two syntaxes:

To	Use
Update data, optionally controlling update type	Syntax 1
Update data passing the requestor object	Syntax 2

Syntax 1

Update data, optionally controlling update type

Description

Updates rows in the DataWindow. With this syntax, you can optionally control the update types:

- Insert
- Update
- Delete

Access

Public

Syntax

instancename.**of_Update** (*accepttext*, *resetflags* {, *insert*, *update*, *delete* })

Argument	Description
instancename	Instance name of u_dw

Argument	Description
accepttext	Boolean indicating whether the DataWindow should automatically perform an AcceptText before performing the update: TRUE—Perform AcceptText (default) FALSE—Do not perform AcceptText
resetflags	Boolean indicating whether *instancename* should automatically reset the update flags: TRUE—Reset the flags (default) FALSE—Do not reset the flags
insert (optional)	Boolean indicating whether to insert rows: TRUE—Insert rows (default) FALSE—Do not insert rows
update (optional)	Boolean indicating whether to modify changed rows: TRUE—Modify rows (default) FALSE—Do not modify rows
delete (optional)	Boolean indicating whether to delete rows: TRUE—Delete rows (default) FALSE—Do not delete rows

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

This function calls the pfc_Update event.

FOR INFO For more information on update flags, see the Update function in the *PowerScript Reference*.

Examples

This example calls the of_Update function:

```
Integer  li_return

li_return = dw_emp.of_Update(TRUE, TRUE)
```

Syntax 2 **Update data passing the requestor object**

Description

Updates rows in the DataWindow, passing a reference to the requestor object.

Access

Public

Syntax

instancename.**of_Update** (*accepttext*, *resetflags*, *requestor*)

Argument	Description
instancename	Instance name of u_dw
accepttext	Boolean indicating whether the DataWindow should automatically perform an AcceptText before performing the update: TRUE—Perform AcceptText (default) FALSE—Do not perform AcceptText
resetflags	Boolean indicating whether *instancename* should automatically reset the update flags: TRUE—Reset the flags (default) FALSE—Do not reset the flags
requestor	PowerObject containing the requestor object

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize update processing, extend the pfc_Update event.

Examples

This example is from the n_cst_luw of_Update function:

```
...
If lb_defined Then
  li_rc = lpo_tocheck.Function Dynamic of_Update &
    (ab_accepttext,  ab_resetflag, &
      lpo_updaterequestor)
  If li_rc < 0 Then Return -1
  Continue
End If
...
```

of_UpdatePrep

Description

Perform update preparation for either a single DataWindow or all DataWindows in a linkage chain. This function calls the pfc_UpdatePrep event, which allows you to code additional update preparation logic.

Access

Public

Syntax

instancename.**of_UpdatePrep** ( )

Argument	Description
instancename	Instance name of u_dw

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize update preparation processing, extend the pfc_UpdatePrep event.

Examples

This example is from the n_cst_luw of_UpdatePrep function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_UpdatePrep()
  If li_rc < 0 Then Return -1
  Continue
End If
...
```

of_UpdatesPending

Description

Determines if there are updates pending for either a single DataWindow or all DataWindows in a linkage chain.

Access

Public

Syntax

instancename.**of_UpdatesPending ( )**

Argument	Description
instancename	Instance name of u_dw

Return value

Integer. Returns values as follows:

- **1** Updates are pending
- **0** No updates pending
- **-1** An error occurred

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize pending updates processing, extend the pfc_UpdatesPending event.

Examples

This example calls the of_UpdatesPending function:

```
...
If lb_defined Then
   la_rc = lpo_tocheck.Dynamic of_UpdatesPending ()
...
```

of_Validation

Description

Performs validation for either a single DataWindow or all DataWindows in a linkage chain.

Access

Public

Syntax

instancename.**of_Validation** ( )

Argument	Description
instancename	Instance name of u_dw

Return value

Integer. Returns 1 if the function succeeds and -1 if a validation error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize validation processing, extend the pfc_Validation event.

Examples

This example calls the of_Validation function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_Validation()
...
```

u_em

Description

EditMask visual user object ancestor.

Ancestry

pfc_u_em
u_em
Events

Destructor	pfc_DDCalendar
GetFocus	pfc_Paste
pfc_Clear	pfc_PreRMBMenu
pfc_Copy	pfc_SelectAll
pfc_Cut	pfc_Undo
pfc_DDCalculator	RButtonUp

Object functions

Public	of_GetParentWindow	Public	of_SetDropdownCalculator
Protected	of_MessageBox	Public	of_SetDropdownCalendar

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

PFC visual user objects are designed to be used with windows that are descendants of w_master.

Usage

Use this visual object in windows instead of the standard PowerBuilder EditMask. U_em event scripts provide integration with PFC menus. Additionally, u_em supports:

- **Cut, copy, paste, clear, and select all** These are triggered automatically by the message router.
- **Right mouse button support** The RButtonUp event enables you to use the right mouse button to perform editing actions. To disable right mouse button support, set ib_rmbmenu to FALSE in the edit mask's Constructor event.
- **Autoselect** This means that text is selected when a user tabs to the edit mask. To disable autoselect, set ib_autoselect to FALSE in the edit mask's Constructor event.
- **Dropdown calculator** You can enable the display of u_calculator, the dropdown calculator, to enable enhanced input for numeric fields.
- **Dropdown calendar** You can enable the display of u_calendar, the dropdown calendar, to enable enhanced input for date fields.

See also

m_edit
u_calculator
u_calendar

Instance variables

U_em has instance variables:

Instance variable	Description	Data type	Ancestry	Usage
ib_autoselect	Indicates whether PFC selects text when the control receives focus	Boolean	Protected	Set to FALSE to disable autoselect
ib_rmbmenu	Indicates whether the m_edit menu displays when the user presses the right mouse button	Boolean	Protected	Set to FALSE to disable right mouse button support
iuo_calculator	Reference variable for dropdown calculator	u_calculator	Public	Use in dot notation to access u_calculator events, functions, and attributes
iuo_calendar	Reference variable for dropdown calendar	u_calendar	Public	Use in dot notation to access u_calendar events, functions, and attributes

Events

U_em includes precoded events:

Destructor	pfc_DDCalendar
GetFocus	pfc_Paste
pfc_Clear	pfc_PreRmbMenu
pfc_Copy	pfc_SelectAll
pfc_Cut	pfc_Undo
pfc_DDCalculator	RButtonUp

Destructor

Description — Destroys instances of u_calculator and u_calendar.

Usage — This event executes when the DataWindow closes.

GetFocus

Description — Updates the parent window so it can set MicroHelp.

Usage — This event calls the pfc_ControlGotFocus event in the parent.

pfc_Clear

Description — Deletes selected text.

Return value — Integer. Returns the number of characters deleted if the event succeeds and -1 if an error occurs.

Usage — The message router calls this event when an EditMask based on u_em has focus and the user selects Edit>Clear.

pfc_Copy

Description — Copies selected text to the clipboard.

Return value — Integer. Returns the number of characters copied if the event succeeds and -1 if an error occurs.

Usage — The message router calls this event when an EditMask based on u_em has focus and the user selects Edit>Copy.

This event is also called by the m_edit popup menu.

pfc_Cut

Description — Deletes selected text and stores it on the clipboard.

Return value — Integer. Returns the number of characters removed if the event succeeds and -1 if an error occurs.

Usage — The message router calls this event when an EditMask based on u_em has focus and the user selects Edit>Cut.

This event is also called by the m_edit popup menu.

pfc_DDCalculator

Description — Displays the dropdown calculator.

Return value — Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this event to display the dropdown calculator.

Examples

This example calls the pfc_DDCalculator event:

```
Integer  li_return

li_return = em_1.Event pfc_DDCalculator()
...
```

pfc_DDCalendar

Description

Displays the dropdown calendar.

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this event to display the dropdown calendar.

Examples

This example calls the pfc_DDCalendar event:

```
Integer  li_return

li_return = em_1.Event pfc_DDCalendar()
...
```

pfc_Paste

Description

Inserts (pastes) the contents of the clipboard at the insertion point.

Return value

Integer. Returns the number of characters that were pasted if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when an EditMask based on u_em has focus and the user selects Edit>Paste.

This event is also called by the m_edit popup menu.

pfc_PreRmbMenu

Description

User event allowing you to modify m_edit contents before display.

Syntax

instancename.Event **pfc_PreRmbMenu** (*editmenu*)

Argument	Description
instancename	Instance name of u_em

Argument	Description
editmenu	M_edit variable containing the popup menu to be displayed (passed by reference)

Return value

None

Usage

Optionally add logic to this event to selectively enable and disable m_edit menu items.

pfc_SelectAll

Description

Selects all text in the EditMask.

Return value

Integer. Returns the number of characters selected if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when an EditMask based on u_em has focus and the user selects Edit>Select All.

This event is also called by the m_edit popup menu.

pfc_Undo

Description

Cancels the last change to the EditMask, restoring the text to the content before the last change.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when an EditMask based on u_em has focus and the user selects Edit>Undo.

RButtonUp

Description

Displays the m_edit popup menu.

Usage

This event executes when the user releases the right mouse button over a control based on u_em.

Functions

U_em includes precoded object functions:

of_GetParentWindow	of_SetDropDownCalculator
of_MessageBox	of_SetDropDownCalendar

of_GetParentWindow

Description: Retrieves a reference to the parent window.

Access: Public

Syntax: *instancename*.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_em
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value: Integer. Returns 1 if the function succeeds and -1 if there is no parent window. If no parent window is found, *window* returns NULL.

Usage: The u_em GetFocus event calls this function.

Examples: This example is from the u_em GetFocus event:

```
Window   lw_parent
IF gnv_app.of_GetMicrohelp() THEN
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
      lw_parent.Dynamic Event &
         pfc_ControlGotFocus (this)
   END IF
END IF
```

of_MessageBox

Description: Displays a MessageBox.

Access: Protected

Syntax: *instancename*.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of u_em
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in EditMasks.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('em_error','Save', &
   as_error, StopSign!, Ok!, 1)
```

of_SetDropDownCalculator

Description

Enables or disables u_calculator, which provides a dropdown calculator.

Access

Public

Syntax

instancename.**of_SetDropDownCalculator** (*boolean*)

Argument	Description
instancename	Instance name of u_em
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the dropdown calculator service

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function to enable or disable the dropdown calculator. The u_em instance variable for this service is iuo_calculator.

Examples

This example calls the of_SetDropDownCalculator function to enable an instance of the dropdown calculator:

```
em_1.of_SetDropDownCalculator(TRUE)
```

of_SetDropDownCalendar

Description

Enables or disables u_calendar, which provides a dropdown calendar for registered columns.

Access

Public

Syntax

instancename.**of_SetDropDownCalendar** (*boolean*)

Argument	Description
instancename	Instance name of u_em
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the dropdown calendar service

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function to enable or disable the dropdown calendar.

Examples

This example calls the of_SetDropDownCalendar function to enable an instance of the dropdown calendar:

```
em_1.of_SetDropDownCalendar(TRUE)
```

u_gb

Description

GroupBox visual user object ancestor.

Ancestry

pfc_u_gb
u_gb
No instance variables, events or functions

Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use this visual object in windows instead of the standard PowerBuilder GroupBox.

u_gr

Description

Graph visual user object.

Ancestry

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

PFC visual user objects are designed to be used with windows that are descendants of w_master.

Usage

Use this visual user object in windows instead of the standard PowerBuilder Graph control. U_gr scripts provide integration with PFC menus.

See also

w_master

Events

U_gr includes a precoded event script:

GetFocus

GetFocus

Description

Updates the parent window so it can set MicroHelp.

Usage

This event calls the pfc_ControlGotFocus event in the parent.

Functions

U_gr includes precoded object functions:

of_GetParentWindow
of_MessageBox

of_GetParentWindow

Description

Retrieves a reference to the parent window.

Access

Public

Syntax

instancename.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_gr
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if there is no parent window. If no parent window is found, *window* returns NULL.

Usage

The u_gr GetFocus event calls this function.

Examples

This example is from the u_gr GetFocus event:

```
Window  lw_parent

IF gnv_app.of_GetMicrohelp() THEN
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
      lw_parent.Dynamic Event &
         pfc_ControlGotFocus (this)
   END IF
END IF
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of u_gr

Argument	Description
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in Graph controls.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('gr_error','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

u_hsb

Description — HorizontalScrollBar visual user object ancestor.

Ancestry

pfc_u_hsb
u_hsb
Events
GetFocus

Functions
Public of_GetParentWindow
Protected of_MessageBox

Library — PFCMAIN.PBL
PFEMAIN.PBL

Object relationships — PFC visual user objects are designed to be used with windows that are descendants of w_master.

Usage — Use this visual object in windows instead of the standard PowerBuilder HorizonalScrollBar. U_hsb event scripts provide integration with PFC menus.

See also — w_master
u_vsb

Events

U_hsb includes one precoded event:

GetFocus

GetFocus

Description — Updates the parent window so it can set MicroHelp.

Usage — This event calls the pfc_ControlGotFocus event in the parent.

Functions

U_hsb includes precoded object functions:

of_GetParentWindow
of_MessageBox

of_GetParentWindow

Description

Retrieves a reference to the parent window.

Access

Public

Syntax

instancename.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_hsb
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if there is no parent window. If no parent window is found, *window* returns NULL.

Usage

The u_hsb GetFocus event calls this function.

Examples

This example is from the u_hsb GetFocus event:

```
Window  lw_parent

//Check for MicroHelp requirements.
IF gnv_app.of_GetMicrohelp() THEN
   //Notify the parent.
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
      lw_parent.Dynamic Event &
         pfc_ControlGotFocus (this)
   END IF
END IF
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of u_hsb
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in HorizontalScrollBars.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('hsb_error','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

u_lb

Description

ListBox visual user object ancestor.

Ancestry

pfc_u_lb
u_lb
Events
GetFocus pfc_SelectAll
pfc_InvertSelection

Functions
Public of_GetParentWindow Protected of_MessageBox
Public of_GetSelected

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

PFC visual user objects are designed to be used with windows that are descendants of w_master.

Usage

Use this visual user object in windows instead of the standard PowerBuilder ListBox control. U_lb event scripts provide integration with PFC menus.

See also

w_master
u_ddlb
u_ddplb
u_plb

Events

U_lb includes precoded events:

GetFocus	pfc_SelectAll
pfc_InvertSelection	

GetFocus

Description

Updates the parent window so it can set MicroHelp.

Usage

This event calls the pfc_ControlGotFocus event in the parent.

pfc_InvertSelection

Description — Inverts the rows selected in the ListBox. Unselected rows become selected; selected rows are unselected.

Return value — Integer. Returns the number of selected rows if the event succeeds and 0 if this is an invalid operation.

Usage — To use this event, add a menu item that uses the message router to call pfc_InvertSelection.

pfc_SelectAll

Description — Selects all rows in the ListBox.

Return value — Integer. Returns the number of selected rows if the event succeeds and 0 if this is an invalid operation.

Usage — The message router calls this event when the user selects Edit>Select All from the menu bar.

Functions

U_lb includes precoded object functions:

of_GetParentWindow	of_MessageBox
of_GetSelected	

of_GetParentWindow

Description — Retrieves a reference to the parent window.

Access — Public

Syntax — *instancename*.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_lb
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if there is no parent window. If no parent window is found, *window* returns NULL.

Usage

The u_lb GetFocus event calls this function.

Examples

This example is from the u_lb GetFocus event:

```
Window  lw_parent
IF gnv_app.of_GetMicrohelp() THEN
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
      lw_parent.Dynamic Event &
         pfc_ControlGotFocus (this)
   END IF
END IF
```

of_GetSelected

Description

Populates the passed structure object with the selected entries in the ListBox.

Access

Public

Syntax

instancename.**of_GetSelected** (*itemattrib*)

Argument	Description
instancename	Instance name of u_lb
itemattrib	N_cst_itemattrib array into which the function places text and an item index (passed by reference)

Return value

Integer. Returns the number of elements in the *itemattrib* array if the function succeeds and 0 if this is not a valid operation.

Examples

This example calls the of_GetSelected function:

```
n_cst_itemattrib   lnv_items[ ]
Integer  li_selected

li_selected = lb_list.of_GetSelected(lnv_items)
IF li_selected = 0 THEN
   MessageBox("LB", "Invalid LB operation")
   Return
END IF
// Logic to read lnv_items follows
...
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of u_lb
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in ListBoxes.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('lb_error','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

u_lvs

Description

Service-based ListView visual user object ancestor. This object uses DataStores to display data in a ListView.

U_lvs is a self-updating object.

U_lvs replaces u_lv
U_lv is the ListView control supported in earlier PFC versions. Although PFC still provides u_lv, it's best to use u_lvs.

Ancestry

pfc_u_lvs
u_lvs

Events

ColumnClick	pfc_NewItem	pfc_Retrieve
Destructor	pfc_Populate	pfc_SelectAll
EndLabelEdit	pfc_PostUpdate	pfc_SetItemAttributes
GetFocus	pfc_PreDeleteItem	pfc_Undo
KeyDown	pfc_PreInsertItem	pfc_Update
pfc_AcceptText	pfc_PreRefreshItem	pfc_UpdatePrep
pfc_AddAll	pfc_PreRMBMenu	pfc_UpdatesPending
pfc_AddItem	pfc_Properties	pfc_Validation
pfc_DeleteItem	pfc_Refresh	RbuttonUp
pfc_InsertItem	pfc_RefreshItem	RightClicked
pfc_InvertSelection	pfc_RenameItem	Sort

Functions

Public	of_AcceptText	Public	of_Reset
Public	of_AddItem	Public	of_SetAlwaysValidate
Public	of_GetInfo	Public	of_SetBase
Protected	of_GetObjects	Public	of_SetDataSource
Public	of_GetParentWindow	Public	of_SetLogicalUnitOfWork
Public	of_GetRow	Public	of_SetSort
Public	of_GetUpdateRequestor	Public	of_SetUpdateable
Public	of_InsertItem	Public	of_SetUpdateRequestor
Public	of_IsAlwaysValidate	Public	of_Update
Public	of_IsUpdateable	Public	of_UpdatePrep
Protected	of_MessageBox	Public	of_UpdatesPending
Public	of_PostUpdate	Public	of_Validation

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

PFC visual user objects are designed to be used with windows that are descendants of w_master. U_lvs also uses:

m_lvs
n_cst_infoattrib
n_cst_luw
n_cst_lvsrv
n_cst_lvsrv_datasource

n_cst_lvsrv_sort
n_ds

Usage

Use this visual user object in windows instead of the standard PowerBuilder ListView control. U_lvs functions make it easy for you to populate a ListView from rows in a DataWindow. U_lvs event scripts also provide integration with PFC menus.

U_lvs includes right mouse button support. If the user clicks the right mouse button over a u_lvs-based ListView control, the m_lvs menu displays. M_lvs allows you to control ListView icon display.

To use u_lvs:

1 Place a u_lvs user object in your window.

2 In the window pfc_PreOpen event (or some other appropriate place), enable ListView services as needed by your application:

```
lv_1.of_SetDataSource(TRUE)
lv_1.of_SetSort(TRUE)
```

3 Specify sorting information:

```
String  ls_exclude[ ]

lv_1.inv_sort.of_SetColumnHeader(TRUE)
ls_exclude[1] = "emp_phone_number"
lv_1.inv_sort.of_SetExclude(ls_exclude)
```

4 Specify data source information:

```
Integer li_count

lv_1.inv_datasource.of_Register("emp_lname",  &
   "d_emplist", SQLCA)
li_count =  &
   lv_1.inv_datasource.of_RegisterReportColumn()
lv_1.inv_datasource.of_SetPictureColumn("1")
```

5 Populate the ListView with information from the registered data source:

```
lv_1.Event pfc_Populate()
```

6 Extend the pfc_Retrieve event to retrieve rows:

```
Any  la_args[20]
n_ds  lds_data

la_args[1] = "windows"     // Retrieval argument
```

```
this.of_Retrieve(la_args, lds_data)
```

7 Extend basic u_lvs functionality by adding PowerScript code to events and user events.

FOR INFO For more information on the ListView control, see the *PowerBuilder User's Guide*.

See also

m_lvs
n_ds
u_tvs

Instance variables

U_lvs includes instance variables:

Instance variable	Description	Data type	Access	Usage
ib_alwaysvalidate	Controls whether the save process includes all objects in the validation process	Boolean	Protected	Set with of_SetAlwaysValidate (default is FALSE)
ib_isupdateable	Indicates whether the ListView can be updated	Boolean	Protected	Set with of_SetUpdateable (default is FALSE)
ib_rmbmenu	Controls right mouse button support	Boolean	Protected	Use to enable or disable right mouse button support (default is TRUE) To disable right mouse button support, set this to FALSE in the ListView's Constructor event
il_lasthandle	Tracks the last ListView item added	Long	Protected	PFC uses this instance variable to track the ListView item
il_rightclicked	Contains the handle of the right-clicked ListView item	Long	Protected	PFC uses this instance variable to track the right-clicked item
inv_base	Reference variable for basic ListView services	n_cst_lvsrv	Public	Use in dot notation to access n_cst_lvsrv functions and attributes

Instance variable	Description	Data type	Access	Usage
inv_datasource	Reference variable for the ListView data source service	n_cst_lvsrv_ datasource	Public	Use in dot notation to access n_cst_lvsrv_datasource functions and attributes
inv_luw	Reference variable for logical unit of work service	n_cst_luw	Protected	Implements the save process
inv_sort	Reference variable for ListView sort service	n_cst_lvsrv_ sort	Public	Use in dot notation to access n_cst_lvsrv_sort functions and attributes
ipo_pendingupdates[ ]	Objects that could be updated	PowerObject	Protected	Internal
ipo_updaterequestor	Owner of the save process	PowerObject	Protected	Internal

Events

U_lvs includes precoded event scripts:

ColumnClick	pfc_PreRmbMenu
Destructor	pfc_Properties
EndLabelEdit	pfc_Refresh
GetFocus	pfc_RefreshItem
KeyDown	pfc_RenameItem
pfc_AcceptText	pfc_Retrieve
pfc_AddAll	pfc_SelectAll
pfc_AddItem	pfc_SetItemAttributes
pfc_DeleteItem	pfc_Undo
pfc_InsertItem	pfc_Update
pfc_InvertSelection	pfc_UpdatePrep
pfc_NewItem	pfc_UpdatesPending
pfc_Populate	pfc_Validation
pfc_PostUpdate	RButtonUp
pfc_PreDeleteItem	RightClicked
pfc_PreInsertItem	Sort
pfc_PreRefreshItem	

ColumnClick

Description — Sorts the ListView contents, using the column whose heading was clicked. If the column is already sorted, this event reverses the sort order.

Usage — To use this event, n_cst_lvsrv_sort must be enabled. This event executes when the user clicks a ListView control's column heading.

Destructor

Description — Destroys all existing ListView service objects.

Usage — This event executes when the ListView is destroyed or the window closes.

EndLabelEdit

Description — Updates the DataStore with user edits to the item label.

This function updates the DataStore only. You must update the database explicitly, using the of_Update or the n_cst_luw of_Save function.

Return value — Integer. Returns 0 if the label update is allowed and 1 if PowerBuilder should prevent the label update.

Usage — This event executes when the user finishes editing an item label.

PFC allows label update only when the associated label column is updatable. If the label column is a computed column, you must override this event or the n_cst_lvsrv_datasource pfc_EndLabelEdit event to update the computed column and return a 0.

GetFocus

Description — Updates the parent window so it can set MicroHelp.

Usage — This event calls the pfc_ControlGotFocus event in the parent.

KeyDown

Description — Checks to see if the DELETE key was pressed.

Usage — This event executes when the user presses a key and is not editing a label.

pfc_AcceptText

Description

Accepts text for the ListView control data source, optionally setting focus if an error occurs.

Syntax

instancename.EVENT **pfc_AcceptText** (*controls*, *focusonerror*)

Argument	Description
instancename	Instance name of u_lvs
controls	PowerObject array containing the objects on which to accept text. This argument is accessed through the *apo_control* argument
focusonerror	Boolean indicating whether focus should be set if an error occurs. This argument is accessed through the *ab_focusonerror* argument

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

Extend this event to perform additional accept text processing.

pfc_AddAll

Description

Adds all rows in the passed DataStore to the ListView.

Syntax

instancename.EVENT **pfc_AddAll** (*data*)

Argument	Description
instancename	Instance name of u_lvs
data	N_ds-based DataStore containing rows to be added. This argument is accessed through the *ads_source* argument

Return value

Integer. Returns the number of items added if the event succeeds and -1 if an error occurs.

Examples

This example is from the pfc_Refresh event:

```
...
inv_datasource.of_getdatasource(lds_source)
   return this.event pfc_AddAll(lds_source)
...
```

pfc_AddItem

Description

Adds the specified row in the data source to the end of the ListView.

Syntax

instancename.EVENT **pfc_AddItem** (*data*, *row*)

Argument	Description
instancename	Instance name of u_lvs
data	N_ds-based DataStore containing rows to be added. This argument is accessed through the *ads_source* argument
row	Long specifying the row containing the data to be added to the ListView. This argument is accessed through the *al_row* argument

Return value

Long. Returns the ListView index of the new row if the event succeeds and -1 if an error occurs.

Examples

This example is from the of_AddItem function:

```
...
li_newindex = this.event pfc_additem &
   (lds_datastore, ll_row)
...
```

pfc_DeleteItem

Description

Deletes the selected rows from the ListView and the DataStore.

Return value

Integer. Returns the number of rows deleted if the event succeeds and -1 if an error occurs.

Examples

This example calls the pfc_DeleteItem event:

```
...
li_return = lv_1.Event pfc_DeleteItem()
...
```

pfc_InsertItem

Description

Inserts a new item into the ListView at the specied position.

Syntax

instancename.EVENT **pfc_InsertItem** (*datastore*, *row*, *position*, *item*)

Argument	Description
instancename	Instance name of u_lvs
datastore	N_ds containing the data for the new item. This can be either the DataStore maintained by n_cst_lvsrv_datasource or another DataStore. If this is another DataStore, the associated DataWindow objects must be the same and the function also adds the row to the DataStore maintained by n_cst_lvsrv_datasource. This argument is accessed through the *ads_source* argument
row	Long specifying the row containing the item. This argument is accessed through the *al_row* argument
position	String specifying the position or location in which to insert the new item. Valid values are: ♦ First ♦ Last ♦ Before ♦ After This argument is accessed through the *as_position* argument
item	Integer containing an index relative to the current position. The function uses this argument if you specify Before or After for *position*. This argument is accessed through the *ai_index* argument

Return value

Integer. Returns the index of the new item if the function succeeds and -1 if an error occurs.

Usage

This event calls the pfc_SetItemAttributes and pfc_PreInsertItem events before inserting the item.

Examples

This example is from the pfc_AddAll event:

```
...
For ll_row = 1 to ll_rowcount
   If this.event pfc_insertitem(ads_source,  &
      ll_row, "last", 0) < 1 then
      Return -1
   End If
...
```

pfc_InvertSelection

Description — Inverts the rows selected in the ListView. Unselected rows become selected; selected rows are unselected.

Return value — Integer. Returns the number of selected rows if the event succeeds and -1 if this is an invalid operation.

Usage — To use this event, add a menu item that uses the message router to call pfc_InvertSelection.

pfc_NewItem

Description — Empty user event that you extend to add information to both the data source and the ListView.

Return value — Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage — You typically use this event to open a dialog box prompting the user for complete information, adding it to the data source and the ListView (via the pfc_InsertItem event) when it closes.

pfc_Populate

Description — Retrieves the data source and uses it to populate the ListView.

Return value — Integer. Returns the number of items added to the ListView if the event succeeds and -1 of an error occurs.

Usage — You must add code to the pfc_Retrieve event to retrieve the data.

Examples — This example calls the pfc_Populate event:

```
String  ls_exclude[ ]
Integer li_count

// ListView Constructor event
this.of_SetDataSource(TRUE)
this.of_SetSort(TRUE)
this.inv_sort.of_SetColumnHeader(TRUE)
ls_exclude[1] = "emp_phone_number"
this.inv_sort.of_SetExclude(ls_exclude)
this.inv_datasource.of_Register("emp_lname",  &
   "d_emplist", SQLCA)
```

```
li_count =  &
   this.inv_datasource.of_RegisterReportColumn()
this.inv_datasource.of_SetPictureColumn("1")
this.Event pfc_Populate()
```

pfc_PostUpdate

Description

Calls the n_cst_luw of_PostUpdate function to perform post-update processing on the specified controls.

Syntax

instancename.EVENT **pfc_PostUpdate** (*controls*)

Argument	Description
instancename	Instance name of u_lvs
controls	PowerObject array containing the objects on which to perform postupdate processing. This argument is accessed through the *apo_control* argument

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

Extend this event to perform additional post-update processing.

pfc_PreDeleteItem

Description

Empty user event to which you add logic to perform predelete processing.

Syntax

instancename.EVENT **pfc_PreDeleteItem** (*index*)

Argument	Description
instancename	Instance name of u_lvs
index	Integer specifying the index of the ListView item to be deleted. This argument is accessed through the *ai_index* argument

Usage

The pfc_DeleteItem event calls this event.

pfc_PreInsertItem

Description

Empty user event to which you add logic to perform pre-insert processing.

Syntax

instancename.Event **pfc_PreInsertItem** (*data, row, lvitem*)

Argument	Description
instancename	Instance name of u_lvs
data	N_ds containing the data for the new item. This argument is accessed through the *ads_obj* argument
row	Long specifying the row containing the new item. This argument is accessed through the *al_row* argument
lvitem	ListViewItem to be inserted. This argument is accessed through the *alvi_item* argument (passed by reference)

Usage

The pfc_InsertItem event calls this event.

pfc_PreRefreshItem

Description

Empty user event to which you add logic that changes ListView item properties before a ListView item is refreshed.

Syntax

instancename.EVENT **pfc_PreRefreshItem** (*index, data, row, lvitem*)

Argument	Description
instancename	Instance name of u_lvs
index	Integer specifying the index of the refreshed item. This argument is accessed through the *ai_index* argument
data	N_ds containing the data for the refreshed item. This argument is accessed through the *ads_obj* argument
row	Long specifying the row containing the refreshed item. This argument is accessed through the *al_row* argument
lvitem	ListViewItem to be refreshed. This argument is accessed through the *alvi_item* argument (passed by reference)

Usage

The pfc_RefreshItem event calls this event.

pfc_PreRmbMenu

Description

Empty user event allowing you to modify m_lvs contents before display.

Syntax

instancename.Event **pfc_PreRmbMenu** (*editmenu*)

Argument	Description
instancename	Instance name of u_lvs
editmenu	M_lvs variable containing the popup menu to be displayed (passed by reference)

Usage

Optionally add logic to this event to selectively enable and disable m_lvs menu items.

pfc_Properties

Description

Empty user event to which you might add logic to display a Properties dialog.

Return value

Integer. Return 1 if the event succeeds and -1 if an error occurs.

pfc_Refresh

Description

Refresh the ListView items with data from the data source.

Return value

Integer. Returns the number of items refreshed if the event succeeds and -1 if an error occurs.

Examples

This example is from the pfc_Populate event:

```
...
If this.event pfc_refresh() <> 1 Then Return -3
...
```

pfc_RefreshItem

Description

Refreshes the specified ListView item, resetting all properties to the defaults specified in the data source.

Syntax

instancename.EVENT **pfc_RefreshItem** (*index*)

Argument	Description
instancename	Instance name of u_lvs
index	Integer specifying the ListView index of the item to refresh

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Examples

This example calls the pfc_RefreshItem event:

```
...
lv_1.Event pfc_RefreshItem(1)
...
```

pfc_RenameItem

Description

Renames the selected item.

Usage

To use this event, the Edit Labels property must be enabled for the ListView.

pfc_Retrieve

Description

Retrieves rows into the ListView's data source.

Syntax

instancename.EVENT **pfc_Retrieve** (*datasource*)

Argument	Description
instancename	Instance name of u_lvs
datasource	N_ds into which rows are retrieved. This argument is accessed through the *ads_data* argument

Return value

Long. Return the number of items retrieved if the event succeeds and -1 if an error occurs.

Usage

Add code to this eventthat uses the of_Retrieve function to retrieve rows for the data source.

Examples

This example shows code you might add to the pfc_Retrieve event:

```
Any  la_args[20]
n_ds  lds_data

la_args[1] = "windows"     // Retrieval argument
this.of_Retrieve(la_args, lds_data)
```

pfc_SelectAll

Description

Selects all rows in the ListView.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when the user selects Edit>Select All from the menu bar.

pfc_SetItemAttributes

Description

Sets default properties for the ListView item before insertion.

Syntax

instancename.EVENT **pfc_SetItemAttributes** (*data*, *row*, *lvitem*)

Argument	Description
instancename	Instance name of u_lvs
data	N_ds containing the data for the item. This argument is accessed through the *ads_obj* argument
row	Long specifying the row containing the item. This argument is accessed through the *al_row* argument
lvitem	ListViewItem to be inserted. This argument is accessed through the *alvi_item* argument (passed by reference)

Usage

Optionally extend this event to change ListView item properties before insertion.

pfc_Undo

Description

Cancels the last change to the ListView, restoring the text to the content before the last change.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a ListView based on u_lvs has focus and the user selects Edit>Undo.

pfc_Update

Description

Calls the n_cst_luw of_Update function to update the specified controls.

Syntax

instancename.EVENT **pfc_Update** (*controls*, *accepttext*, *resetflags*)

Argument	Description
instancename	Instance name of u_lvs

Argument	Description
controls	PowerObject array containing the objects to update. This argument is accessed through the *apo_control* argument
accepttext	Boolean specifying whether n_cst_luw should automatically perform an AcceptText before performing the Update (TRUE) or not (FALSE)
resetflags	Boolean specifying whether n_cst_luw should automatically reset DataWindow update flags (TRUE) or not (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

The of_Update function calls this event.

pfc_UpdatePrep

Description

Empty user event to which you can add code that prepares for update.

Syntax

instancename.EVENT **pfc_UpdatePrep** (*controls*)

Argument	Description
instancename	Instance name of u_lvs
controls	PowerObject array containing the objects to update. This argument is accessed through the *apo_control* argument

Return value

Long. Return 1 if the update preparation succeeds and -1 to halt the update process.

Usage

The of_UpdatePrep function calls this function.

pfc_UpdatesPending

Description

Determines if there are pending updates for the ListView.

Syntax

instancename.EVENT **pfc_UpdatesPending** (*controls, pending*)

Argument	Description
instancename	Instance name of u_lvs
controls	PowerObject array containing the objects to update. This argument is accessed through the *apo_control* argument

Argument	Description
pending	PowerObject array into which the event places objects with pending updates. This argument is accessed through the *apo_pending* argument (passed by reference)

Return value

Integer. Returns 1 if there are pending updates, 0 if there are no pending updates, and -1 if an error occurs.

Usage

This event is called by the of_GetUpdatesPending function.

pfc_Validation

Description

Checks for required fields.

Syntax

instancename.EVENT **pfc_Validation** (*controls*)

Argument	Description
instancename	Instance name of u_lvs
controls	PowerObject array containing the objects to validate. This argument is accessed through the *apo_control* argument

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is called by the of_Validation function.

RButtonUp

Description

Displays the m_lvs popup menu.

Usage

This event executes when the user releases the right mouse button.

RightClicked

Description

Tracks the clicked item.

Usage

This event executes when the user presses the right mouse button.

Sort

Description

Performs the sort comparison when the sort type is UserDefined!.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

Extend the n_cst_lvsrv_sort pfc_Sort event to create site-specific sort logic.

Functions

U_lvs includes precoded object functions:

of_AcceptText	of_Reset
of_AddItem	of_SetAlwaysValidate
of_GetInfo	of_SetBase
of_GetObjects	of_SetDataSource
of_GetParentWindow	of_SetLogicalUnitOfWork
of_GetRow	of_SetSort
of_GetUpdateRequestor	of_SetUpdateable
of_InsertItem	of_SetUpdateRequestor
of_IsAlwaysValidate	of_Update
of_IsUpdateable	of_UpdatePrep
of_MessageBox	of_UpdatesPending
of_PostUpdate	of_Validation

of_AcceptText

Description

Performs an AcceptText function for the data source.

Access

Public

Syntax

instancename.**of_AcceptText** (*focusonerror*)

Argument	Description
instancename	Instance name of u_lvs
focusonerror	Boolean indicating whether PFC sets focus to the first item in error when an error occurs

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize accept text processing, extend the pfc_AcceptText event.

Examples

This example is from the n_cst_luw of_AcceptText function:

```
...
If lb_defined Then
   li_rc =  &
      lpo_tocheck.Function Dynamic of_AcceptText &
         (ab_focusonerror)
   If li_rc < 0 Then Return -1

...
```

of_AddItem

Description

Adds a new item to the end of the ListView control.

Access

Public

Syntax

instancename.**of_AddItem** (*rowinfo*)

Argument	Description
instancename	Instance name of u_lvs
rowinfo	Any array containing data for the row

Return value

Long. Returns the index of the added item if the functions succeeds and -1 if an error occurs.

Usage

If the information in *rowinfo* is not already in the data source, this function adds the new row to the data source. If you are using the ListView data source service, the data types of the elements in *rowinfo* must match those in the DataWindow object specified in the n_cst_lvsrv_datasource of_Register function.

Examples

This example calls the of_AddItem function:

```
Any  la_row[ ]

la_row[1] = 500
la_row[2] = "Nelsen"
la_row[3] = "Rodney"
lv_1.of_AddItem(la_row)
```

of_GetInfo

Description

Retrieves object information.

Access

Public

Syntax

instancename.**of_GetInfo** (*infoobject*)

Argument	Description
instancename	Instance name of u_lvs
infoobject	N_cst_infoattrib instance into which the function places information (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_GetInfo function:

```
n_cst_infoattrib  lnv_info

dw_1.inv_base.of_GetInfo(lnv_info)
MessageBox("Info",  &
  "Description: " + lnv_info.is_description  &
    + ". Name: " + lnv_info.is_name)
```

of_GetObjects

Description

Retrieves the objects to be updated.

Access

Protected

Syntax

instancename.**of_GetObjects** (*objects*)

Argument	Description
instancename	Instance name of u_lvs
objects	PowerObject array into which the function places objects to be updated (passed by reference)

Return value

Integer. Returns the number of elements in the *objects* array if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the of_AcceptText function:

```
...
Else
   this.of_Getobjects(lpo_updatearray)
End If
```

. . .

of_GetParentWindow

Description

Retrieves a reference to the parent window.

Access

Public

Syntax

instancename.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_lvs
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if there is no parent window. If no parent window is found, *window* returns NULL.

Usage

The u_lvs GetFocus event calls this function.

Examples

This example is from the u_lvs GetFocus event:

```
Window   lw_parent

IF gnv_app.of_GetMicrohelp() THEN
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
      lw_parent.Dynamic Event &
         pfc_ControlGotFocus (this)
   END IF
END IF
```

of_GetRow

Description

Retrieves the DataStore and DataStore row for the specified ListView index.

Access

Public

Syntax

instancename.**of_GetRow** (*index, data, row*)

Argument	Description
instancename	Instance name of u_lvs
index	Integer specifying the index for the row to be retrieved

Argument	Description
data	N_ds into which the function places a reference to the DataStore (passed by reference)
row	Long into which the function places the row number that corresponds to *index* (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_GetRow function:

```
n_ds  lds_data
Long  ll_row

lv_1.of_GetRow(1, lds_data, ll_row)
MessageBox("ListView",  &
   "DataObject is: " + lds_data.DataObject  &
   + " Row is: " + String(ll_row))
```

of_GetUpdateRequestor

Description

Retrieves a reference to the object requesting an update.

Access

Public

Syntax

instancename.**of_GetUpdateRequestor** (*requestor*)

Argument	Description
instancename	Instance name of u_lvs
requestor	PowerObject into which the function places a reference to the object requesting the update (passed by reference)

Return value

Integer. Returns 1 if the function succeeds, 0 if there is no update requestor, and -1 if an error occurs.

Usage

Call this function if you are extending the pfc_Save process and need to access the update requestor.

of_InsertItem

Inserts an item into the ListView. There are two syntaxes:

To	Use
Insert from a DataStore	Syntax 1
Insert from an array	Syntax 2

Syntax 1 **Insert an item from a DataStore**

Description

Adds a new item to the ListView using a row from a DataStore.

Access

Public

Syntax

instancename.**of_InsertItem** (*datastore*, *row* {, *position*, *index* })

Argument	Description
instancename	Instance name of u_lvs
datastore	N_ds containing the data for the new item. This can be either the DataStore maintained by n_cst_lvsrv_datasource or another DataStore. If this is another DataStore, the associated DataWindow objects must be the same and the function also adds the row to the DataStore maintained by n_cst_lvsrv_datasource (passed by reference)
row	Long specifying the row containing data for the new item
position (optional)	String specifying the position under the current parent in which to insert the new item. Valid values are: ♦ First ♦ Last (default) ♦ Before ♦ After
index (optional)	Integer containing an index relative to the current position. The function uses this argument if you specify Before or After for *position*

Return value

Integer. Returns the index of the new item if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_InsertItem function:

```
Long  ll_row

ll_row = ids_data.GetRow()
lv_1.of_InsertItem(ids_data, ll_row)
```

Syntax 2 **Insert an item from an array**

Description Adds a new item to the ListView using a value from an array.

Access Public

Syntax *instancename*.**of_InsertItem** (*colvalues* {, *position*, *index* })

Argument	Description
instancename	Instance name of u_lvs
colvalues	Any array containing data for the ListView
position (optional)	String specifying the position or location in which to insert the new item. Valid values are: ♦ First ♦ Last (default) ♦ Before ♦ After
index (optional)	Integer containing an index relative to the current position. The function uses this argument if you specify Before or After for *position*

Return value Integer. Returns the index of the new item if the function succeeds and -1 if an error occurs.

Examples This example calls the of_InsertItem function:

```
Any  la_row[ ]

la_row[1] = 500
la_row[2] = "Nelsen"
la_row[3] = "Rodney"
lv_1.of_InsertItem(la_row, "First", 0)
```

of_IsAlwaysValidate

Description Reports whether the default save process always performs validation.

Access Public

Syntax *instancename*.**of_IsAlwaysValidate** ( )

Argument	Description
instancename	Instance name of u_lvs

Return value

Boolean. Returns TRUE if the default save process always performs validation and FALSE if it does not.

Examples

This example calls the of_IsAlwaysValidate function:

```
IF lv_1.of_IsAlwaysValidate() = TRUE THEN
   MessageBox("LV", "Always validate")
ELSE
   MessageBox("LV", "Sometimes validate")
END IF
```

of_IsUpdateable

Description

Reports whether the ListView's data source is updatable and should be included in a window's default save processing.

Access

Public

Syntax

instancename.**of_IsUpdateable** ( )

Argument	Description
instancename	Instance name of u_lvs

Return value

Boolean. Returns TRUE if the ListView's data source is updatable and FALSE if it is not.

Usage

The pfc_UpdatesPending event calls this function.

Examples

This example is from the pfc_UpdatesPending event:

```
...
If Not of_IsUpdateable() Then Return NO_UPDATESPENDING
...
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of u_lvs
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in ListViews.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('lv_error','Save', &
   as_error, StopSign!, Ok!, 1)
```

of_PostUpdate

Description

Calls the pfc_PostUpdate event, which clears update flags and allows you to code additional post-update processing.

Access

Public

Syntax

instancename.**of_PostUpdate** ( )

Argument	Description
instancename	Instance name of u_lvs

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize post-update processing, extend the pfc_PostUpdate event.

Examples

This example is from the n_cst_luw of_PostUpdate function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_PostUpdate()
...
```

of_Reset

Description

Deletes all items from both the ListView and the data source.

Access

Public

Syntax

instancename.**of_Reset** ( )

Argument	Description
instancename	Instance name of u_lvs

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Examples

This example calls the of_Reset function:

```
lv_1.of_Reset()
```

of_SetAlwaysValidate

Description

Specifies whether the default save process always performs validation.

Access

Public

Syntax

instancename.**of_SetAlwaysValidate** (*boolean*)

Argument	Description
instancename	Instance name of u_lvs
boolean	Boolean specifying whether the default save process always perform validation (TRUE) or only performs validation if a control has pending updates

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetAlwaysValidate function:

```
lv_1.of_SetAlwaysValidate(TRUE)
```

of_SetBase

Description

Enables or disables n_cst_lvsrv, which provides basic ListView services.

Access

Public

Syntax

instancename.**of_SetBase** (*boolean*)

Argument	Description
instancename	Instance name of u_lvs
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) basic ListView services

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_lvsrv. This instance is named inv_base.

Because all ListView services are descendants of n_cst_lvsrv (and have n_cst_lvsrv functions available to them), use this object when you require basic ListView services only.

Examples

This example calls the of_SetBase function to enable basic ListView services:

```
lv_1.of_SetBase(TRUE)
```

of_SetDataSource

Description

Enables or disables n_cst_lvsrv_datasource, which provides data access for ListViews.

Access

Public

Syntax

instancename.**of_SetDataSource** (*boolean*)

Argument	Description
instancename	Instance name of u_lvs
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) n_cst_lvsrv_datasource

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_lvsrv_datasource. This instance is named inv_datasource.

Examples

This example calls the of_SetDataSource function:

```
lv_1.of_SetDataSource(TRUE)
```

of_SetLogicalUnitOfWork

Description

Enables or disables n_cst_luw, which provides the logical unit of work service.

Access

Public

Syntax

instancename.**of_SetLogicalUnitOfWork** (*boolean*)

Argument	Description
instancename	Instance name of u_lvs
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) n_cst_luw

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_luw. This instance is named inv_luw. If you do not enable n_cst_luw, u_lvs enables it automatically.

Examples

This example calls the of_SetLogicalUnitOfWork function:

```
lv_1.of_SetLogicalUnitOfWork(TRUE)
```

of_SetSort

Description

Enables or disables n_cst_lvsrv_sort, which provides the ListView sort service.

Access

Public

Syntax

instancename.**of_SetSort** (*boolean*)

Argument	Description
instancename	Instance name of u_lvs
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the ListView sort service

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_lvsrv_sort. This instance is named inv_sort.

Examples

This example calls the of_SetSort function:

```
lv_1.of_SetSort(TRUE)
```

of_SetUpdateable

Description

Specifies whether the ListView is updatable and should be included in the default save process.

Access

Public

Syntax

instancename.**of_SetUpdateable** (*boolean*)

Argument	Description
instancename	Instance name of u_lvs

Argument	Description
boolean	Boolean indicating whether the ListView is updatable. All updatable ListViews are included in the default save processing

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function to enable default save processing for ListViews (by default, they are not updatable).

Examples

This example calls the of_SetUpdateable function:

```
lv_1.of_SetUpdateable(TRUE)
```

of_SetUpdateRequestor

Description

Creates a reference to the object requesting an update within a logical unit of work.

Access

Public

Syntax

instancename.**of_SetUpdateRequestor** (*requestor*)

Argument	Description
instancename	Instance name of u_lvs
requestor	PowerObject containing the object requesting the update

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the of_Update function:

```
...
If IsValid(inv_luw) Then
   inv_luw.of_SetUpdateRequestor(apo_requestor)
End If
...
```

of_Update

Description

Saves all rows in the DataStore associated with the ListView.

Access

Public

Syntax

instancename.**of_Update** (*accept*, *resetflag* {, *requestor* })

Argument	Description
instancename	Instance name of u_lvs
accept	Boolean indicating whether the Update function performs an AcceptText before saving rows to the database
resetflag	Boolean indicating whether the Update function resets the update flags
requestor	PowerObject containing the requestor object

Return value

Integer. Returns 1 if the update succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize update processing, extend the pfc_Update event.

Examples

This example is from the n_cst_luw of_Update function:

```
...
If lb_defined Then
  li_rc = lpo_tocheck.Function Dynamic of_Update &
    (ab_accepttext, ab_resetflag, &
      lpo_updaterequestor)
  If li_rc < 0 Then Return -1
  Continue
End If
...
```

of_UpdatePrep

Description

Calls the pfc_UpdatePrep event, which allows you to code additional update preparation logic.

Access

Public

Syntax

instancename.**of_UpdatePrep** ( )

Argument	Description
instancename	Instance name of u_lvs

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the n_cst_luw of_UpdatePrep function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_UpdatePrep ()
  If li_rc < 0 Then Return -1
  Continue
End If
...
```

of_UpdatesPending

Description

Determines if there are updates pending in the ListView.

Access

Public

Syntax

instancename.**of_UpdatesPending** ( )

Argument	Description
instancename	Instance name of u_lvs

Return value

Integer. Returns values as follows:

- **1** Updates are pending
- **0** No updates pending
- **-1** An error occurred

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize pending updates processing, extend the pfc_UpdatesPending event.

Examples

This example calls the of_UpdatesPending function:

```
...
If lb_defined Then
   la_rc = lpo_tocheck.Dynamic of_UpdatesPending()
...
```

of_Validation

Description

Performs validation on the ListView data source.

Access

Public

Syntax

instancename.**of_Validation** ( )

Argument	Description
instancename	Instance name of u_lvs

Return value

Integer. Returns 1 if the function succeeds and -1 if a validation error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize validation processing, extend the pfc_Validation event.

Examples

This example calls the of_Validation function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_Validation()
...
```

u_mle

Description

MultiLineEdit visual user object ancestor.

Ancestry

pfc_u_mle
u_mle
Events
GetFocus pfc_PreRmbMenu
pfc_Clear pfc_SelectAll
pfc_Copy pfc_Undo
pfc_Cut RButtonUp
pfc_Paste

Object functions
Public of_GetParentWindow
Protected of_MessageBox

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

PFC visual user objects are designed to be used with windows that are descendants of w_master.

Usage

Use this visual object in windows instead of the standard PowerBuilder MultiLineEdit. U_mle event scripts provide integration with PFC menus. Additionally, u_mle supports:

- **Cut, copy, paste, clear, and select all** These are triggered automatically by the message router.
- **Right mouse button support** The RButtonUp event enables you to use the right mouse button to perform editing actions. To disable right mouse button support, set ib_rmbmenu to FALSE in the MultiLineEdit's Constructor event.
- **Autoselect** This means that text is selected when a user tabs to the edit mask. To disable autoselect, set ib_autoselect to FALSE in the MultiLineEdit's Constructor event.

See also

m_edit

Instance variables

U_mle includes instance variables:

Instance variable	Description	Data type	Access	Usage
ib_autoselect	Indicates whether PFC selects text automatically when the control receives focus	Boolean	Protected	Set this to TRUE to enable autoselect (default is FALSE)
ib_rmbmenu	Indicates whether the m_edit menu displays when the user presses the right mouse button	Boolean	Protected	Set this to FALSE to disable right mouse button support (default is TRUE)

Events

U_mle includes precoded events:

GetFocus	pfc_PreRmbMenu
pfc_Clear	pfc_SelectAll
pfc_Copy	pfc_Undo
pfc_Cut	RButtonUp
pfc_Paste	

GetFocus

Description

Updates the parent window so it can set MicroHelp.

Usage

This event calls the pfc_ControlGotFocus event in the parent.

pfc_Clear

Description

Deletes selected text.

Return value

Integer. Returns the number of characters deleted if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a MultiLineEdit based on u_mle has focus and the user selects Edit>Clear from the menu bar.

pfc_Copy

Description

Copies selected text to the clipboard.

Return value

Integer. Returns the number of characters copied if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a MultiLineEdit based on u_mle has focus and the user selects Edit>Copy from the menu bar. This event is also called by the m_edit popup menu.

pfc_Cut

Description

Deletes selected text and stored it on the clipboard.

Return value

Integer. Returns the number of characters removed if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a MultiLineEdit based on u_mle has focus and the user selects Edit>Cut from the menu bar. This event is also called by the m_edit popup menu.

pfc_Paste

Description

Inserts (pastes) the contents of the clipboard at the insertion point.

Return value

Integer. Returns the number of characters that were pasted if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a MultiLineEdit based on u_mle has focus and the user selects Edit>Paste from the menu bar. This event is also called by the m_edit popup menu.

pfc_PreRmbMenu

Description

User event allowing you to modify m_edit contents before display.

Syntax

instancename.EVENT **pfc_PreRmbMenu** (*editmenu*)

Argument	Description
instancename	Instance name of u_mle

Argument	Description
editmenu	M_edit variable containing the popup menu to be displayed (passed by reference)

Usage

Add logic to this event to selectively enable and disable m_edit menu items.

pfc_SelectAll

Description

Selects all text in the MultiLineEdit.

Return value

Integer. Returns the number of characters selected if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a MultiLineEdit based on u_mle has focus and the user selects Edit>Select All from the menu bar. This event is also called by the m_edit popup menu.

pfc_Undo

Description

Cancels the last change to the MultiLineEdit, restoring the text to the content before the last change.

Return value

Integer. Returns 1 if the event succeeds 0 and -1 if an error occurs.

Usage

The message router calls this event when a MultiLineEdit based on u_mle has focus and the user selects Edit>Undo.

RButtonUp

Description

Displays the m_edit popup menu.

Usage

This event executes when the user releases the right mouse button over a control based on u_mle.

Functions

U_mle includes precoded object functions:

of_GetParentWindow

of_MessageBox

of_GetParentWindow

Description

Retrieves a reference to the parent window.

Access

Public

Syntax

instancename.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_mle
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if there is no parent window.

Usage

The u_mle GetFocus event calls this function.

Examples

This example is from the u_mle GetFocus event:

```
Window  lw_parent

IF gnv_app.of_GetMicrohelp() THEN
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
      lw_parent.Dynamic Event pfc_ControlGotFocus &
         (this)
   END IF
END IF
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of u_mle
id	String specifying an ID for the message

Argument	Description
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in MultiLineEdits.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('mle_error','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

u_oc

Description

OLE visual user object ancestor.

Ancestry

pfc_u_oc
u_oc
Events

GetFocus	pfc_OpenObject
pfc_Clear	pfc_Paste
pfc_Copy	pfc_PasteSpecial
pfc_Cut	pfc_PreRMBMenu
pfc_EditObject	pfc_UpdateLinks
pfc_InsertObject	RButtonUp

Object functions

Public	of_GetParentWindow
Protected	of_MessageBox

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

PFC visual user objects are designed to be used with windows that are descendants of w_master.

Usage

Use this visual object in windows instead of the standard PowerBuilder OLE 2.0 window control. U_oc event scripts provide integration with PFC menus. Additionally, u_oc supports:

- **Cut, copy, paste, paste special, clear, and select all** These are triggered automatically by the message router.
- **Right mouse button support** The RButtonUp event enables you to use the right mouse button to perform editing actions. To disable right mouse button support, set ib_rmbmenu to FALSE in the OLE control's Constructor event.
- **Object actions** U_oc provides events to activate the object (inplace and offsite), update links, and insert an object into the OLE control.

See also

m_oc

Instance variables

U_oc contains one instance variable:

Instance variable	Description	Data type	Ancestry	Usage
ib_rmbmenu	Indicates whether the m_oc menu displays when the user presses the right mouse button	Boolean	Protected	Set this to FALSE to disable right mouse button support

Events

U_oc includes precoded events:

GetFocus	pfc_OpenObject
pfc_Clear	pfc_Paste
pfc_Copy	pfc_PasteSpecial
pfc_Cut	pfc_PreRmbMenu
pfc_EditObject	pfc_UpdateLinks
pfc_InsertObject	RButtonUp

GetFocus

Description

Updates the parent window so it can set MicroHelp.

Usage

This event calls the pfc_ControlGotFocus event in the parent.

pfc_Clear

Description

Deletes the OLE object.

Return value

Integer. Returns 0 if the event succeeds and -9 if an error occurs.

Usage

The message router calls this event when an OLE control based on u_oc has focus and the user selects Edit>Clear from a menu that descends from the PFC m_master menu.

pfc_Copy

Description

Copies the OLE object to the clipboard.

Return value

Integer. Returns one of the following values:

0 Success

-1 Container is empty
-2 Copy failed
-9 Other error

Usage

The message router calls this event when an OLE control based on u_oc has focus and the user selects Edit>Copy from a menu that descends from the PFC m_master menu. This event is also called by the m_oc popup menu.

pfc_Cut

Description

Deletes the selected item or text and stores it on the clipboard.

Return value

Integer. Returns values as follows:

-1 Container is empty
-2 Cut failed
-9 Other error

Usage

The message router calls this event when an OLE control based on u_oc has focus and the user selects Edit>Cut. This event is also called by the m_oc popup menu.

pfc_EditObject

Description

Activates the object in place.

Return value

Integer. Returns values as follows:

0 Success
-1 Control is empty
-2 Invalid verb for the object
-3 Verb not implemented by the object
-4 No verbs supported by the object
-5 Object can't execute the verb now
-9 Other error

Usage

Call this event to activate an object in-place.

The message router calls this event when an OLE control based on u_oc has focus and the user selects Edit>Object>Edit from a menu that descends from the PFC m_master menu. This event is also called by the m_oc popup menu.

pfc_InsertObject

Description

Displays the Insert Object dialog box, which allows you to insert an object into the OLE control.

Return value

Integer. Returns values as follows:

0 Success
1 The user canceled out of the dialog box
-9 Other error

Usage

Call this event to insert an object into the OLE control.

The message router triggers this event when an OLE control based on u_oc has focus and the user selects Edit>Insert Object from a menu that descends from the PFC m_master menu.

pfc_OpenObject

Description

Activates the object offsite.

Return value

Integer. Returns values as follows:

0 Success
-1 Control is empty
-2 Invalid verb for the object
-3 Verb not implemented by the object
-4 No verbs supported by the object
-5 Object can't execute the verb now
-9 Other error

Usage

Call this event to activate an object offsite.

The message router calls this event when an OLE control based on u_oc has focus and the user selects Edit>Object>Open from a menu that descends from the PFC m_master menu. This event is also called by the m_oc popup menu.

pfc_Paste

Description

Inserts (pastes) the contents of the clipboard at the insertion point.

Return value

Integer. Returns values as follows:

0 Success
-1 No data or clipboard contents cannot be embedded
-9 Other error

Usage

The message router calls this event when an OLE control based on u_oc has focus and the user selects Edit>Paste from a menu that descends from the PFC m_master menu. This event is also called by the m_oc popup menu.

pfc_PasteSpecial

Description

Displays a standard OLE 2.0 dialog box allowing the user to choose whether to embed or link the OLE object on the clipboard when pasting it in the specified control. Embedding is the equivalent of calling the Paste function and linking is the same as calling PasteLink.

Return value

Integer. Returns values as follows:

0 Success
1 User canceled out of the OLE 2.0 dialog box
-1 No data found
-9 Other error

Usage

The message router calls this event when an OLE control based on u_oc has focus and the user selects Edit>Paste Special from a menu that descends from the PFC m_master menu.

pfc_PreRmbMenu

Description

User event allowing you to modify m_oc contents prior to display.

Syntax

instancename.Event **pfc_PreRmbMenu** (*editmenu*)

Argument	Description
instancename	Instance name of u_oc
editmenu	M_oc variable containing the popup menu to be displayed (passed by reference)

Usage

Optionally add logic to this event to selectively enable and disable m_oc menu items.

pfc_UpdateLinks

Description

Attempts to find a file linked to an OLE container. If the linked file is not found, a dialog tells the user and lets them bring up a second dialog for find the file or changing the link.

Return value

Integer. Returns 0 if the event succeeds and -1 if an error occurs.

Usage

Call this event to update links for an OLE control.

The message router triggers this event when an OLE control based on u_oc has focus and the user selects Edit>Update Links from a menu that descends from the PFC m_master menu.

RButtonUp

Description

Displays the m_oc popup menu.

Usage

This event executes when the user releases the right mouse button over a control based on u_oc.

Functions

U_oc includes precoded object functions:

of_GetParentWindow
of_MessageBox

of_GetParentWindow

Description

Retrieves a reference to the parent window.

Access

Public

Syntax

instancename.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_oc
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if there is no parent window.

Usage

The u_oc GetFocus event calls this function.

Examples

This example is from the u_oc GetFocus event:

```
Window   lw_parent

IF gnv_app.of_GetMicrohelp() THEN
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
      lw_parent.DynamicEvent pfc_ControlGotFocus &
         (this)
   END IF
END IF
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id*, *title*, *message*, *icon*, *button*, *default*)

Argument	Description
instancename	Instance name of u_oc
id	String specifying an ID for the message
title	String specifying the title of the OLEControl
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!

Argument	Description
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in OLE controls.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

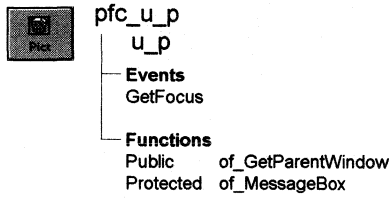
```
of_Messagebox('oc_error','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

u_p

Description

Picture visual user object ancestor.

Ancestry



Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

PFC visual user objects are designed to be used with windows that are descendants of w_master.

Usage

Use this visual object in windows instead of the standard PowerBuilder Picture control. U_p event scripts provide integration with PFC menus.

See also

u_pb

Events

U_p includes a precoded event script:

GetFocus

GetFocus

Description

Updates the parent window so it can set MicroHelp.

Usage

This event calls the pfc_ControlGotFocus event in the parent.

Functions

U_p includes precoded object functions:

of_GetParentWindow
of_MessageBox

of_GetParentWindow

Description

Returns a reference to the parent window.

Access

Public

Syntax

instancename.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_p
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if there is no parent window. If no parent window is found, *window* returns NULL.

Usage

The u_p GetFocus event calls this function.

Examples

This example is from the u_p GetFocus event:

```
Window   lw_parent

IF gnv_app.of_GetMicrohelp() THEN
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
      lw_parent.Dynamic Event &
         pfc_ControlGotFocus(this)
   END IF
END IF
```

of_MessageBox

Description

Displays a MessageBox

Access

Protected

Syntax

instancename.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of u_p

Argument	Description
id	String specifying an ID for the message
title	String specifying the title of the Picture control
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in Picture controls.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('p_error','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

u_pb

Description

PictureButton visual user object ancestor.

Ancestry

pfc_u_pb
u_pb
Events
GetFocus

Functions
Public of_GetParentWindow
Protected of_MessageBox

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

PFC visual user objects are designed to be used with windows that are descendants of w_master.

Usage

Use this visual object in windows instead of the standard PowerBuilder PictureButton control. U_pb event scripts provide integration with PFC menus.

See also

u_cb
u_p

Events

U_pb includes a precoded event script:

GetFocus

GetFocus

Description

Updates the parent window so it can set MicroHelp.

Usage

This event calls the pfc_ControlGotFocus event in the parent.

Functions

U_pb includes precoded object functions:

of_GetParentWindow
of_MessageBox

of_GetParentWindow

Description

Returns a reference to the parent window.

Access

Public

Syntax

instancename.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_pb
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if there is no parent window. If no parent window is found, *window* returns NULL.

Usage

The u_pb GetFocus event calls this function.

Examples

This example is from the u_pb GetFocus event:

```
Window   lw_parent

IF gnv_app.of_GetMicrohelp() THEN
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
      lw_parent.Dynamic Event &
         pfc_ControlGotFocus (this)
   END IF
END IF
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of u_pb

Argument	Description
id	String specifying an ID for the message
title	String specifying the title of the PictureButton
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in PictureButtons.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('pb_error','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

u_plb

Description

PictureListBox user object ancestor.

Ancestry

pfc_u_plb
u_plb
Events
GetFocus pfc_SelectAll
pfc_InvertSelection

Object functions
Public of_GetParentWindow Protected of_MessageBox
Protected of_GetSelected

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

PFC visual user objects are designed to be used with windows that are descendants of w_master.

Usage

Use this visual object in windows instead of the standard PowerBuilder PictureListBox. U_plb event scripts provide integration with PFC menus.

See also

u_lb

Events

U_plb includes precoded events:

GetFocus	pfc_SelectAll
pfc_InvertSelection	

GetFocus

Description

Updates the parent window so it can set MicroHelp.

Usage

This event calls the pfc_ControlGotFocus event in the parent.

pfc_InvertSelection

Description

Inverts the rows selected in the PictureListBox. Unselected rows become selected; selected rows are cleared.

Return value — Integer. Returns the number of selected rows if the event succeeds and 0 if this is an invalid operation.

Usage — The message router calls this event when the user selects Edit>Invert Selection from the menu bar.

pfc_SelectAll

Description — Selects all rows in the PictureListBox.

Return value — Integer. Returns the number of selected rows if the event succeeds and 0 if this is an invalid operation.

Usage — The message router calls this event when the user selects Edit>Select All from the menu bar.

Functions

U_plb includes precoded object functions:

of_GetParentWindow	of_MessageBox
of_GetSelected	

of_GetParentWindow

Description — Returns a reference to the parent window.

Access — Public

Syntax — *instancename*.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_plb
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value — Integer. Returns 1 if the function succeeds and -1 if there is no parent window. If no parent window is found, *window* returns NULL.

Usage — The u_plb GetFocus event calls this function.

Examples

This example is from the u_plb GetFocus event:

```
Window   lw_parent

IF gnv_app.of_GetMicrohelp() THEN
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
      lw_parent.Dynamic Event &
         pfc_ControlGotFocus(this)
   END IF
END IF
```

of_GetSelected

Description

Populates a passed object with all selected entries in the PictureListBox.

Access

Public

Syntax

instancename.**of_GetSelected** (*selecteditems*)

Argument	Description
instancename	Instance name of u_plb
selecteditems	N_cst_itemattrib array into which the function places selected items (passed by reference)

Return value

Integer. Returns the number of selected items if the function succeeds and 0 if this is an invalid operation.

Usage

Call this function to access selected items.

Examples

This example calls the of_GetSelected function:

```
n_cst_itemattrib   lnv_items[ ]
Integer   li_selected

li_selected = plb_list.of_GetSelected(lnv_items)
IF li_selected = 0 THEN
   MessageBox("PLB", "Invalid PLB operation")
   Return
END IF
// Logic to read lnv_items follows
...
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id*, *title*, *message*, *icon*, *button*, *default*)

Argument	Description
instancename	Instance name of u_plb
id	String specifying an ID for the message
title	String specifying the title of the PictureListBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in PictureListBoxes.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('plb_error','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

u_rb

Description — RadioButton visual user object ancestor.

Ancestry

pfc_u_rb
u_rb
Events
GetFocus
Functions
Public of_GetParentWindow
Protected of_MessageBox

Library — PFCMAIN.PBL
PFEMAIN.PBL

Object relationships — PFC visual user objects are designed to be used with windows that are descendants of w_master.

Usage — Use this visual object in windows instead of the standard PowerBuilder RadioButton control. U_rb event scripts provide integration with PFC menus.

See also — u_cbx

Events

U_rb includes a precoded event script:

GetFocus

GetFocus

Description — Updates the parent window so it can set MicroHelp.

Usage — This event calls the pfc_ControlGotFocus event in the parent.

Functions

U_rb includes precoded object functions:

of_GetParentWindow
of_MessageBox

of_GetParentWindow

Description

Returns a reference to the parent window.

Access

Public

Syntax

instancename.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_rb
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if there is no parent window. If no parent window is found, *window* returns NULL.

Usage

The u_rb GetFocus event calls this function.

Examples

This example is from the u_rb GetFocus event:

```
Window   lw_parent

IF gnv_app.of_GetMicrohelp() THEN
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
      lw_parent.Dynamic Event &
         pfc_ControlGotFocus(this)
   END IF
END IF
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of u_rb

Argument	Description
id	String specifying an ID for the message
title	String specifying the title of the RadioButton
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in RadioButtons.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('rb_error','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

u_rte

Description

RichTextEdit visual user object ancestor.

Ancestry

pfc_u_rte
u_rte

Events

Destructor	pfc_LastPage	pfc_PrintPreview
FileExists	pfc_NextPage	pfc_ReplaceDlg
GetFocus	pfc_Open	pfc_Ruler
pfc_Clear	pfc_Paste	pfc_Save
pfc_Copy	pfc_PrePrintDlg	pfc_SaveAs
pfc_Cut	pfc_PreRMBMenu	pfc_SelectAll
pfc_FindDlg	pfc_PreviousPage	pfc_Undo
pfc_FirstPage	pfc_Print	PrintHeader
pfc_InsertFile	pfc_PrintDlg	RButtonUp
pfc_InsertPicture	pfc_PrintImmediate	

Object Functions

Public	of_GetContinuousPages	Public	of_SetFileName
Public	of_GetFileName	Public	of_SetFind
Public	of_GetIgnoreFileExists	Public	of_SetIgnoreFileExists
Public	of_GetPageInputField	Public	of_SetPageInputField
Public	of_GetParentWindow	Public	of_SetStartPageNumber
Public	of_GetStartPageNumber	Public	of_SetTextStyleBold
Public	of_GetTextStyle	Public	of_SetTextStyleItalic
Public	of_InsertDocument	Public	of_SetTextStyleStrikeout
Public	of_InsertPicture	Public	of_SetTextStyleSubscript
Public	of_MessageBox	Public	of_SetTextStyleSuperscript
Public	of_OpenDocument	Public	of_SetTextStyleUnderline
Public	of_SetContinuousPages		

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

PFC visual user objects are designed to be used with windows that are descendants of w_master. U_rte also uses:

F_SetFileSrv
M_edit
N_cst_filesrv
N_cst_rtefind
N_cst_textstyleattrib
S_printdlgattrib

Usage

Use this visual user object in windows instead of the standard PowerBuilder RichTextEdit control. U_rte event scripts provide integration with PFC menus.

U_rte is a window control

U_rte is a subclassed version of the RichTextEdit window control. It is not a DataWindow control using the Rich Text presentation style.

U_rte is fully integrated with menus that descend from PFC's m_master menu, including events that respond to the appropriate menu items. Additionally, u_rte includes right mouse button support. If the user clicks the right mouse button over a u_rte-based RichTextEdit control, the m_edit menu displays. M_edit allows you to perform basic editing functions.

To use u_rte:

1 Place a u_rte user object in your window.

2 (Optional) Enable the find service:

```
rte_doc.of_SetFind(TRUE)
```

3 Associate a document with the RichTextEdit control. This example uses the of_OpenDocument function to display a dialog box prompting the user for the file:

```
String  ls_document

rte_doc.of_OpenDocument(ls_document)
```

4 Call additional u_rte events and functions, as needed.

FOR INFO For more information on the RichTextEdit control, see the *PowerBuilder User's Guide*.

See also

n_cst_rtefind
u_dw

Instance variables

U_rte includes instance variables:

Instance variable	Description	Data type	Access	Usage
ib_continuouspages	Indicates whether continuous page numbering is enabled when printing	Boolean	Protected	Use of_GetContinuousPages and of_SetContinuousPages to access
ib_ignorefileexists	Indicates whether u_rte asks the user to confirm when saving a file that already exists	Boolean	Protected	Use of_GetIgnoreFileExists and of_SetIgnoreFileExists to access

Instance variable	Description	Data type	Access	Usage
ib_ongoingfind	Indicates whether a find operation is in progress	Boolean	Protected	Internal
ib_rmbmenu	Indicates whether right mouse button support is enabled	Boolean	Protected	When this is TRUE, PFC displays the m_edit popup menu when a user presses the right mouse button over this control To disable right mouse button support, set this variable to FALSE in the control's Constructor event
il_currentinstance	Tracks continuous printing	Long	Protected	Internal
il_currentprintpage	Tracks current page number	Long	Protected	Internal
il_startpagenumber	Specifies the first page that receives a page number	Boolean	Protected	Use of_GetStartPageNumber and of_SetStartPageNumber to access
inv_filesrv	Reference variable for the file service	n_cst_filesrv	Protected	Internal
inv_find	Structure containing information used in find and replace processing	n_cst_rtefind	Public	Internal
is_filename	Name of the current file	String	Protected	Use the of_GetFileName and of_SetFileName functions to access
is_pageinputfield	Specifies the name of the field that PFC uses to place the page number	String	Protected	Use of_GetPageInputField and of_SetPageInputField to access

Events

U_rte includes precoded event scripts:

Destructor	pfc_PreRmbMenu
FileExists	pfc_PreviousPage
GetFocus	pfc_Print
pfc_Clear	pfc_PrintDlg
pfc_Copy	pfc_PrintImmediate
pfc_Cut	pfc_PrintPreview
pfc_FindDlg	pfc_ReplaceDlg
pfc_FirstPage	pfc_Ruler
pfc_InsertFile	pfc_Save
pfc_InsertPicture	pfc_SaveAs
pfc_LastPage	pfc_SelectAll
pfc_NextPage	pfc_Undo
pfc_Open	PrintHeader
pfc_Paste	RButtonUp
pfc_PrePrintDlg	

Destructor

Description — Destroys the inv_filesrv instance.

Usage — This event executes when the u_rte-based control is destroyed or the window closes.

FileExists

Description — Prompts the user that the file to be saved already exists. If the ib_ignorefileexists instance variable is TRUE, PFC does not prompt the user.

Usage — This event executes when the user saves a file that already exists.

GetFocus

Description — Updates the parent window so it can set MicroHelp.

Usage — This event calls the pfc_ControlGotFocus event in the parent.

pfc_Clear

Description — Deletes selected text.

Return value

Integer. Returns the number of characters deleted if the event succeeds and -1 if an error occurs.

Usage

This event is triggered when a RichTextEdit control based on u_rte has focus and the user selects Edit>Clear from the menu bar of a menu descended from the PFC m_master menu.

pfc_Copy

Description

Copies selected text to the clipboard.

Return value

Integer. Returns the number of characters copied if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a RichTextEdit control based on u_rte has focus and the user selects Edit>Copy from the menu bar of a menu descended from the PFC m_master menu. This event is also called by the m_edit popup menu.

pfc_Cut

Description

Deletes selected text and stores it on the clipboard.

Return value

Integer. Returns the number of characters removed if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a RichTextEdit control based on u_rte has focus and the user selects Edit>Cut from the menu bar of a menu descended from the PFC m_master menu. This event is also called by the m_edit popup menu.

pfc_FindDlg

Description

Displays a Find dialog box by calling the n_cst_rtefind service's pfc_FindDlg event.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is called when the user selects Edit>Find from the menu bar of a menu descended from the PFC m_master menu.

pfc_FirstPage

Description	Scrolls to the first page.
Return value	Integer. Returns 1 if the event succeeds and -1 if an error occurs.
	Integer. Returns 1 if the event succeeds and -1 if an error occurs.
Usage	The message router calls this event when a RichTextEdit control based on u_rte has focus and the user selects View>First Page from the menu bar of a menu descended from the PFC m_master menu.

pfc_InsertFile

Description	Displays a dialog box prompting the user to choose a file to be copied at the current insertion point.
Return value	Integer. Returns 1 if the event succeeds, 0 if the user cancels out of the dialog box, and -1 if an error occurs.
Usage	The message router calls this event when a RichTextEdit control based on u_rte has focus and the user selects Insert>File from the menu bar of a menu descended from the PFC m_master menu.

pfc_InsertPicture

Description	Displays a dialog box prompting the user to choose a bitmap to be copied at the current insertion point.
Return value	Integer. Returns 1 if the event succeeds, 0 if the user cancels out of the dialog box, and -1 if an error occurs.
Usage	The message router calls this event when a RichTextEdit control based on u_rte has focus and the user selects Insert>Picture from the menu bar of a menu descended from the PFC m_master menu.

pfc_LastPage

Description	Scrolls to the last page.
Return value	Integer. Returns the number of the last page if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a RichTextEdit control based on u_rte has focus and the user selects View>Last Page from the menu bar of a menu descended from the PFC m_master menu.

pfc_NextPage

Description

Scrolls to the next page.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a RichTextEdit control based on u_rte has focus and the user selects View>Next Page from the menu bar of a menu descended from the PFC m_master menu.

pfc_Open

Description

Prompts the user for a file to open and places the specified file in the RichTextEdit control. If there is already a document in the control, the function prompts the user to save it.

Return value

Integer. Returns 1 if the event succeeds, 0 if the user cancels out of the dialog box, and -1 if an error occurs.

Usage

The message router calls this event when a RichTextEdit control based on u_rte has focus and the user selects File>Open from the menu bar of a menu descended from the PFC m_master menu.

pfc_Paste

Description

Inserts (pastes) the contents of the clipboard at the insertion point.

Return value

Integer. Returns the number of characters that were pasted if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a RichTextEdit control based on u_rte has focus and the user selects Edit>Paste from the menu bar of a menu descended from the PFC m_master menu.

pfc_PrePrintDlg

Description

Empty user event allowing you to modify the properties passed to the n_cst_platform of_PrintDlg function.

Syntax

instancename.EVENT **pfc_PrePrintDlg** (*attributes*)

Argument	Description
instancename	Instance name of u_rte
attributes	S_printdlgattrib variable into which the event places additional printing information. This argument is accessed through the *astr_printdlg* argument (passed by reference)

Usage

This event is called by pfc_PrintDlg before calling the n_cst_platform of_PrintDlg function.

You can use this event to modify or extend the information passed in the s_printdlgattrib structure.

Examples

This example contains code you might add to the pfc_PrePrintDlg event:

```
// Default copies to 1
astr_printdlg.l_copies = 1
```

pfc_PreRmbMenu

Description

User event allowing you to modify m_edit contents before display.

Syntax

instancename.EVENT **pfc_PreRmbMenu** (*editmenu*)

Argument	Description
instancename	Instance name of u_rte
editmenu	M_edit variable containing the popup menu to be displayed (passed by reference)

Usage

You can add logic to this event to selectively enable and disable m_edit menu items.

pfc_PreviousPage

Description

Scrolls to the Previous page.

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

The message router calls this event when a RichTextEdit control based on u_rte has focus and the user selects View>Previous Page from the menu bar of a menu descended from the PFC m_master menu.

pfc_Print

Description

Calls the pfc_PrintDlg function and prints the RichTextEdit control, as specified in the Print dialog box.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is triggered when the RichTextEdit control has focus and the user selects File>Print from the menu bar of a menu descended from the PFC m_master menu.

pfc_PrintDlg

Description

Initializes the s_printdlgattrib structure with the RichTextEdit control's current settings, displays the Print dialog box by calling the n_cst_platform of_PrintDlg function, and resets the settings, as specified by the user.

Syntax

instancename.EVENT **pfc_PrintDlg** (*attributes*)

Argument	Description
instancename	Instance name of u_rte
attributes	S_printdlgattrib variable into which the event places printing information. This argument is accessed through the *astr_printdlg* argument (passed by reference)

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is called by the pfc_Print event.

pfc_PrintImmediate

Description

Prints the current RichTextEdit control without displaying the Print dialog box.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is called when the RichTextEdit control has focus and the user selects File>Print Immediate from the menu bar of a menu descended from the PFC m_master menu.

pfc_PrintPreview

Description	Toggles between print preview and edit mode.
Return value	Boolean. Returns TRUE if the RichTextEdit control has been placed in preview mode and FALSE if it has been placed in edit mode.
Usage	The message router calls this event when a RichTextEdit control based on u_rte has focus and the user selects File>Print Preview from the menu bar of a menu descended from the PFC m_master menu.

pfc_ReplaceDlg

Description	Displays a Replace dialog box by calling the n_cst_rtefind service's pfc_ReplaceDlg event.
Return value	Integer. Returns 1 if the event succeeds and -1 if an error occurs.
Usage	This event is called when the user selects View>Replace from the menu bar of a menu descended from the PFC m_master menu.

pfc_Ruler

Description	Toggles display of the ruler and tab bar.
Return value	Boolean. Returns TRUE if the ruler has been displayed and FALSE if it has been hidden.
Usage	The message router calls this event when a RichTextEdit control based on u_rte has focus and the user selects View>Ruler from the menu bar of a menu descended from the PFC m_master menu.

pfc_Save

Description	Saves the current contents of the RichTextEdit control. If the current contents is not associated with a file, the function displays a dialog box prompting the user to specify a filename.
Return value	Integer. Returns 1 if the event succeeds, 0 if the user cancels out of the dialog box, and -1 if an error occurs.

Usage

The message router calls this event when a RichTextEdit control based on u_rte has focus and the user selects File>Save from the menu bar of a menu descended from the PFC m_master menu.

pfc_SaveAs

Description

Saves the current contents of the RichTextEdit control, prompting the user to specify a filename.

Return value

Integer. Returns 1 if the event succeeds, 0 if the user cancels out of the dialog box, and -1 if an error occurs.

Usage

The message router calls this event when a RichTextEdit control based on u_rte has focus and the user selects File>Save As from the menu bar of a menu descended from the PFC m_master menu.

pfc_SelectAll

Description

Selects all text in the RichTextEdit control.

Return value

Integer. Returns the number of characters selected if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a RichTextEdit control based on u_rte has focus and the user selects Edit>Select All from the menu bar of a menu descended from the PFC m_master menu.

pfc_Undo

Description

Cancels the last change to the RichTextEdit control, restoring the text to the content before the last change.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a RichTextEdit control based on u_rte has focus and the user selects Edit>Undo from the menu bar of a menu descended from the PFC m_master menu.

PrintHeader

Description

Sets page numbers for printing.

Usage

This event executes when you print the RichTextEdit control.

RButtonUp

Description

Displays the m_edit popup menu.

Usage

This event executes when the user releases the right mouse button over a control based on u_rte.

Functions

U_rte includes precoded object functions:

of_GetContinuousPages	of_SetFileName
of_GetFileName	of_SetFind
of_GetIgnoreFileExists	of_SetIgnoreFileExists
of_GetPageInputField	of_SetPageInputField
of_GetParentWindow	of_SetStartPageNumber
of_GetStartPageNumber	of_SetTextStyleBold
of_GetTextStyle	of_SetTextStyleItalic
of_InsertDocument	of_SetTextStyleStrikeout
of_InsertPicture	of_SetTextStyleSubscript
of_MessageBox	of_SetTextStyleSuperscript
of_OpenDocument	of_SetTextStyleUnderline
of_SetContinuousPages	

of_GetContinuousPages

Description

Reports whether PFC uses continuous page numbering when the RichTextEdit control is shared to a DataWindow.

Access

Public

Syntax

instancename.**of_GetContinuousPages** ( )

Argument	Description
instancename	Instance name of u_rte

Return value

Boolean. Returns TRUE if continuous page numbering is enabled and FALSE if it is not.

Examples

This example calls the of_GetContinuousPages function:

```
IF rte_doc.of_GetContinuousPages() THEN
   MessageBox("RTE", &
      "Continuous page numbering is enabled")
ELSE
   MessageBox("RTE", &
      "Continuous page numbering is disabled")
END IF
```

of_GetFileName

Description

Retrieves the name of the file associated with the RichTextEdit control.

Access

Public

Syntax

instancename.**of_GetFileName** ( )

Argument	Description
instancename	Instance name of u_rte

Return value

String. Returns the current filename or an empty string if one has not been set with the of_SetFilename function.

Examples

This example calls the of_GetFileName function:

```
String   ls_filename

ls_filename = rte_doc.of_GetFileName()
MessageBox("RTE", "Filename is " + ls_filename)
```

of_GetIgnoreFileExists

Description

Reports whether PFC displays a message before overwriting an existing file.

Access

Public

Syntax

instancename.**of_GetIgnoreFileExists** ( )

Argument	Description
instancename	Instance name of u_rte

Return value

Boolean. Returns TRUE if PFC displays no message before overwriting an existing file and FALSE if PFC prompts the user before overwriting an existing file.

Examples

This example calls the of_GetIgnoreFileExists function:

```
IF rte_doc.of_GetIgnoreFileExists() THEN
  MessageBox("RTE", "No prompt before overwriting")
ELSE
  MessageBox("RTE", "PFC prompts before overwriting")
END IF
```

of_GetPageInputField

Description

Retrieves the name of the field PFC uses to place the page number.

Access

Public

Syntax

instancename.**of_GetPageInputField** ( )

Argument	Description
instancename	Instance name of u_rte

Return value

String. Returns the name of the field PFC uses to place the page number.

Examples

This example calls the of_GetPageInputField function:

```
String   ls_field

ls_field = rte_doc.of_GetPageInputField( )
MessageBox("RTE", "Page input field is " + ls_field)
```

of_GetParentWindow

Description

Returns the parent of the current window.

Access

Public

Syntax

instancename.**of_GetParentWindow** (*parent*)

Argument	Description
instancename	Instance name of u_rte
parent	Window variable into which the function places the parent of the current window (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the GetFocus event.

```
Window  lw_parent

IF gnv_app.of_GetMicrohelp() THEN
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
      lw_parent.Dynamic Event &
         pfc_ControlGotFocus (this)
   END IF
END IF
```

of_GetStartPageNumber

Description

Retrieves the number of the page on which page numbers are initially displayed. For example, if this value is 3, the first two pages do not display page numbers and the third page displays a 3.

Access

Public

Syntax

instancename.**of_GetStartPageNumber ( )**

Argument	Description
instancename	Instance name of u_rte

Return value

Long. Returns the number of the page upon which page numbers are initially displayed

Examples

This example calls the of_GetStartPageNumber function:

```
Long   ll_return

ll_return = rte_doc.of_GetStartPageNumber()
MessageBox("RTE", &
   "Start page number is " + String(ll_return))
```

of_GetTextStyle

Description

Calculates all text style settings for the currently selected text.

Access

Public

Syntax

instancename.**of_GetTextStyle** (*textstyle*)

Argument	Description
instancename	Instance name of u_rte
textstyle	N_cst_textstyleattrib instance into which the function places text style settings (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function to determine text style settings for the currently selected text.

Examples

This example calls the of_GetTextStyle function:

```
n_cst_textstyleattrib  lnv_style
Integer  li_return
n_cst_conversion  lnv_conv

li_return = rte_doc.of_GetTextStyle &
   (lnv_style)
IF li_return = -1 THEN
   MessageBox("RTE", "GetTextStyle error")
ELSE
   MessageBox("RTE", &
     + "Text styles are as follows:~r~n" &
     + "Bold: " &
     +lnv_conv.of_String(lnv_style.ib_bold) &
     + "~r~nItalic: " &
     + lnv_conv.of_String(lnv_style.ib_italic) &
     + "~r~nStrikeout: " &
     + lnv_conv.of_String(lnv_style.ib_strikeout) &
     + "~r~nSubscript: " &
     + lnv_conv.of_String(lnv_style.ib_subscript) &
     + "~r~nSuperscript: " &
     + lnv_conv.of_String(lnv_style.ib_superscript) &
     + "~r~nUnderlined: " &
     + lnv_conv.of_String(lnv_style.ib_underlined))
END IF
```

of_InsertDocument

Description

Displays the Insert File dialog box, prompting the user for the name of the file to copy into the current document, placing it at the current insertion point.

Access

Public

Syntax

instancename.**of_InsertDocument** (*filename*)

Argument	Description
instancename	Instance name of u_rte
filename	String into which the function places the fully qualified name of the file inserted into the document

Return value

Integer. Returns 1 if the function succeeds, 0 if the user cancels out of the dialog box, and -1 if an error occurs.

Examples

This example calls the of_InsertDocument function:

```
Integer  li_return
String   ls_filename

li_return = rte_doc.of_InsertDocument(ls_filename)
IF li_return = -1 THEN
   MessageBox("RTE", "Insert file error")
ELSE
   gnv_app.of_GetFrame().SetMicroHelp &
      ("Inserted file: " + String(ls_filename))
END IF
```

of_InsertPicture

Description

Displays the Insert Picture dialog box, prompting the user for the name of the bitmap to copy into the current document, placing it at the current insertion point.

Access

Public

Syntax

instancename.**of_InsertPicture** (*bitmap*)

Argument	Description
instancename	Instance name of u_rte

Argument	Description
bitmap	String into which the function places the fully qualified name of the bitmap inserted into the document

Return value

Integer. Returns 1 if the function succeeds, 0 if the user cancels out of the dialog box, and -1 if an error occurs.

Examples

This example calls the of_InsertPicture function:

```
Integer  li_return
String   ls_filename

li_return = rte_doc.of_InsertPicture(ls_filename)
IF li_return = -1 THEN
   MessageBox("RTE", "Insert file error")
ELSE
   gnv_app.of_GetFrame().SetMicroHelp &
      ("Inserted file: " + String(ls_filename))
END IF
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id*, *title*, *message*, *icon*, *button*, *default*)

Argument	Description
instancename	Instance name of u_rte
id	String specifying an ID for the message
title	String specifying the title of the RichTexEdit control
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!

Argument	Description
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in RichTextEdit controls.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('rte_error','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

of_OpenDocument

Description

Displays the Open dialog box, prompting the user for the name of the file to place into the RichTextEdit control. This function prompts you before discarding the control's current contents.

Access

Public

Syntax

instancename.**of_OpenDocument** (*filename*)

Argument	Description
instancename	Instance name of u_rte
filename	String into which the function places the fully qualified name of the selected file

Return value

Integer. Returns 1 if the function succeeds, 0 if the user cancels out of the dialog box, and -1 if an error occurs.

Examples

This example calls the of_OpenDocument function:

```
String   ls_filename
Integer  li_return

li_return = rte_doc.of_OpenDocument(ls_filename)
IF li_return = -1 THEN
   MessageBox("RTE", "File open error")
   Return
END IF
```

of_SetContinuousPages

Description

Specifies whether PFC uses continuous page numbering when the RichTextEdit control is shared to a DataWindow.

Access

Public

Syntax

instancename.**of_SetContinuousPages** (*boolean*)

Argument	Description
instancename	Instance name of u_rte
boolean	Boolean specifying whether continuous page numbering is used (TRUE) or not (FALSE)

Return value

None

Examples

This example calls the of_SetContinuousPages function:

```
rte_doc.of_SetContinuousPages(TRUE)
```

of_SetFileName

Description

Specifies the name of the file associated with the RichTextEdit control.

Access

Public

Syntax

instancename.**of_SetFileName** (*filename*)

Argument	Description
instancename	Instance name of u_rte

Argument	Description
filename	String specifying the fully qualified name of the file displayed in the RichTextEdit control

Return value

None

Usage

The of_OpenDocument function automatically specifies the name of the file associated with the RichTextEdit control. But if you open a document with of_InsertDocument or the PowerScript InsertDocument function, you should call this function to set the filename.

Examples

This example calls the of_SetFileName function:

```
Integer  li_return
String   ls_filename

li_return = rte_doc.of_InsertDocument(ls_filename)
IF li_return = -1 THEN
   MessageBox("RTE", "Insert file error")
ELSE
   rte_doc.of_SetFileName(ls_filename)
END IF
```

of_SetFind

Description

Enables or disables n_cst_rtefind, which provides the find and replace service.

Access

Public

Syntax

instancename.**of_SetFind** (*boolean*)

Argument	Description
instancename	Instance name of u_rte
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the find and replace service

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_rtefind. This instance is named inv_find.

Examples

This example calls the of_SetFind function to enable the RichTextEdit find and replace service:

```
rte_doc.of_SetFind(TRUE)
```

of_SetIgnoreFileExists

Description

Specifies whether PFC displays a message before overwriting an existing file.

Access

Public

Syntax

instancename.**of_SetIgnoreFileExists** (*boolean*)

Argument	Description
instancename	Instance name of u_rte
boolean	Boolean specifying whether PFC displays a message before overwriting an existing file (TRUE) or not (FALSE)

Return value

None

Examples

This example calls the of_SetIgnoreFileExists function:

```
String   ls_filename
Integer  li_return

li_return = &
   rte_doc.of_OpenDocument(ls_filename)
IF li_return = -1 THEN
   MessageBox("RTE", "File open error")
   Return
ELSE
   rte_doc.of_SetIgnoreFileExists(TRUE)
END IF
```

of_SetPageInputField

Description

Specifies the name of the field PFC uses to place the page number.

Access

Public

Syntax

instancename.**of_SetPageInputField** ( )

Argument	Description
instancename	Instance name of u_rte

Return value

None

Examples

This example calls the of_SetPageInputField function:

```
rte_doc.of_SetContinuousPages(TRUE)
rte_doc.of_SetPageInputField("PAGE NUMBER")
rte_doc.of_SetStartPageNumber(2)
```

of_SetStartPageNumber

Description

Specifies the number of the page upon which page numbers are initially displayed. For example, if this value is 3, the first two pages do not display page numbers and the third page displays a 3.

Access

Public

Syntax

instancename.**of_SetStartPageNumber** (*firstpage*)

Argument	Description
instancename	Instance name of u_rte
firstpage	Long specifying the number of the page upon which page numbers are initially displayed

Return value

None

Examples

This example calls the of_SetStartPageNumber function:

```
rte_doc.of_SetContinuousPages(TRUE)
rte_doc.of_SetPageInputField("PAGE NUMBER")
rte_doc.of_SetStartPageNumber(2)
```

of_SetTextStyleBold

Description

Sets the currently selected text to bold, leaving all other text attributes as they were.

Access

Public

Syntax

instancename.**of_SetTextStyleBold** (*boolean*)

Argument	Description
instancename	Instance name of u_rte
boolean	Boolean indicating whether to add the bold attribute (TRUE) or remove it (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetTextStyleBold function:

```
Integer  li_return

li_return = rte_doc.of_SetTextStyleBold(TRUE)
IF li_return = -1 THEN
   MessageBox("RTE","Error setting text style")
END IF
```

of_SetTextStyleItalic

Description

Sets the currently selected text to italic, leaving all other text attributes as they were.

Access

Public

Syntax

instancename.**of_SetTextStyleItalic** (*boolean*)

Argument	Description
instancename	Instance name of u_rte
boolean	Boolean indicating whether to add the italic attribute (TRUE) or remove it (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetTextStyleItalic function:

```
Integer  li_return

li_return = rte_doc.of_SetTextStyleItalic(TRUE)
IF li_return = -1 THEN
   MessageBox("RTE","Error setting text style")
END IF
```

of_SetTextStyleStrikeout

Description

Sets the currently selected text to strikeout, leaving all other text attributes as they were.

Access

Public

Syntax

instancename.**of_SetTextStyleStrikeout** (*boolean*)

Argument	Description
instancename	Instance name of u_rte
boolean	Boolean indicating whether to add the strikeout attribute (TRUE) or remove it (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetTextStyleStrikeout function:

```
Integer  li_return

li_return = rte_doc.of_SetTextStyleStrikeout(TRUE)
IF li_return = -1 THEN
   MessageBox("RTE","Error setting text style")
END IF
```

of_SetTextStyleSubscript

Description

Sets the currently selected text to subscript, leaving all other text attributes as they were.

Access

Public

Syntax

instancename.**of_SetTextStyleSubscript** (*boolean*)

Argument	Description
instancename	Instance name of u_rte
boolean	Boolean indicating whether to add the subscript attribute (TRUE) or remove it (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetTextStyleSubscript function:

```
Integer  li_return

li_return = rte_doc.of_SetTextStyleSubscript(TRUE)
IF li_return = -1 THEN
   MessageBox("RTE","Error setting text style")
END IF
```

of_SetTextStyleSuperscript

Description

Sets the currently selected text to superscript, leaving all other text attributes as they were.

Access

Public

Syntax

instancename.**of_SetTextStyleSuperscript** (*boolean*)

Argument	Description
instancename	Instance name of u_rte
boolean	Boolean indicating whether to add the superscript attribute (TRUE) or remove it (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetTextStyleSuperscript function:

```
Integer   li_return

li_return = rte_doc.of_SetTextStyleSuperscript(TRUE)
IF li_return = -1 THEN
   MessageBox("RTE","Error setting text style")
END IF
```

of_SetTextStyleUnderline

Description

Sets the currently selected text to underline, leaving all other text attributes as they were.

Access

Public

Syntax

instancename.**of_SetTextStyleUnderline** (*boolean*)

Argument	Description
instancename	Instance name of u_rte
boolean	Boolean indicating whether to add the underline attribute (TRUE) or remove it (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetTextStyleUnderline function:

```
Integer  li_return
```

```
li_return = rte_doc.of_SetTextStyleUnderline(TRUE)
IF li_return = -1 THEN
   MessageBox("RTE","Error setting text style")
END IF
```

u_sle

Description

SingleLineEdit visual user object ancestor.

Ancestry

pfc_u_sle
u_sle
Events

GetFocus	pfc_PreRmbMenu
pfc_Clear	pfc_SelectAll
pfc_Copy	pfc_Undo
pfc_Cut	RButtonUp
pfc_Paste	

Object functions

Public	of_GetParentWindow
Protected	of_MessageBox

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

PFC visual user objects are designed to be used with windows that are descendants of w_master.

Usage

Use this visual object in windows instead of the standard PowerBuilder SingleLineEdit control. U_sle event scripts provide integration with PFC menus. Additionally, u_sle supports:

- **Cut, copy, paste, clear, and select all** These events are called by PFC menus.
- **Right mouse button support** The RButtonUp event enables you to use the right mouse button to perform editing actions. To disable right mouse button support, set ib_rmbmenu to FALSE in the SingleLineEdit's Constructor event.
- **Autoselect** This means that text is selected when a user tabs to the edit mask. To disable autoselect, set ib_autoselect to FALSE in the SingleLineEdit's Constructor event.

See also

u_mle

Instance variables

U_sle includes instance variables:

Instance variable	Description	Data type	Access	Usage
ib_autoselect	Indicates whether PFC selects text automatically when the control receives focus	Boolean	Protected	Set this to TRUE to enable autoselect (default is FALSE)
ib_rmbmenu	Indicates whether the m_edit menu displays when the user presses the right mouse button	Boolean	Protected	Set this to FALSE to disable right mouse button support (default is TRUE)

Events

U_sle includes precoded events:

GetFocus	pfc_PreRmbMenu
pfc_Clear	pfc_SelectAll
pfc_Copy	pfc_Undo
pfc_Cut	RButtonUp
pfc_Paste	

GetFocus

Description

Updates the parent window so it can set MicroHelp.

Usage

This event calls the pfc_ControlGotFocus event in the parent.

pfc_Clear

Description

Deletes selected text.

Return value

Integer. Returns the number of characters deleted if the event succeeds and -1 if an error occurs.

Usage

This event is triggered when a SingleLineEdit control based on u_sle has focus and the user selects Edit>Clear from the menu bar of a menu descended from the PFC m_master menu.

pfc_Copy

Description	Copies selected text to the clipboard.
Return value	Integer. Returns the number of characters copied if the event succeeds and -1 if an error occurs.
Usage	The message router calls this event when a SingleLineEdit control based on u_sle has focus and the user selects Edit>Copy from the menu bar of a menu descended from the PFC m_master menu. This event is also called by the m_edit popup menu.

pfc_Cut

Description	Deletes selected text and stored it on the clipboard.
Return value	Integer. Returns the number of characters removed if the event succeeds and -1 if an error occurs.
Usage	The message router calls this event when a SingleLineEdit control based on u_sle has focus and the user selects Edit>Cut from the menu bar of a menu descended from the PFC m_master menu. This event is also called by the m_edit popup menu.

pfc_Paste

Description	Inserts (pastes) the contents of the clipboard at the insertion point.
Return value	Integer. Returns the number of characters that were pasted if the event succeeds and -1 if an error occurs.
Usage	The message router calls this event when a SingleLineEdit control based on u_sle has focus and the user selects Edit>Paste from the menu bar of a menu descended from the PFC m_master menu. This event is also called by the m_edit popup menu.

pfc_PreRmbMenu

Description	User event allowing you to modify m_edit contents prior to display.
Syntax	*instancename*.EVENT **pfc_PreRmbMenu** (*editmenu*)

Argument	Description
instancename	Instance name of u_sle
editmenu	M_edit variable containing the popup menu to be displayed (passed by reference)

Usage

You can add logic to this event to selectively enable and disable m_edit menu items.

pfc_SelectAll

Description

Selects all text in the SingleLineEdit.

Return value

Integer. Returns the number of characters selected if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a SingleLineEdit control based on u_sle has focus and the user selects Edit>Select All from the menu bar of a menu descended from the PFC m_master menu. This event is also called by the m_edit popup menu.

pfc_Undo

Description

Cancels the last change to the SingleLineEdit control, restoring the text to the content before the last change.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a SingleLineEdit control based on u_sle has focus and the user selects Edit>UndoAll from the menu bar of a menu descended from the PFC m_master menu.

RButtonUp

Description

Displays the m_edit popup menu.

Usage

This event executes when the user releases the right mouse button over a control based on u_sle.

Functions

U_sle includes precoded object functions:

of_GetParentWindow
of_MessageBox

of_GetParentWindow

Description — Retrieves a reference to the parent window.

Access — Public

Syntax — *instancename*.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_sle
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value — Integer. Returns 1 if the function succeeds and -1 if there is no parent window. If no parent window is found, *window* returns NULL.

Usage — The u_sle GetFocus event calls this function.

Examples — This example is from the u_sle GetFocus event:

```
Window   lw_parent

IF gnv_app.of_GetMicrohelp() THEN
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
      lw_parent.Dynamic Event &
         pfc_ControlGotFocus (this)
   END IF
END IF
```

of_MessageBox

Description — Displays a MessageBox.

Access — Protected

Syntax — *instancename*.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of u_sle
id	String specifying an ID for the message
title	String specifying the title of the SingleLineEdit
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in SingleLineEdits.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('sle_error','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

u_st

Description

StaticText visual user object ancestor.

Ancestry

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

PFC visual user objects are designed to be used with windows that are descendants of w_master.

Usage

Use this visual object in windows instead of the standard PowerBuilder StaticText control. U_st event scripts provide integration with PFC menus.

Descendants

u_st_splitbar

See also

u_mle
u_sle

Events

U_st includes one precoded event:

GetFocus

GetFocus

Description

Updates the parent window so it can set MicroHelp.

Usage

This event calls the pfc_ControlGotFocus event in the parent.

Functions

U_st includes precoded object functions:

of_GetParentWindow
of_MessageBox

of_GetParentWindow

Description

Returns a reference to the parent window.

Access

Public

Syntax

instancename.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_st
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if there is no parent window. If no parent window is found, *window* returns NULL.

Usage

The u_st GetFocus event calls this function.

Examples

This example is from the u_st GetFocus event:

```
Window  lw_parent

IF gnv_app.of_GetMicrohelp() THEN
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
      lw_parent.Dynamic Event &
         pfc_ControlGotFocus (this)
   END IF
END IF
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of u_st
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in StaticText controls.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('st_error','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

u_tab

Description

Tab control visual user object ancestor.

Ancestry

pfc_u_tab
u_tab

Events

pfc_AcceptText	pfc_UpdatesPending
pfc_PostUpdate	pfc_Validation
pfc_Update	Resize
pfc_UpdatePrep	

Functions

Public	of_AcceptText	Public	of_SetLogicalUnitOfWork
Public	of_GetInfo	Public	of_SetResize
Public	of_GetParentWindow	Protected	of_SetUpdateable
Protected	of_GetUpdateObjects	Protected	of_SetUpdateObjects
Protected	of_IsAlwaysValidate	Protected	of_SetUpdateRequestor
Protected	of_IsUpdateable	Public	of_Update
Protected	of_MessageBox	Public	of_UpdatePrep
Public	of_PostUpdate	Public	of_UpdatesPending
Protected	of_SetAlwaysValidate	Public	of_Validation

Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use descendants of this visual user object in the User Object painter instead of the visual standard tab user object. Using the User Object painter, add u_tabpg-based tab pages, coding events and functions as needed.

U_tab is a self-updating object.

You typically do not place this object directly into the Window painter.

See also

n_cst_resize
u_tabpg

Instance variables

U_tab includes instance variables:

Instance variable	Description	Data type	Access	Usage
ib_alwaysvalidate	Controls whether the save process includes all objects in the validation process	Boolean	Protected	Set with of_SetAlwaysValidate (default is FALSE)

Instance variable	Description	Data type	Access	Usage
ib_isupdateable	Indicates whether the Tab control can be updated	Boolean	Protected	Set with of_SetUpdateable (default is FALSE)
inv_luw	Reference variable for logical unit of work service	n_cst_luw	Protected	Implements the update process
inv_resize	Reference variable for the resize service	n_cst_resize	Public	Use in dot notation to access n_cst_resize functions and attributes
ipo_pendingupdates[ ]	Objects that could be updated	PowerObject	Protected	Internal
ipo_updateobjects[ ]	Objects to be updated	PowerObject	Protected	Internal
ipo_updaterequestor	Owner of the save process	PowerObject	Protected	Internal
NO_ACTION	Constant set to 0	Integer	Public	Internal

Events

U_tab includes a precoded event:

pfc_AcceptText	pfc_UpdatesPending
pfc_PostUpdate	pfc_Validation
pfc_Update	Resize
pfc_UpdatePrep	

pfc_AcceptText

Description

Calls the n_cst_luw of_AcceptText function.

Syntax

instancename.Event **pfc_AcceptText** (*controls*, *focusonerror*)

Argument	Description
instancename	Instance name of u_tab
controls	PowerObject array containing the objects on which to accept text. This argument is accessed through the *apo_control* argument

Argument	Description
focusonerror	Boolean indicating whether focus should be set if an error occurs. This argument is accessed through the *ab_focusonerror* argument

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

The of_AcceptText function calls this event.

pfc_PostUpdate

Description

Calls the n_cst_luw of_PostUpdate function.

Syntax

instancename.Event **pfc_PostUpdate** (*controls*)

Argument	Description
instancename	Instance name of u_tab
controls	PowerObject array containing the objects on which to perform post-update processing. This argument is accessed through the *apo_control* argument

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

Internal.

pfc_Update

Description

Calls the n_cst_luw of_Update function to update the specified controls.

Syntax

instancename.Event **pfc_Update** (*controls, accepttext, resetflags*)

Argument	Description
instancename	Instance name of u_tab
controls	PowerObject array containing the objects to update. This argument is accessed through the *apo_control* argument
accepttext	Boolean specifying whether n_cst_luw should automatically perform an AcceptText prior to performing the Update (TRUE) or not (FALSE)
resetflags	Boolean specifying whether n_cst_luw should automatically reset DataWindow update flags (TRUE) or not (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

The of_Update function calls this event.

pfc_UpdatePrep

Description

Calls the n_cst_luw of_UpdatePrep function.

Syntax

instancename.Event **pfc_UpdatePrep** (*controls*)

Argument	Description
instancename	Instance name of u_tab
controls	PowerObject array containing the objects to update. This argument is accessed through the *apo_control* argument

Return value

Long. Return 1 if the update preparation succeeds and -1 to halt the update process.

Usage

The of_UpdatePrep function calls this function.

pfc_UpdatesPending

Description

Calls the n_cst_luw of_GetUpdatesPending function.

Syntax

instancename.Event **pfc_UpdatesPending** (*controls*, *pending*)

Argument	Description
instancename	Instance name of u_tab
controls	PowerObject array containing the objects to update. This argument is accessed through the *apo_control* argument
pending	PowerObject array into which the event places objects with pending updates. This argument is accessed through the *apo_pending* argument (passed by reference)

Return value

Integer. Returns 1 if there are pending updates, 0 if there are no pending updates, and -1 if an error occurs.

Usage

This event is called by the of_GetUpdatesPending function.

pfc_Validation

Description

Calls the n_cst_luw of_Validation function.

Syntax

instancename.Event **pfc_Validation** (*controls*)

Argument	Description
instancename	Instance name of u_tab
controls	PowerObject array containing the objects to validate. This argument is accessed through the *apo_control* argument

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is called by the of_Validation function.

Resize

Description

Triggers automatic resize processing, if enabled in n_cst_resize.

Usage

This event executes when the user resizes the window.

Functions

U_tab includes precoded object functions:

of_AcceptText	of_SetLogicalUnitOfWork
of_GetInfo	of_SetResize
of_GetParentWindow	of_SetUpdateable
of_GetUpdateObjects	of_SetUpdateObjects
of_IsAlwaysValidate	of_SetUpdateRequestor
of_IsUpdateable	of_Update
of_MessageBox	of_UpdatePrep
of_PostUpdate	of_UpdatesPending
of_SetAlwaysValidate	of_Validation

of_AcceptText

Description

Performs an AcceptText function for controls on the tab's tab pages.

Access

Public

Syntax

instancename.**of_AcceptText** (*focusonerror*)

Argument	Description
instancename	Instance name of u_tab
focusonerror	Boolean indicating whether PFC sets focus to the first item in error when an error occurs

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize accept text processing, extend the pfc_AcceptText event.

Examples

This example is from the n_cst_luw of_AcceptText function:

```
...
If lb_defined Then
   li_rc =  &
      lpo_tocheck.Function Dynamic of_AcceptText &
         (ab_focusonerror)
   If li_rc < 0 Then Return -1

...
```

of_GetInfo

Description

Retrieves object information.

Access

Public

Syntax

instancename.**of_GetInfo** (*infoobject*)

Argument	Description
instancename	Instance name of u_tab
infoobject	N_cst_infoattrib instance into which the function places information (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_GetInfo function:

```
n_cst_infoattrib  lnv_info

tab_1.of_GetInfo(lnv_info)
```

```
MessageBox("Info",  &
  "Description: " + lnv_info.is_description  &
    + ". Name: " + lnv_info.is_name)
```

of_GetParentWindow

Description

Returns a reference to the parent window.

Access

Public

Syntax

instancename.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_tab
window	Window variable into which the function places a reference to the parent window

Return value

Integer. Returns 1 if the function succeeds and -1 if there is no parent window. If no parent window is found, *window* returns NULL.

Usage

Call this function from within a tab to determine the parent window.

Examples

This example calls the of_GetParentWindow function:

```
Window  lw_parent

of_GetParentWindow(lw_parent)
IF IsValid(lw_parent) THEN
   ... // Continue processing
END IF
```

of_GetUpdateObjects

Description

Retrieves the current default array of objects affected by the update process.

Access

Protected

Syntax

instancename.**of_GetUpdateObjects** (*objects*)

Argument	Description
instancename	Instance name of u_tab
objects	PowerObject array into which the function places objects to be updated (passed by reference)

Return value

Integer. Returns the number of elements in the *objects* array if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_GetUpdateObjects function:

```
PowerObject  lpo_objs[ ]
Integer  li_return, li_count

li_return = this.of_GetUpdateObjects(lpo_objs)
FOR li_count = 1 to li_return
   IF lpo_objs[li_count] = ids_data THEN
      Return 1
   END IF
NEXT
li_return++
lpo_objs[li_return] = ids_data
Return this.of_SetUpdateObjects(lpo_objs)
```

of_IsAlwaysValidate

Description

Reports whether the default save process always performs validation, regardless of whether objects have updates pending.

Access

Public

Syntax

instancename.**of_IsAlwaysValidate ( )**

Argument	Description
instancename	Instance name of u_tab

Return value

Boolean. Returns TRUE if the default save process always performs validation and FALSE if it does not.

Examples

This example calls the of_IsAlwaysValidate function:

```
IF tab_1.of_IsAlwaysValidate() = TRUE THEN
   MessageBox("Tab", "Always validate")
ELSE
   MessageBox("Tab", "Sometimes validate")
END IF
```

of_IsUpdateable

Description

Reports whether controls on the Tab control's tab pages are updatable.

Access

Public

Syntax

instancename.**of_IsUpdateable** ( )

Argument	Description
instancename	Instance name of u_tab

Return value

Boolean. Returns TRUE if controls are updatable and FALSE if they is not.

Examples

This example is from the pfc_UpdatesPending event:

```
...
If Not of_IsUpdateable() Then Return NO_UPDATESPENDING
...
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of u_tab
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!

Argument	Description
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in Tab controls.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('tab_error','Save', &
   as_error, StopSign!, Ok!, 1)
```

of_PostUpdate

Description

Calls the pfc_PostUpdate event, which clears update flags and allows you to code additional post update processing.

Access

Public

Syntax

instancename.**of_PostUpdate ( )**

Argument	Description
instancename	Instance name of u_tab

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

To customize post update processing, extend the pfc_PostUpdate event.

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize post-update processing, extend the pfc_PostUpdate event.

Examples

This example is from the n_cst_luw of_PostUpdate function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_PostUpdate()
...
```

of_SetAlwaysValidate

Description

Specifies whether the default save process always performs validation, regardless of whether there are pending updates.

Access

Public

Syntax

instancename.**of_SetAlwaysValidate** (*boolean*)

Argument	Description
instancename	Instance name of u_tab
boolean	Boolean specifying whether the default save process always perform validation (TRUE) or only performs validation if a control has pending updates (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetAlwaysValidate function:

```
tab_1.of_SetAlwaysValidate(TRUE)
```

of_SetLogicalUnitOfWork

Description

Enables or disables n_cst_luw, which provides the logical unit of work service.

Access

Public

Syntax

instancename.**of_SetLogicalUnitOfWork** (*boolean*)

Argument	Description
instancename	Instance name of u_tab

Argument	Description
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) n_cst_luw

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_luw. This instance is named inv_luw. If you do not enable n_cst_luw, u_tab enables it automatically.

Examples

This example calls the of_SetLogicalUnitOfWork function:

```
tab_1.of_SetLogicalUnitOfWork(TRUE)
```

of_SetResize

Description

Enables or disables n_cst_resize, the resize service.

Access

Public

Syntax

instancename.**of_SetResize** (*boolean*)

Argument	Description
instancename	Instance name of u_tab
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) an instance of n_cst_resize

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_resize. This instance is named inv_resize.

Examples

This example calls the of_SetResize function in a window Open event to enable the resize service for tab_1:

```
tab_1.of_SetResize(TRUE)
```

of_SetUpdateable

Description

Specifies whether the Tab is updatable.

Access

Public

Syntax

instancename.**of_SetUpdateable** (*boolean*)

Argument	Description
instancename	Instance name of u_tab
boolean	Boolean indicating whether the Tab is updatable. All updatable Tabs are included in the default save processing

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetUpdateable function:

```
tab_1.of_SetUpdateable(FALSE)
```

of_SetUpdateObjects

Description

Sets a new default array containing objects for which updates are attempted.

Access

Protected

Syntax

instancename.**of_SetUpdateObjects** (*requester*)

Argument	Description
instancename	Instance name of u_tab
requester	PowerObject array containing the object to be updated

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

of_SetUpdateRequestor

Description

Creates a reference to the object requesting an update within a logical unit of work.

Access

Public

Syntax

instancename.**of_SetUpdateRequestor** (*requester*)

Argument	Description
instancename	Instance name of u_tab
requester	PowerObject containing the object requesting the update

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the of_Update function:

```
...
If IsValid(inv_luw) Then
   inv_luw.of_SetUpdateRequestor(apo_requestor)
End If
...
```

of_Update

Description

Calls the pfc_Update event.

Access

Public

Syntax

instancename.**of_Update** (*accept*, *resetflag* {, *requestor*})

Argument	Description
instancename	Instance name of u_tab
accept	Boolean indicating whether the Update function performs an AcceptText before saving rows to the database
resetflag	Boolean indicating whether the Update function resets the update flags
requestor	PowerObject containing the requestor object

Return value

Integer. Returns 1 if the update succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize update processing, extend the pfc_Update event.

Examples

This example is from the n_cst_luw of_Update function:

```
...
If lb_defined Then
  li_rc = lpo_tocheck.Function Dynamic of_Update &
    (ab_accepttext,  ab_resetflag, &
      lpo_updaterequestor)
  If li_rc < 0 Then Return -1
  Continue
```

```
End If
...
```

of_UpdatePrep

Description

Calls the pfc_UpdatePrep event.

Access

Public

Syntax

instancename.**of_UpdatePrep** ( )

Argument	Description
instancename	Instance name of u_tab

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize update preparation processing, extend the pfc_UpdatePrep event.

Examples

This example is from the n_cst_luw of_UpdatePrep function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_UpdatePrep ()
  If li_rc < 0 Then Return -1
  Continue
End If
...
```

of_UpdatesPending

Description

Calls the pfc_UpdatesPending event.

Access

Public

Syntax

instancename.**of_UpdatesPending** ( )

Argument	Description
instancename	Instance name of u_tab

Return value

Integer. Returns values as follows:

- **1** Updates are pending
- **0** No updates pending
- **-1** An error occurred

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize pending updates processing, extend the pfc_UpdatesPending event.

Examples

This example calls the of_UpdatesPending function:

```
...
If lb_defined Then
   la_rc = lpo_tocheck.Dynamic of_UpdatesPending ()
...
```

of_Validation

Description

Calls the pfc_Validation event.

Access

Public

Syntax

instancename.**of_Validation ( )**

Argument	Description
instancename	Instance name of u_tab

Return value

Integer. Returns 1 if the function succeeds and -1 if a validation error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize validation processing, extend the pfc_Validation event.

Examples

This example calls the of_Validation function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_Validation()
...
```

u_tvs

Description

Service-based TreeView visual user object ancestor. This object uses DataStores to display data in a TreeView.

U_tvs is a self-updating object.

Ancestry

pfc_u_tvs
u_tvs

Events

BeginDrag	pfc_NewItem	pfc_RenameItem
Destructor	pfc_Populate	pfc_Retrieve
DragDrop	pfc_PostUpdate	pfc_SetItemAttributes
EndLabelEdit	pfc_PreDeleteItem	pfc_Undo
GetFocus	pfc_PreInsertItem	pfc_Update
ItemExpanding	pfc_PreRefreshItem	pfc_UpdatePrep
KeyDown	pfc_PreRefreshLevel	pfc_UpdatesPending
pfc_AcceptText	pfc_PreRmbMenu	pfc_Validation
pfc_AddAll	pfc_Properties	RButtonUp
pfc_AddItem	pfc_RefreshItem	RightClicked
pfc_DeleteItem	pfc_RefreshLevel	
pfc_InsertItem		

Functions

Public	of_AcceptText	Public	of_Reset
Public	of_AddItem	Public	of_Retrieve
Public	of_CanUndo	Public	of_SetAlwaysValidate
Public	of_GetInfo	Public	of_SetBase
Public	of_GetNextLevel	Public	of_SetLevelSource
Protected	of_GetObjects	Public	of_SetLogicalUnitOfWork
Public	of_GetParentWindow	Public	of_SetPrint
Public	of_GetUpdateRequestor	Public	of_SetRmbMenu
Public	of_InsertItem	Public	of_SetUpdateable
Public	of_IsAlwaysValidate	Public	of_SetUpdateRequestor
Public	of_IsRmbMenu	Public	of_Update
Public	of_IsUpdateable	Public	of_UpdatePrep
Protected	of_MessageBox	Public	of_UpdatesPending
Public	of_Populate	Public	of_Validation
Public	of_PostUpdate		

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

PFC visual user objects are designed to be used with windows that are descendants of w_master. U_tvs also uses:

m_tvs
n_cst_infoattrib
n_ds
n_cst_luw
n_cst_tvsrv
n_cst_tvsrv_levelsource
n_cst_print

Usage

Use this visual user object in windows instead of the standard PowerBuilder TreeView control. U_tvs event scripts provide integration with PFC menus.

To use u_tvs:

1 Place a u_tvs user object in your window.

2 Enable TreeView services as needed:

```
this.of_SetPrint(TRUE)
this.of_SetLevelSource(TRUE)
```

3 Specify the data source for each TreeView level:

```
String  ls_sql
Integer  li_return

ls_keys[1] = "dept_id"
this.inv_levelsource.of_Register(1,  &
   "dept_name", "", "d_deptlist", SQLCA, "")
      ls_keys[1] = "emp_id"
this.inv_levelsource.of_Register(2,   &
   "emp_lname", ":parent.1.dept_id",  &
      "d_empbydept", SQLCA, "")
```

4 Specify additional TreeView display information as needed:

```
this.inv_levelsource.of_SetPictureColumn(1, "1")
this.inv_levelsource.of_SetSelectedPictureColumn &
   (1, "2")
this.inv_levelsource.of_SetPictureColumn(2, "4")
this.inv_levelsource.of_SetSelectedPictureColumn &
   (2, "5")
```

5 Retrieve rows for the first level of the TreeView:

```
this.event pfc_populate(0)
```

6 Extend the pfc_Retrieve event to implement retrieval arguments:

```
Integer  li_level, li_count
Any  la_args[20]

li_level = this.of_GetNextLevel(al_parent)
IF li_level = 2 THEN
   li_count = this.inv_levelsource.of_GetArgs  &
     (al_parent, li_level, la_args)
END IF
```

```
Return of_Retrieve(al_parent, la_args, ads_data)
```

7 Call additional u_tvs functions as needed.

FOR INFO For more information on the TreeView control, see the *PowerBuilder User's Guide*.

See also

m_tvs
u_lvs
n_cst_tvsrv
n_cst_tvsrv_levelsource
n_cst_tvsrv_print

Instance variables

U_tvs includes instance variables:

Instance variable	Description	Data type	Access	Usage
ib_alwaysvalidate	Controls whether the save process includes all objects in the validation process	Boolean	Protected	Set with of_SetAlwaysValidate (default is FALSE)
ib_isupdateable	Indicates whether the TreeView can be updated	Boolean	Protected	Set with of_SetUpdateable (default is FALSE)
ib_rmbmenu	Controls right mouse button support	Boolean	Protected	Set with of_SetRmbMenu
il_dragsource	Handle of the dragged item	Long	Public	PFC uses this instance variable to track a dragged TreeView item
il_dragtarget	Handle of drag target	Long	Public	PFC uses this instance variable to track a dragged TreeView item
il_rightclicked	Item clicked with the right mouse button	Long	Protected	Internal
im_view	Popup menu	m_tvs	Protected	Internal
inv_base	Reference variable for basic TreeView services	n_cst_tvsrv	Public	Use in dot notation to access n_cst_tvsrv functions and attributes

Instance variable	Description	Data type	Access	Usage
inv_levelsource	Reference variable for the TreeView data source service	n_cst_tvsrv_ levelsource	Public	Use in dot notation to access n_cst_tvsrv_levelsource functions and attributes
inv_luw	Reference variable for logical unit of work service	n_cst_luw	Protected	Implements the save process
inv_print	Reference variable for TreeView print service	n_cst_tvsrv_ print	Public	Use in dot notation to access n_cst_tvsrv_print functions and attributes
ipo_pendingupdates[ ]	Objects that could be updated	PowerObject	Protected	Internal
ipo_updaterequestor	Owner of the save process	PowerObject	Protected	Internal

Events

U_tvs includes precoded events:

BeginDrag	pfc_PreRefreshItem
Destructor	pfc_PreRefreshLevel
DragDrop	pfc_PreRmbMenu
EndLabelEdit	pfc_Properties
GetFocus	pfc_RefreshItem
KeyDown	pfc_RefreshLevel
ItemExpanding	pfc_RenameItem
pfc_AcceptText	pfc_Retrieve
pfc_AddAll	pfc_SearchCompare
pfc_AddItem	pfc_SetItemAttributes
pfc_DeleteItem	pfc_Undo
pfc_InsertItem	pfc_Update
pfc_NewItem	pfc_UpdatePrep
pfc_Populate	pfc_UpdatesPending
pfc_PostUpdate	pfc_Validation
pfc_PreDeleteItem	RButtonUp
pfc_PreInsertItem	RightClicked

BeginDrag

Description	Saves the handle of a dragged item in the il_dragsource public instance variable.
Usage	This event executes when the user drags a label.

Destructor

Description	Destroys all enabled service objects.
Usage	This event executes when the TreeView is destroyed or the window closes.

DragDrop

Description	Saves the handle of the dropped item in the il_dragtarget instance variable.
Usage	This event executes when the user releases a dragged object over the TreeView. You should extend this event to allow dragging and dropping of TreeView items from one branch to another. Be sure to verify that the drag source is the current TreeView control.

EndLabelEdit

Description	Updates the DataStore with user edits to the item label. This function updates the DataStore only. You must update the database explicitly, using the of_Update function or the w_master pfc_Save process.
Usage	This event executes when the user finishes editing an item label. If you are using computed columns to display item labels, override this event to update DataStore rows as appropriate.

GetFocus

Description	Updates the parent window so it can set MicroHelp.
Usage	This event executes when the control receives focus. It triggers the pfc_ControlGotFocus event in the parent.

KeyDown

Description

Checks to see if the DELETE key was pressed.

Usage

This event executes when the user presses a key and is not editing a label.

ItemExpanding

Description

Populates an item with its children.

Usage

This event executes when the user expands an item.

pfc_AcceptText

Description

Accepts text for the data displayed in a TreeView control, optionally setting focus if an error occurs.

Syntax

instancename.EVENT **pfc_AcceptText** (*controls, focusonerror*)

Argument	Description
instancename	Instance name of u_tvs
controls	PowerObject array containing the objects on which to accept text. This argument is accessed through the *apo_control* argument
focusonerror	Boolean indicating whether focus should be set if an error occurs. This argument is accessed through the *ab_focusonerror* argument

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

Extend this event to perform additional accept text processing.

pfc_AddAll

Description

Adds all rows in the passed DataStore under the specified TreeView item.

Syntax

instancename.EVENT **pfc_AddAll** (*parent, data*)

Argument	Description
instancename	Instance name of u_tvs

Argument	Description
parent	Long specifying the handle of the TreeView item under which the new items are added. This argument is accessed through the *al_parent* argument
data	N_ds-based DataStore containing rows to be added. This argument is accessed through the *ads_source* argument

Return value

Long. Returns the number of items added if the event succeeds and -1 if an error occurs.

Examples

This example is from the of_Populate function:

```
...
Return this.event pfc_AddAll(al_parent, lds_data)
...
```

pfc_AddItem

Description

Adds the specified row as the last item under the specified parent.

Syntax

instancename.EVENT **pfc_AddItem** (*parent*, *data*, *row*)

Argument	Description
instancename	Instance name of u_tvs
parent	Long specifying the handle of the TreeView item under which the new item is added. This argument is accessed through the *al_parent* argument
data	N_ds-based DataStore containing the row to be added. This argument is accessed through the *ads_source* argument
row	Long specifying the row containing the data to be added to the TreeView. This argument is accessed through the *al_row* argument

Return value

Long. Returns the handle of the new item if the event succeeds and -1 if an error occurs.

Examples

This example is from the of_AddItem function:

```
...
ll_newindex = this.event pfc_additem  &
   (al_parent, lds_datastore, ll_row)
...
```

pfc_DeleteItem

Description

Deletes the selected item from the TreeView and the DataStore.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Examples

This example calls the pfc_DeleteItem event:

```
...
li_return = tv_1.Event pfc_DeleteItem()
...
```

pfc_InsertItem

Description

Inserts a new item into the TreeView.

Syntax

instancename.EVENT **pfc_InsertItem** (*parent*, *datastore*, *row*, *position*, *item*)

Argument	Description
instancename	Instance name of u_tvs
parent	Long specifying the handle of the TreeView item under which the new item is added. This argument is accessed through the *al_parent* argument
datastore	N_ds containing the data for the new item. This can be either the DataStore maintained by n_cst_tvsrv_levelsource or another DataStore. If this is another DataStore, the associated DataWindow objects must be the same and the function also adds the row to the DataStore maintained by n_cst_tvsrv_levelsource. This argument is accessed through the *ads_source* argument
row	Long specifying the row containing the item to be inserted. This argument is accessed through the *al_row* argument
position	String specifying the position under the current parent in which to insert the new item Values are: ♦ First ♦ Last ♦ Sort ♦ After This argument is accessed through the *as_position* argument

Argument	Description
item	Long containing the handle to the item before the item to be inserted. The function uses this argument if you specify After for *position*. This argument is accessed through the *al_handle* argument

Return value

Integer. Returns the handle of the new item if the function succeeds and -1 if an error occurs.

Usage

This event calls the pfc_SetItemAttributes and pfc_PreInsertItem events before inserting the item.

Examples

This example is from the pfc_AddAll event:

```
...
For ll_row = 1 to ll_rowcount
   If this.event pfc_insertitem(al_parent,  &
      ads_source, ll_row, "last", 0) < 1 then
        Return -1
   End If
   ll_count++
...
```

pfc_NewItem

Description

Empty user event that you extend to add information to both the data source and the TreeView.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

You typically use this event to open a dialog box prompting the user for complete information, adding it to the data source and the TreeView (via the n_cst_tvsrv_levelsource of_InsertItem function) when it closes.

pfc_Populate

Description

Retrieves the data source and uses it to populate the TreeView as children of the specified parent.

Syntax

instancename.EVENT **pfc_Populate** (*parent*)

Argument	Description
instancename	Instance name of u_tvs

Argument	Description
parent	Long specifying the handle of the TreeView item under which the new items are added. This argument is accessed through the *al_parent* argument

Return value

Long. Returns the number of items added if the event succeeds and -1 if an error occurs.

Usage

You must also add code to the pfc_Retrieve event.

Examples

This example is from the ItemExpanding event:

```
...
If This.FindItem(ChildTreeItem!, handle) = -1 Then
   if this.GetItem(handle, ltvi_This) = 1 then
      li_rc = this.event pfc_Populate(handle)
      If li_rc < 1 Then
...
```

pfc_PostUpdate

Description

Calls the n_cst_luw of_PostUpdate function to perform post-update processing on the specified controls.

Syntax

instancename.EVENT **pfc_PostUpdate** (*controls*)

Argument	Description
instancename	Instance name of u_tvs
controls	PowerObject array containing the objects on which to perform post-update processing. This argument is accessed through the *apo_control* argument

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

Extend this event to perform additional post-update processing.

pfc_PreDeleteItem

Description

Empty user event to which you add logic to perform pre-delete processing.

Syntax

instancename.EVENT **pfc_PreDeleteItem** (*handle*)

Argument	Description
instancename	Instance name of u_tvs
handle	Long specifying the handle of the TreeView item to be deleted. This argument is accessed through the *al_handle* argument

Usage

The pfc_DeleteItem event calls this event.

pfc_PreInsertItem

Description

Empty user event to which you add logic to perform pre-insert processing.

Syntax

instancename.EVENT **pfc_PreInsertItem** (*parent*, *data*, *row*, *tvitem*)

Argument	Description
instancename	Instance name of u_tvs
parent	Long specifying the handle of the TreeView item under which the new item is added. This argument is accessed through the *al_parent* argument
data	N_ds containing the data for the new item. This argument is accessed through the *ads_obj* argument
row	Long specifying the row containing the new item. This argument is accessed through the *al_row* argument
tvitem	TreeViewItem to be inserted. This argument is accessed through the *atvi_item* argument (passed by reference)

Usage

The pfc_InsertItem event calls this event.

pfc_PreRefreshItem

Description

Empty user event to which you add logic that changes properties before a TreeView item is refreshed.

Syntax

instancename.EVENT **pfc_PreRefreshItem** (*handle*, *data*, *row*, *tvitem*)

Argument	Description
instancename	Instance name of u_tvs
handle	Long specifying the handle of the TreeView item to be refreshed. This argument is accessed through the *al_handle* argument

Argument	Description
data	N_ds containing the data for the refreshed item. This argument is accessed through the *ads_obj* argument
row	Long specifying the row containing the refreshed item. This argument is accessed through the *al_row* argument
tvitem	TreeViewItem to be refreshed. This argument is accessed through the *atvi_item* argument (passed by reference)

Usage

The pfc_RefreshItem event calls this event. The code you add here will likely be the same as code you add to pfc_PreInsertItem.

pfc_PreRefreshLevel

Description

Empty user event to which you add logic that executes before a TreeView level is refreshed.

Syntax

instancename.EVENT **pfc_PreRefreshLevel** (*level*)

Argument	Description
instancename	Instance name of u_tvs
level	Integer specifying the TreeView level to refresh. This argument is accessed through the *ai_level* argument

Usage

The pfc_RefreshLevel event calls this event.

pfc_PreRmbMenu

Description

Empty user event allowing you to modify m_tvs contents before display.

Syntax

instancename.EVENT **pfc_PreRmbMenu** (*editmenu*)

Argument	Description
instancename	Instance name of u_tvs
editmenu	M_tvs variable containing the popup menu to be displayed (passed by reference)

Usage

Optionally add logic to this event to selectively enable and disable m_lvs menu items.

pfc_Properties

Description

Empty user event to which you might add logic to display a Properties dialog.

Return value

Integer. Return 1 if the event succeeds and -1 if an error occurs.

pfc_RefreshItem

Description

Refreshes the specified TreeView item, resetting all properties to the defaults specified in the data source.

Syntax

instancename.EVENT **pfc_RefreshItem** (*handle*)

Argument	Description
instancename	Instance name of u_tvs
handle	Long specifying the handle of the TreeView item to refresh

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Examples

This example is from the pfc_RefreshLevel event:

```
...
For ll_row = 1 to ll_rowcount
   ll_handle = inv_levelsource.of_GetHandle  &
     (lds_source, ll_row, ai_level)
   If ll_handle < 1 Then Continue
   If this.event pfc_RefreshItem  &
     (ll_Handle) = -1 Then Return -1
Next
...
```

pfc_RefreshLevel

Description

Refreshes the specified TreeView level., resetting all properties to the defaults specified in the data source.

Syntax

instancename.EVENT **pfc_RefreshLevel** (*level*)

Argument	Description
instancename	Instance name of u_tvs
level	Integer specifying the TreeView level to refresh

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Examples

This example calls the pfc_RefreshLevel event:

```
...
tv_1.Event pfc_RefreshLevel(1)
...
```

pfc_RenameItem

Description

Renames the selected item.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

Internal.

pfc_Retrieve

Description

Retrieves rows into a data source.

Syntax

instancename.EVENT **pfc_Retrieve** (*parent*, *data*)

Argument	Description
instancename	Instance name of u_tvs
parent	Long specifying the handle of the TreeView item under which new items are retrieved. This argument is accessed through the *al_parent* argument
data	N_ds into which the event places the retrieved rows. This argument is accessed through the *ads_data* argument (passed by reference)

Return value

Long. Returns the number of items retrieved if the event succeeds and -1 if an error occurs.

Usage

Extend this event to perform retrieval.

Examples

This example shows code you might add to the pfc_Retrieve event:

```
Integer  li_level, li_count
Any  la_args[20]

li_level = this.of_GetNextLevel(al_parent)
IF li_level = 2 THEN
```

```
    li_count = this.inv_levelsource.of_GetArgs  &
      (al_parent, li_level, la_args)
END IF
Return of_Retrieve(al_parent, la_args, ads_data)
```

pfc_SearchCompare

Description

Compares TreeView data or item text with a target string.

Syntax

instancename.EVENT **pfc_SearchCompare** (*handle*, *attribute*, *target*, *respectcase*, *fullcompare*)

Argument	Description
instancename	Instance name of u_tvs
handle	Long containing the handle of the item being compared
attribute	String specifying the TreeView attribute to compare. Valid values are: ♦ Data ♦ Label
target	Any data type specifying the search target
respectcase	Boolean indicating whether the search is case-sensitive
fullcompare	Boolean indicating whether the specified attribute must match *target* exactly (TRUE) or the attribute can simply contain *target* (FALSE). This argument only applies when *target* is a string

Return value

Boolean. Returns TRUE if a match is found and FALSE if no match is found.

Usage

This event is called by the n_cst_tvsrv of_SearchChild function, which is called by of_FindItem.

If your application requires a more complex comparison, override this event.

pfc_SetItemAttributes

Description

Sets default properties for the TreeView item before insertion.

Syntax

instancename.EVENT **pfc_SetItemAttributes** (*parent*, *data*, *row*, *tvitem*)

Argument	Description
instancename	Instance name of u_tvs

Argument	Description
parent	Long specifying the handle of the TreeView item under which the new item is inserted. This argument is accessed through the *al_parent* argument
data	N_ds containing the data for the item. This argument is accessed through the *ads_obj* argument
row	Long specifying the row from which to populate the item. This argument is accessed through the *al_row* argument
tvitem	TreeViewItem to be inserted. This argument is accessed through the *atvi_item* argument (passed by reference)

Usage

Optionally extend this event to change TreeView item properties before insertion.

pfc_Undo

Description

Cancels the last edit, insert, or delete to the TreeView, restoring the text to the content before the last change.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

The message router calls this event when a TreeView based on u_tvs has focus and the user selects Edit>Undo.

pfc_Update

Description

Calls the n_cst_luw of_Update function to update the specified controls.

Syntax

instancename.EVENT **pfc_Update** (*controls*, *accepttext*, *resetflags*)

Argument	Description
instancename	Instance name of u_tvs
controls	PowerObject array containing the objects to update. This argument is accessed through the *apo_control* argument
accepttext	Boolean specifying whether n_cst_luw should automatically perform an AcceptText before performing the Update (TRUE) or not (FALSE)
resetflags	Boolean specifying whether n_cst_luw should automatically reset DataWindow update flags (TRUE) or not (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

The of_Update function calls this event.

pfc_UpdatePrep

Description

Empty user event to which you can add code that prepares for update.

Syntax

instancename.EVENT **pfc_UpdatePrep** (*controls*)

Argument	Description
instancename	Instance name of u_tvs
controls	PowerObject array containing the objects to update. This argument is accessed through the *apo_control* argument

Return value

Long. Return 1 if the update preparation succeeds and -1 to halt the update process.

Usage

The of_UpdatePrep function calls this function.

pfc_UpdatesPending

Description

Determines if there are pending updates for the TreeView data sources.

Syntax

instancename.EVENT **pfc_UpdatesPending** (*controls*, *pending*)

Argument	Description
instancename	Instance name of u_tvs
controls	PowerObject array containing the objects to update. This argument is accessed through the *apo_control* argument
pending	PowerObject array into which the event places objects with pending updates. This argument is accessed through the *apo_pending* argument (passed by reference)

Return value

Integer. Returns 1 if there are pending updates, 0 if there are no pending updates, and -1 if an error occurs.

Usage

This event is called by the of_GetUpdatesPending function.

pfc_Validation

Description

Checks for required fields.

Syntax

instancename.EVENT **pfc_Validation** (*controls*)

Argument	Description
instancename	Instance name of u_tvs
controls	PowerObject array containing the objects to validate. This argument is accessed through the *apo_control* argument

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is called by the of_Validation function.

RButtonUp

Description

Displays the m_tvs popup menu.

Usage

This event executes when the user releases the right mouse button.

RightClicked

Description

Tracks the clicked item.

Usage

This event executes when the user presses the right mouse button.

Functions

U_tvs includes precoded object functions:

of_AcceptText	of_Reset
of_AddItem	of_Retrieve
of_CanUndo	of_SetAlwaysValidate
of_GetInfo	of_SetBase
of_GetNextLevel	of_SetLevelSource
of_GetObjects	of_SetLogicalUnitOfWork
of_GetParentWindow	of_SetPrint
of_GetUpdateRequestor	of_SetRMBMenu
of_InsertItem	of_SetUpdateable
of_IsAlwaysValidate	of_SetUpdateRequestor

of_IsRmbMenu	of_Update
of_IsUpdateable	of_UpdatePrep
of_MessageBox	of_UpdatesPending
of_Populate	of_Validation
of_PostUpdate	

of_AcceptText

Description

Performs an AcceptText function for each level's data source.

Access

Public

Syntax

instancename.**of_AcceptText** (*focusonerror*)

Argument	Description
instancename	Instance name of u_tvs
focusonerror	Boolean indicating whether PFC sets focus to the first item in error when an error occurs

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize accept text processing, extend the pfc_AcceptText event.

Examples

This example is from the n_cst_luw of_AcceptText function:

```
...
If lb_defined Then
   li_rc =  &
      lpo_tocheck.Function Dynamic of_AcceptText &
         (ab_focusonerror)
   If li_rc < 0 Then Return -1

...
```

of_AddItem

Description

Adds a new item as the last item under the specified parent.

Access

Public

Syntax

instancename.**of_AddItem** (*parent, rowinfo*)

Argument	Description
instancename	Instance name of u_tvs
parent	Long specifying the handle of the TreeView item under which the new item is inserted
rowinfo	Any array containing data for the row

Return value

Long. Returns the index of the added item if the function succeeds and -1 if an error occurs.

Usage

If the information in *rowinfo* is not already in the data source, this function adds the new row to the data source. If you are using the TreeView level source service, the data types of the elements in *rowinfo* must match those in the DataWindow object specified in the n_cst_tvsrv_levelsource of_Register function.

Examples

This example is from the of_Refresh function:

```
Long  ll_return
Long  ll_handle
Any  la_row[ ]
TreeViewItem  ltvi_item

ll_handle = tv_1.FindItem(CurrentTreeItem! , 0)
tv_1.GetItem(ll_handle, ltvi_item)
IF ltvi_item.Level = 1 THEN
    la_row[1] = 1100
    la_row[2] = "Department of Defense"
    ll_return = tv_1.of_AddItem(ll_handle, la_row)
    tv_1.Event pfc_RefreshLevel(ltvi_item.Level)
END IF
```

of_CanUndo

Description

Reports whether the TreeView can undo the last edit, insertion, or deletion, returning the operation type.

Access

Public

Syntax

instancename.**of_CanUndo** (*undotype*)

Argument	Description
instancename	Instance name of u_tvs

Argument	Description
undotype	String into which the function places the type of the last operation (passed by reference): ♦ **UNDO_EDIT or "Edit"** Edit ♦ **UNDO_INSERT or "Insert"** Insertion ♦ **UNDO_DELETE or "Delete"** Deletion

Return value

Boolean. Returns TRUE if the last operation can be undone and FALSE if it cannot.

Examples

This example calls the of_CanUndo function:

```
String  ls_undotype

IF tv_1.of_CanUndo(ls_undotype) THEN
   MessageBox("TreeView", "Can undo")
ELSE
   MessageBox("TreeView", "Cannot undo")
END IF
```

of_GetInfo

Description

Retrieves object information.

Access

Public

Syntax

instancename.**of_GetInfo** (*infoobject*)

Argument	Description
instancename	Instance name of u_tvs
infoobject	N_cst_infoattrib instance into which the function places information (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_GetInfo function:

```
n_cst_infoattrib  lnv_info

tv_1.of_GetInfo(lnv_info)
MessageBox("Info",  &
  "Description: " + lnv_info.is_description  &
```

```
        + ". Name: " + lnv_info.is_name)
```

of_GetNextLevel

Description

Determines the DataStore level below the passed parent.

Access

Public

Syntax

instancename.**of_GetNextLevel** (*parent*)

Argument	Description
instancename	Instance name of u_tvs
parent	Long specifying the handle of the TreeView item for which the next lowest level is returned

Return value

Long. Returns the number of the level below *parent* if the function succeeds and -1 if an error occurs.

Examples

This example is from the pfc_PreRetrieve event:

```
...
If isvalid(inv_levelsource) then
   li_level = of_GetNextLevel(al_parent)
   If li_level < 1 then Return
...
```

of_GetObjects

Description

Retrieves the objects to be updated.

Access

Protected

Syntax

instancename.**of_GetObjects** (*objects*)

Argument	Description
instancename	Instance name of u_tvs
objects	PowerObject array into which the function places objects to be updated (passed by reference)

Return value

Integer. Returns the number of elements in the *objects* array if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the of_AcceptText function:

```
...
Else
   this.of_Getobjects(lpo_updatearray)
End If
...
```

of_GetParentWindow

Description

Returns the parent of the current window.

Access

Public

Syntax

instancename.**of_GetParentWindow** (*parent*)

Argument	Description
instancename	Instance name of u_tvs
parent	Window variable into which the function places the parent of the current window (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the GetFocus event.

```
Window  lw_parent

IF gnv_app.of_GetMicrohelp() THEN
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
      lw_parent.Dynamic Event &
         pfc_ControlGotFocus (this)
   END IF
END IF
```

of_GetUpdateRequestor

Description

Retrieves a reference to the object requesting an update.

Access

Public

Syntax

instancename.**of_GetUpdateRequestor** (*requestor*)

Argument	Description
instancename	Instance name of u_tvs
requestor	PowerObject into which the function places a reference to the object requesting the update (passed by reference)

Return value

Integer. Returns 1 if the function succeeds, 0 if there is no update requestor and -1 if an error occurs.

Usage

Call this function if you are extending the pfc_Save process and need to access the update requestor.

of_InsertItem

Inserts an item into the TreeView. There are two syntaxes:

To	Use
Insert from a DataStore	Syntax 1
Insert from an array	Syntax 2

Syntax 1

Insert an item from a DataStore

Description

Adds a new item to the TreeView using a row from a DataStore.

Access

Public

Syntax

instancename.**of_InsertItem** (*parent, datastore, row* {, *position, handle* })

Argument	Description
instancename	Instance name of u_tvs
parent	Long specifying the handle of the TreeView item under which the new item is inserted
datastore	N_ds containing the data for the new item. This can be either the DataStore maintained by n_cst_tvsrv_levelsource or another DataStore. If this is another DataStore, the associated DataWindow objects must be the same and the function also adds the row to the DataStore maintained by n_cst_tvsrv_levelsource (passed by reference)
row	Long specifying the row from which to populate the new item

Argument	Description
position (optional)	String specifying the position under the current parent in which to insert the new item Values are: ♦ First ♦ Last (default) ♦ Sort ♦ After
handle (optional)	Long containing the handle of the TreeView item after which the item is added. The function uses this argument if you specify After for *position*

Return value

Long. Returns the handle of the new item if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_InsertItem function:

```
Long  ll_row, ll_handle
TreeViewItem  ltvi_item

ll_handle = tv_1.FindItem(CurrentTreeItem! , 0)
tv_1.GetItem(ll_handle, ltvi_item)
ll_row = ids_datastore.GetRow()
IF ltvi_item.Level = 1 THEN
   ll_return = tv_1.of_InsertItem  &
      (ll_handle, ids_data, ll_row, "Sorted", 0)
END IF
```

Syntax 2

Insert an item from an array

Description

Adds a new item to the TreeView using a value from an array.

Access

Public

Syntax

instancename.**of_InsertItem** (*parent*, *colvalues* {, *position*, *handle* })

Argument	Description
instancename	Instance name of u_tvs
parent	Long specifying the handle of the TreeView item under which the new item is inserted
colvalues	Any array containing data for the TreeView

Argument	Description
position (optional)	String specifying the position under the current parent in which to insert the new item Values are: ♦ First ♦ Last (default) ♦ Before ♦ After
handle (optional)	Long containing the handle of the TreeView item after which the item is added. The function uses this argument if you specify After for *position*

Return value

Integer. Returns the index of the new item if the function succeeds and -1 if an error occurs.

Usage

If you are using the TreeView level source service, the data types of the elements in *colvalues* must match those in the DataWindow object specified in the n_cst_tvsrv_levelsource of_Register function.

Examples

This example calls the of_InsertItem function:

```
Any  la_row[ ]
Long  ll_handle
TreeViewItem ltvi_item

ll_handle = tv_1.FindItem(CurrentTreeItem! , 0)
la_row[1] = 1200
la_row[2] = "Commerce"
tv_1.of_InsertItem(ll_handle, la_row, "First", 0)
```

of_IsAlwaysValidate

Description

Reports whether the default save process always performs validation.

Access

Public

Syntax

instancename.**of_IsAlwaysValidate ( )**

Argument	Description
instancename	Instance name of u_tvs

Return value

Boolean. Returns TRUE if the default save process always performs validation and FALSE if it does not.

Examples

This example calls the of_IsAlwaysValidate function:

```
IF tv_1.of_IsAlwaysValidate() = TRUE THEN
   MessageBox("TV", "Always validate")
ELSE
   MessageBox("TV", "Sometimes validate")
END IF
```

of_IsRmbMenu

Description

Reports whether the popup menu is enabled.

Access

Public

Syntax

instancename.**of_IsRmbMenu** ( )

Argument	Description
instancename	Instance name of u_tvs

Return value

Boolean. Returns TRUE if the m_tvs popup menu is enabled and FALSE if it is not.

Examples

This example calls the of_IsRmbMenu function:

```
IF tv_1.of_IsRmbMenu() = TRUE THEN
   MessageBox("TV", "RMB enabled")
ELSE
   MessageBox("TV", "RMB disabled")
END IF
```

of_IsUpdateable

Description

Reports whether the TreeView's data source is updatable.

Access

Public

Syntax

instancename.**of_IsUpdateable** ( )

Argument	Description
instancename	Instance name of u_tvs

Return value

Boolean. Returns TRUE if the TreeView's data source is updatable and FALSE if it is not.

Usage

The pfc_UpdatesPending event calls this function.

Examples

This example is from the pfc_UpdatesPending event:

```
...
If Not of_IsUpdateable() Then Return NO_UPDATESPENDING
...
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of u_tvs
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!

Argument	Description
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in TreeViews.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('tv_error','Save', &
   as_error, StopSign!, Ok!, 1)
```

of_Populate

Description

Populates a TreeView item with its child items.

Access

Public

Syntax

instancename.**of_Populate** (*parent*)

Argument	Description
instancename	Instance name of u_tvs
parent	Long specifying the handle of the handle of the TreeView item to be populated

Return value

Integer. Returns the number of items added if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

The example is from the pfc_Populate event:

```
IF (al_parent < 0) or IsNull(al_parent) then Return -1
Return of_populate(al_parent)
```

of_PostUpdate

Description

Calls the pfc_PostUpdate event, which clears update flags and allows you to code additional post-update processing.

Access

Public

Syntax

instancename.**of_PostUpdate** ( )

Argument	Description
instancename	Instance name of u_tvs

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

To customize post update processing, extend the pfc_PostUpdate event.

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize post-update processing, extend the pfc_PostUpdate event.

Examples

This example is from the n_cst_luw of_PostUpdate function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_PostUpdate()
...
```

of_Reset

Description

Deletes all items from both the TreeView and the data source.

Access

Public

Syntax

instancename.**of_Reset** ( )

Argument	Description
instancename	Instance name of u_tvs

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Examples

This example calls the of_Reset function:

```
tv_1.of_Reset()
```

of_Retrieve

Description

Retrieves rows into a DataStore.

Syntax

instancename.EVENT **of_Retrieve** (*parent*, *args*, *data*)

Argument	Description
instancename	Instance name of u_tvs
parent	Long specifying the handle of the TreeView item under which new items are retrieved
args	20-element Any array specifying retrieval arguments
data	N_ds into which the function places the retrieved rows (passed by reference)

Return value

Long. Returns the number of items retrieved if the function succeeds, 0 if no action was taken, and -1 if an error occurs.

Examples

This example (from a pfc_Retrieve event) calls the of_Retrieve function:

```
Integer  li_level, li_count
Any  la_args[20]

li_level = this.of_GetNextLevel(al_parent)
IF li_level = 2 THEN
   li_count = this.inv_levelsource.of_GetArgs  &
     (al_parent, li_level, la_args)
END IF
Return of_Retrieve(al_parent, la_args, ads_data)
```

of_SetAlwaysValidate

Description

Specifies whether the default save process always performs validation.

Access

Public

Syntax

instancename.**of_SetAlwaysValidate** (*boolean*)

Argument	Description
instancename	Instance name of u_tvs
boolean	Boolean specifying whether the default save process always perform validation (TRUE) or only performs validation if a control has pending updates (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetAlwaysValidate function:

```
tv_1.of_SetAlwaysValidate(TRUE)
```

of_SetBase

Description

Enables or disables n_cst_tvsrv, which provides basic TreeView services.

Access

Public

Syntax

instancename.**of_SetBase** (*boolean*)

Argument	Description
instancename	Instance name of u_tvs
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) basic TreeView services

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_tvsrv. This instance is named inv_base.

Because all TreeView services are descendants of n_cst_tvsrv (and have n_cst_tvsrv functions available to them), use this object when you require basic TreeView services only.

Examples

This example calls the of_SetBase function to enable basic TreeView services:

```
tv_1.of_SetBase(TRUE)
```

of_SetLevelSource

Description

Enables or disables n_cst_tvsrv_levelsource, which provides data access for TreeView levels.

Access

Public

Syntax

instancename.**of_SetLevelSource** (*boolean*)

Argument	Description
instancename	Instance name of u_tvs
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) n_cst_tvsrv_levelsource

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_tvsrv_levelsource. This instance is named inv_levelsource.

Examples

This example calls the of_SetLevelSource function:

```
tv_1.of_SetLevelSource(TRUE)
```

of_SetLogicalUnitOfWork

Description

Enables or disables n_cst_luw, which provides the logical unit of work service.

Access

Protected

Syntax

instancename.**of_SetLogicalUnitOfWork** (*boolean*)

Argument	Description
instancename	Instance name of u_tvs
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) n_cst_luw

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_luw. This instance is named inv_luw. If you do not enable n_cst_luw, u_tvs enables it automatically.

Examples

This example calls the of_SetLogicalUnitOfWork function:

```
tv_1.of_SetLogicalUnitOfWork(TRUE)
```

of_SetPrint

Description

Enables or disables n_cst_tvsrv_print, which provides the TreeView print service.

Access

Public

Syntax

instancename.**of_SetPrint** (*boolean*)

Argument	Description
instancename	Instance name of u_tvs

Argument	Description
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the TreeView print service

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_tvsrv_print. This instance is named inv_print.

Examples

This example calls the of_SetPrint function:

```
tv_1.of_SetPrint(TRUE)
```

of_SetRMBMenu

Description

Enables or disables right mouse button support for m_tvs, the TreeView popup menu.

Access

Public

Syntax

instancename.**of_SetRMBMenu** (*boolean*)

Argument	Description
instancename	Instance name of u_tvs
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) right mouse button support

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetRMBMenu function:

```
tv_1.of_SetRMBMenu(TRUE)
```

of_SetUpdateable

Description

Specifies whether the TreeView is updatable and should be included in the default save process.

Access

Public

Syntax

instancename.**of_SetUpdateable** (*boolean*)

Argument	Description
instancename	Instance name of u_tvs
boolean	Boolean indicating whether the TreeView is updatable. All updatable TreeViews are included in the default save processing

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function to enable default save processing for TreeViews that are updatable (by default, TreeViews are not updatable).

Examples

This example calls the of_SetUpdateable function:

```
tv_1.of_SetUpdateable(FALSE)
```

of_SetUpdateRequestor

Description

Creates a reference to the object requesting an update within a logical unit of work.

Access

Public

Syntax

instancename.**of_SetUpdateRequestor** (*requestor*)

Argument	Description
instancename	Instance name of u_tvs
requestor	PowerObject containing the object requesting the update

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the of_Update function:

```
...
If IsValid(inv_luw) Then
   inv_luw.of_SetUpdateRequestor(apo_requestor)
End If
...
```

of_Update

Description

Saves all rows in the DataStores associated with the TreeView.

Access

Public

Syntax

instancename.**of_Update** (*accept, resetflag* {, *requestor* })

Argument	Description
instancename	Instance name of u_tvs
accept	Boolean indicating whether the Update function performs an AcceptText before saving rows to the database
resetflag	Boolean indicating whether the Update function resets the update flags
requestor	PowerObject containing the requestor object

Return value

Integer. Returns 1 if the update succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize update processing, extend the pfc_Update event.

Examples

This example is from the n_cst_luw of_Update function:

```
...
If lb_defined Then
  li_rc = lpo_tocheck.Function Dynamic of_Update &
    (ab_accepttext,  ab_resetflag, &
      lpo_updaterequestor)
  If li_rc < 0 Then Return -1
  Continue
End If
...
```

of_UpdatePrep

Description

Calls the pfc_UpdatePrep event, which allows you to code additional update preparation logic.

Access

Public

Syntax

instancename.**of_UpdatePrep** ( )

Argument	Description
instancename	Instance name of u_tvs

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize update preparation processing, extend the pfc_UpdatePrep event.

Examples

This example is from the n_cst_luw of_UpdatePrep function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_UpdatePrep ()
  If li_rc < 0 Then Return -1
  Continue
End If
...
```

of_UpdatesPending

Description

Determines if there are updates pending in the DataStores associated with the TreeView.

Access

Public

Syntax

instancename.**of_UpdatesPending ( )**

Argument	Description
instancename	Instance name of u_tvs

Return value

Integer. Returns values as follows:

- **1** Updates are pending
- **0** No updates pending
- **-1** An error occurred

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize pending updates processing, extend the pfc_UpdatesPending event.

Examples

This example calls the of_UpdatesPending function:

```
...
If lb_defined Then
   la_rc = lpo_tocheck.Dynamic of_UpdatesPending ()
...
```

of_Validation

Description

Performs validation on the TreeView data sources.

Access

Public

Syntax

instancename.**of_Validation ( )**

Argument	Description
instancename	Instance name of u_tvs

Return value

Integer. Returns 1 if the function succeeds and -1 if a validation error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize validation processing, extend the pfc_Validation event.

Examples

This example calls the of_Validation function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_Validation()
...
```

u_vsb

Description

VerticalScrollBar visual user object ancestor.

Ancestry

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

PFC visual user objects are designed to be used with windows that are descendants of w_master.

Usage

Use this visual object in windows instead of the standard PowerBuilder VerticalScrollBar control.

U_vsb event scripts provide integration with PFC menus.

See also

u_hsb

Events

U_vsb includes one precoded event script:

GetFocus

GetFocus

Description

Updates the parent window so it can set MicroHelp.

Usage

This event calls the pfc_ControlGotFocus event in the parent.

Functions

U_vsb includes precoded object functions:

of_GetParentWindow
of_MessageBox

of_GetParentWindow

Description

Retrieves a reference to the parent window.

Access

Public

Syntax

instancename.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_vsb
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if there is no parent window. If no parent window is found, *window* returns NULL.

Usage

The u_vsb GetFocus event calls this function.

Examples

This example is from the u_vsb GetFocus event:

```
Window  lw_parent

IF gnv_app.of_GetMicrohelp() THEN
   of_GetParentWindow(lw_parent)
   IF IsValid(lw_parent) THEN
      lw_parent.Dynamic Event &
         pfc_ControlGotFocus (this)
   END IF
END IF
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of u_vsb
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in VerticalScrollBar controls.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('vsb_error','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

CHAPTER 7

Custom Visual User Objects

About this chapter

This chapter describes the custom visual user objects in PFC.

Contents

The custom visual user objects are listed in alphabetical order. Each object's discussion includes alphabetical listings of instance variables, events, and object functions.

u_base

Description

Base object for custom visual user objects (such as u_calculator and u_calendar).

U_base is a self-updating object.

Ancestry

pfc_u_base
u_base

Events

Destructor	pfc_UpdatePrep
pfc_AcceptText	pfc_UpdatesPending
pfc_PostUpdate	pfc_Validation
pfc_Update	Resize

Object functions

Public	of_AcceptText	Public	of_SetLogicalUnitOfWork
Public	of_GetInfo	Public	of_SetResize
Public	of_GetParentWindow	Protected	of_SetUpdateable
Protected	of_GetUpdateObjects	Protected	of_SetUpdateObjects
Protected	of_IsAlwaysValidate	Protected	of_SetUpdateRequestor
Protected	of_IsUpdateable	Public	of_Update
Protected	of_MessageBox	Public	of_UpdatePrep
Public	of_PostUpdate	Public	of_UpdatesPending
Protected	of_SetAlwaysValidate	Public	of_Validation

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

n_cst_infoattrib
n_cst_luw
n_cst_resize

Usage

Use this object as the base object for custom visual user objects. U_base contains all the events and functions necessary for a PFC self updating object.

FOR INFO For more on self updating objects see n_cst_luw on page 1139.

To use u_base:

1 Use the User Object painter to create a descendant of u_base.

2 Add controls, events, and functions as necessary.

Descendants

u_calculator
u_calendar
u_progressbar
u_tabpg

See also

u_base

Instance variables

U_base includes instance variables:

Instance variable	Description	Data type	Access	Usage
CONTINUE_ACTION	Constant set to 1	Integer	Public	Internal
FAILURE	Constant set to -1	Integer	Public	Internal
ib_alwaysvalidate	Controls whether the save process includes all objects in the validation process	Boolean	Protected	Set with of_SetAlwaysValidate (default is FALSE)
ib_isupdateable	Indicates whether the object can be updated	Boolean	Protected	Set with of_SetUpdateable (default is FALSE)
inv_luw	Reference variable for logical unit of work service	n_cst_luw	Protected	Set with of_SetLogicalUnitOfWork
inv_resize	Reference variable for resize service	n_cst_resize	Public	Set with of_SetResize
ipo_pendingupdates[ ]	Default list of objects to be updated	PowerObject	Protected	Internal
ipo_updateobjects[ ]	Customized list of objects to be updated	PowerObject	Protected	Internal
ipo_tempupdateobjects[ ]	One-time list of objects to be updated	PowerObject	Protected	Internal
ipo_updaterequestor	Owner of the save process	PowerObject	Protected	Internal
NO_ACTION	Constant set to 0	Integer	Public	Internal
PREVENT_ACTION	Constant set to 0	Integer	Public	Internal
SUCCESS	Constant set to 1	Integer	Public	Internal

Events

U_base includes precoded events:

Destructor	pfc_UpdatePrep
pfc_AcceptText	pfc_UpdatesPending

pfc_PostUpdate	pfc_Validation
pfc_Update	Resize

Destructor

Description

Destroys all enabled service objects.

Usage

This event executes when the control is destroyed or the window closes.

pfc_AcceptText

Description

Calls the n_cst_luw of_AcceptText function, which in turn calls the of_AcceptText function for all controls that implement it.

Syntax

instancename.EVENT **pfc_AcceptText** (*controlarray, focusonerror*)

Argument	Description
instancename	Instance name of the u_base descendant
controlarray	PowerObject array containing the controls for which to accept text. From within the pfc_AcceptText event, access this value through the *apo_control* argument
focusonerror	Boolean specifying whether to set focus on a DataWindow column with errors. From within the pfc_AcceptText event, access this value through the *ab_focusonerror* argument

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

The of_AcceptText function calls this event.

pfc_PostUpdate

Description

Calls the n_cst_luw of_PostUpdate function, which in turn calls the of_PostUpdate function for all controls that implement it.

Syntax

instancename.EVENT **pfc_PostUpdate** (*controlarray*)

Argument	Description
instancename	Instance name of the u_base descendant

Argument	Description
controlarray	PowerObject array containing the controls for which to perform post update processing. From within the pfc_AcceptText event, access this value through the *apo_control* argument

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

The of_PostUpdate function calls this event.

You can extend this event to perform additional post-update processing.

pfc_Update

Description

Calls the n_cst_luw of_Update function, which in turn calls the of_Update function for all controls that implement it.

Syntax

instancename.EVENT **pfc_Update** (*controlarray*)

Argument	Description
instancename	Instance name of the u_base descendant
controlarray	PowerObject array containing the controls to be updated. From within the pfc_Update event, access this value through the *apo_control* argument

Return value

Integer. Returns 1 if the event succeeds and -1 if one or more update errors occur.

Usage

The of_Update function calls this event.

pfc_UpdatePrep

Description

Empty user event to which you can add code that prepares for update.

Syntax

instancename.EVENT **pfc_UpdatePrep** (*controls*)

Argument	Description
instancename	Instance name of the u_base descendant
controls	PowerObject array containing the objects to update. This argument is accessed through the *apo_control* argument

Return value

Long. Return 1 if the update preparation succeeds and -1 to halt the update process.

Usage

The of_UpdatePrep function calls this event.

pfc_UpdatesPending

Description

Calls the n_cst_luw of_UpdatesPending function, which in turn calls the of_UpdatesPending function for all controls that implement it.

Syntax

instancename.EVENT **pfc_UpdatesPending** (*controlarray*)

Argument	Description
instancename	Instance name of the u_base descendant
controlarray	PowerObject array containing controls to be tested for pending updates

Return value

Integer. Returns values as follows:

1 Pending updates were found
0 No pending updates
-1 AcceptText failed

Usage

The of_UpdatesPending function calls this event.

pfc_Validation

Description

Calls the n_cst_luw of_Validation function, which in turn calls the of_Validation function for all controls that implement it.

Syntax

instancename.EVENT **pfc_Validation** (*controlarray*)

Argument	Description
instancename	Instance name of the u_base descendant
controlarray	PowerObject control array containing controls to be validate

Return value

Integer. Returns 1 if there are no validation errors and -1 if a validation error occurs.

Usage

The of_Validation function calls this event.

Resize

Description

Triggers resize processing, if enabled for the object.

Usage

This event executes when the user resizes the window and the custom visual user object instance has been registered with the window or tab on which it is placed.

To enable resize processing, you must enabled the resize service for the u_base descendant instance and for the window or tab containing the instance.

Functions

U_base contains precoded object functions:

of_AcceptText	of_SetLogicalUnitOfWork
of_GetInfo	of_SetResize
of_GetParentWindow	of_SetUpdateable
of_GetUpdateObjects	of_SetUpdateObjects
of_IsAlwaysValidate	of_SetUpdateRequestor
of_IsUpdateable	of_Update
of_MessageBox	of_UpdatePrep
of_PostUpdate	of_UpdatesPending
of_SetAlwaysValidate	of_Validation

of_AcceptText

Description

Calls the pfc_AcceptText event.

Access

Public

Syntax

instancename.**of_AcceptText** (*focusonerror*)

Argument	Description
instancename	Instance name of the u_base descendant
focusonerror	Boolean indicating whether PFC sets focus to the first item in error when an error occurs

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize accept text processing, extend the pfc_AcceptText event.

Examples

This example is from the n_cst_luw of_AcceptText function:

```
...
If lb_defined Then
   li_rc =  &
      lpo_tocheck.Function Dynamic of_AcceptText &
         (ab_focusonerror)
   If li_rc < 0 Then Return -1

...
```

of_GetInfo

Description

Retrieves object information.

Access

Public

Syntax

instancename.**of_GetInfo** (*infoobject*)

Argument	Description
instancename	Instance name of the u_base descendant
infoobject	N_cst_infoattrib instance into which the function places information (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_GetInfo function:

```
n_cst_infoattrib  lnv_info

w_emplist.of_GetInfo(lnv_info)
MessageBox("Info",  &
  "Description: " + lnv_info.is_description  &
    + ". Name: " + lnv_info.is_name)
```

of_GetParentWindow

Description

Retrieves a reference to the parent window.

Access

Public

Syntax

instancename.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of u_base
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value

Integer. Returns 1 if the function succeeds, -1 if there is no parent window, and NULL if no parent window is found.

Examples

This example calls the of_GetParentWindow function:

```
Integer  li_return
Window  lw_window
...
li_return = uo_calc.of_GetParentWindow(lw_window)
...
```

of_GetUpdateObjects

Description

Retrieves the current default array of objects affected by the update process.

Access

Protected

Syntax

instancename.**of_GetUpdateObjects** (*objects*)

Argument	Description
instancename	Instance name of the u_base descendant
objects	PowerObject array into which the function places objects to be updated (passed by reference)

Return value

Integer. Returns the number of elements in the *objects* array if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_GetUpdateObjects function:

```
PowerObject  lpo_objs[ ]
Integer  li_return, li_count

li_return = this.of_GetUpdateObjects(lpo_objs)
FOR li_count = 1 to li_return
   IF lpo_objs[li_count] = ids_data THEN
```

```
            Return 1
        END IF
    NEXT
    li_return++
    lpo_objs[li_return] = ids_data
    Return this.of_SetUpdateObjects(lpo_objs)
```

of_IsAlwaysValidate

Description: Reports whether the default save process always performs validation.

Access: Protected

Syntax: *instancename*.**of_IsAlwaysValidate ( )**

Argument	Description
instancename	Instance name of the u_base descendant

Return value: Boolean. Returns TRUE if the default save process always performs validation and FALSE if it does not.

Examples: This example calls the of_IsAlwaysValidate function:

```
IF this.of_IsAlwaysValidate() = TRUE THEN
   MessageBox("Base", "Always validate")
ELSE
   MessageBox("Base", "Sometimes validate")
END IF
```

of_IsUpdateable

Description: Reports whether the object is updatable.

Access: Protected

Syntax: *instancename*.**of_IsUpdateable ( )**

Argument	Description
instancename	Instance name of the u_base descendant

Return value: Boolean. Returns TRUE if the window is updatable and FALSE if it is not.

Usage: Internal.

Examples

This example is from the pfc_UpdatesPendingRef event:

```
...
If Not of_IsUpdateable() Then Return NO_UPDATESPENDING
...
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of the u_base descendant
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in custom visual user objects.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('cvuo_error','Save', &
   as_error, StopSign!, Ok!, 1)
```

of_PostUpdate

Description

Calls the pfc_PostUpdate event.

Access

Public

Syntax

instancename.**of_PostUpdate** ( )

Argument	Description
instancename	Instance name of the u_base descendant

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize post-update processing, extend the pfc_PostUpdate event.

Examples

This example is from the n_cst_luw of_PostUpdate function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_PostUpdate()
...
```

of_SetAlwaysValidate

Description

Specifies whether the default save process always performs validation.

Access

Protected

Syntax

instancename.**of_SetAlwaysValidate** (*boolean*)

Argument	Description
instancename	Instance name of the u_base descendant
boolean	Boolean specifying whether the default save process always performs validation (TRUE) or only performs validation if a control has pending updates (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetAlwaysValidate function:

```
this.of_SetAlwaysValidate(TRUE)
```

of_SetLogicalUnitOfWork

Description

Enables or disables n_cst_luw, which provides the logical unit of work service.

Access

Public

Syntax

instancename.**of_SetLogicalUnitOfWork** (*boolean*)

Argument	Description
instancename	Instance name of the u_base descendant
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) n_cst_luw

Return value

Integer. Returns 1 if the function succeeds, 0 if the service is already enabled, and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_luw. This instance is named inv_luw. If you do not enable n_cst_luw, u_base enables it automatically.

Examples

This example calls the of_SetLogicalUnitOfWork function:

```
this.of_SetLogicalUnitOfWork(TRUE)
```

of_SetResize

Description

Enables or disables n_cst_resize (the resize service).

Access

Public

Syntax

instancename.**of_SetResize** (*boolean*)

Argument	Description
instancename	Instance name of u_base
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) an instance of n_cst_resize

Return value

Integer. Returns 1 if the function succeeds, 0 if the resize service is already enabled, and -1 if an error occurs.

Usage

After calling this function, apply n_cst_resize functions as necessary to the controls contained in the u_base descendant.

Examples

This example calls the of_SetResize function:

```
this.of_SetResize(TRUE)
```

of_SetUpdateable

Description

Specifies whether the object is updatable.

Access

Protected

Syntax

instancename.**of_SetUpdateable** (*boolean*)

Argument	Description
instancename	Instance name of the u_base descendant
boolean	Boolean indicating whether the window is updatable

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function to enable default save processing. By default, u_base is not updatable.

Examples

This example calls the of_SetUpdateable function:

```
this.of_SetUpdateable(TRUE)
```

of_SetUpdateObjects

Description

Sets a new default array containing objects for which updates are attempted.

Access

Protected

Syntax

instancename.**of_SetUpdateObjects** (*requestor*)

Argument	Description
instancename	Instance name of u_dw
requestor	PowerObject array containing the object to be updated

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function to customize the objects updated by the save process. You can even add other windows to the save process.

Examples

This example calls the of_SetUpdateObjects function:

```
PowerObject  lpo_objs[ ]
Integer  li_count

lpo_objs = this.control
li_count = UpperBound(lpo_objs)
li_count++
// Update w_other as well as this object
lpo_objs[li_count] = w_other
Return this.of_SetUpdateObjects(lpo_objs)
```

of_SetUpdateRequestor

Description

Creates a reference to the object requesting an update within a logical unit of work.

Access

Protected

Syntax

instancename.**of_SetUpdateRequestor** (*requestor*)

Argument	Description
instancename	Instance name of the u_base descendant
requestor	PowerObject containing the object requesting the update

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

of_Update

Description

Calls the pfc_Update event.

Access

Public

Syntax

instancename.**of_Update** (*accept*, *resetflag* {, *requestor*})

Argument	Description
instancename	Instance name of the u_base descendant
accept	Boolean indicating whether the Update function performs an AcceptText before saving rows to the database
resetflag	Boolean indicating whether the Update function resets the update flags
requestor (optional)	PowerObject containing the requestor object

Return value

Integer. Returns 1 if the update succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize update processing, extend the pfc_Update event.

Examples

This example is from the n_cst_luw of_Update function:

```
...
If lb_defined Then
  li_rc = lpo_tocheck.Function Dynamic of_Update &
    (ab_accepttext,  ab_resetflag, &
      lpo_updaterequestor)
  If li_rc < 0 Then Return -1
  Continue
End If
...
```

of_UpdatePrep

Description

Calls the pfc_UpdatePrep event, which allows you to code additional update preparation logic.

Access

Public

Syntax

instancename.**of_UpdatePrep** ( )

Argument	Description
instancename	Instance name of the u_base descendant

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize update preparation processing, extend the pfc_UpdatePrep event.

Examples

This example is from the n_cst_luw of_UpdatePrep function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_UpdatePrep()
  If li_rc < 0 Then Return -1
  Continue
End If
...
```

of_UpdatesPending

Description

Calls the pfc_UpdatesPending event.

Access

Public

Syntax

instancename.**of_UpdatesPending ( )**

Argument	Description
instancename	Instance name of the u_base descendant

Return value

Integer. Returns values as follows:

- **1** Updates are pending
- **0** No updates pending
- **-1** An error occurred

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize pending updates processing, extend the pfc_UpdatesPending event.

Examples

This example calls the of_UpdatesPending function:

```
...
If lb_defined Then
   la_rc = lpo_tocheck.Dynamic of_UpdatesPending()
...
```

of_Validation

Description

Calls the pfc_Validation event.

Access

Public

Syntax

instancename.**of_Validation** ( )

Argument	Description
instancename	Instance name of the u_base descendant

Return value

Integer. Returns 1 if the function succeeds and -1 if a validation error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize validation processing, extend the pfc_Validation event.

Examples

This example calls the of_Validation function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_Validation()
...
```

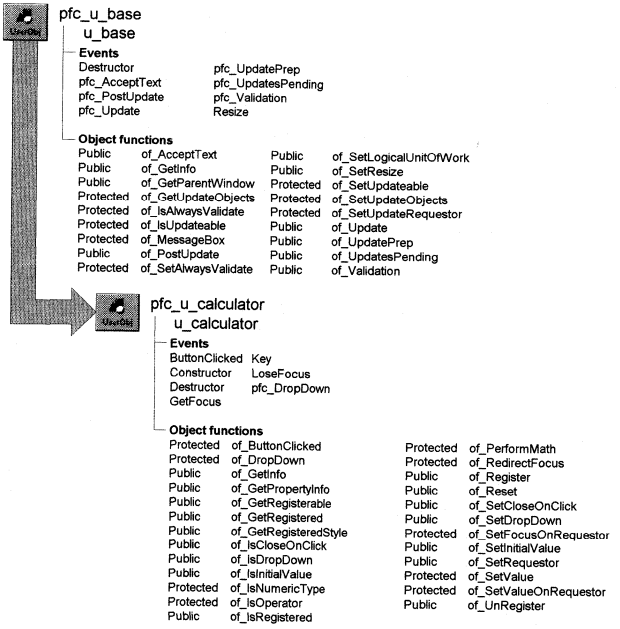
u_calculator

Description

Calculator object for use with numeric values in DataWindow and EditMask controls. Users use the calculator to enter values and calculations, which the object displays in the associated DataWindow column or EditMask control.

You typically use the calculator object as a dropdown object, displaying when a DataWindow column gets focus or when the user clicks the dropdown arrow. However, you can also place the calculator object directly onto a window.

Ancestry



Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Object relationships

n_cst_dropdown
n_cst_calculatorattrib
n_cst_infoattrib
n_cst_propertyattrib
u_dw
u_em

Usage

Use this object with EditMask controls that display numeric or decimal values or with DataWindow columns that display numeric values. For automatic calculator display, DataWindow columns must use the DropDownListBox edit style. For manual calculator display, DataWindow columns can use the DropDownListBox, Edit, or EditMask edit style.

To use this object for DataWindow columns:

1 Determine which columns in your DataWindow object are appropriate for a dropdown calculator. Use the DataWindow painter to assign the DropDownListBox, Edit, or EditMask edit style to these columns.

2 In the Window painter, define a DataWindow control based on u_dw.

3 Enable the dropdown calculator for the DataWindow by calling the of_SetDropDownCalculator function. This example is from the DataWindow control's Constructor event:

```
this.of_SetDropDownCalculator(TRUE)
```

4 Register one or more columns by calling the of_Register function:

```
this.iuo_calculator.of_Register("salary", &
  this.iuo_calculator.DDLB_WITHARROW))
```

5 Specify whether the dropdown closes on a single-click by calling the of_SetCloseOnClick function:

```
this.iuo_calculator.of_SetCloseOnClick(TRUE)
```

For information on using u_calculator with EditMask controls, see the *PFC User's Guide*.

See also

n_cst_dropdown
u_calendar
u_dw
u_em

Instance variables

U_calculator includes instance variables:

Instance variable	Description	Data type	Access	Usage
DDLB	Constant set to 2	Integer	Public	Use with of_ Register
DDLB_WITHARROW	Constant set to 3	Integer	Public	Use with of_ Register

Instance variable	Description	Data type	Access	Usage
EMPTY	Constant set to empty string	String	Protected	Internal
ib_closeonclick	Controls close on click	Boolean	Protected	Set with of_SetCloseOnClick
ib_initialvalue	Controls whether blank fields are initialized	Boolean	Protected	Set with of_SetInitialValue
ib_validresetvalue	Reports whether associated field contains a number	Boolean	Protected	Internal
idbl_repeatvalue	Repeat value	Double	Protected	Internal
idbl_value	Running total	Double	Protected	Internal
idrg_requestor	Generic reference	DragObject	Protected	Internal
idw_requestor	Associated DataWindow control	DataWindow	Protected	Internal
iem_requestor	Associated EditMask control	EditMask	Protected	Internal
ii_dwcolumnstyle[]	Edit styles of registered columns	Integer	Protected	Set with of_Register
inv_calculatorattrib	Calculator information	n_cst_calculatorattrib	Protected	Internal
inv_dropdown	Reference to dropdown service	n_cst_dropdown	Public	Internal
is_curroperator	Current operator	String	Protected	Internal
is_currvalue	Current value	String	Protected	Internal
is_dwcolumns[ ]	Registered columns	String	Protected	Set with of_Register

Instance variable	Description	Data type	Access	Usage
is_dwcolumnsexp[ ]	Original properties for registered columns	String	Protected	Set by of_Register
is_prevkeystroke	Previous key	String	Protected	Internal
is_repeatoperator	Repeat operator	String	Protected	Internal
NONE	Constant set to 1	Integer	Public	Use with of_Register

Events

U_calculator includes precoded events:

ButtonClicked	Key
Constructor	LoseFocus
Destructor	Pfc_DropDown
GetFocus	

ButtonClicked

Description

Calls the of_ButtonClicked function.

Applies to Dw_calculator

Usage

This event executes when the user clicks a command button, types a number, types an operator, or types a period.

Constructor

Description

Initializes object settings.

Applies to Dw_calculator and u_calculator

Usage

This event executes when the object is created.

Destructor

Description

Destroys the inv_dropdown instance variable.

Applies to U_calculator

Usage

This event executes when the u_calculator instance is destroyed.

GetFocus

Description

Calls the of_RedirectFocus function.

Applies to Dw_calculator

Usage

This event executes when a control receives focus, typically by clicking.

Key

Description

Determines which key was pressed and calls the of_ButtonClicked function.

Applies to Dw_calculator

Usage

This event executes when the user presses a key.

LoseFocus

Description

Hides the calculator.

Applies to Dw_calculator

Usage

This event executes when the calculator loses focus.

Pfc_DropDown

Description

Calls the of_DropDown function.

Applies to U_calculator

Return value

Integer. Returns 1 if the event succeeds, 0 if the DataWindow column has not been registered, and -1 if an error occurs.

Usage

This event is called by the u_dw pfc_DDCalculator event.

Functions

U_calculator contains precoded object functions:

of_ButtonClicked	of_PerformMath
of_DropDown	of_RedirectFocus
of_GetInfo	of_Register
of_GetPropertyInfo	of_Reset
of_GetRegisterable	of_SetCloseOnClick
of_GetRegistered	of_SetDropDown
of_GetRegisteredStyle	of_SetFocusOnRequestor
of_IsCloseOnClick	of_SetInitialValue
of_IsDropDown	of_SetRequestor
of_IsInitialValue	of_SetValue
of_IsNumericType	of_SetValueOnRequestor
of_IsOperator	of_UnRegister
of_IsRegistered	

of_ButtonClicked

Description

Processes button clicks.

Access

Protected

Syntax

dwcontrol.instancename.**of_ButtonClicked** (*key*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calculator with an EditMask control)
instancename	Instance name of u_calculator (the u_dw and u_em default for this value is iuo_calculator)
key	String specifying the text of the clicked button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from a the u_calculator dw_employee ButtonClicked event:

```
...
If Len(ls_buttontext) > 0 Then
   of_ButtonClicked(ls_buttontext)
End If
...
```

of_DropDown

Description

Displays the dropdown calculator in the appropriate location.

Access

Public

Syntax

dwcontrol.instancename.**of_DropDown** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calculator with an EditMask control)
instancename	Instance name of u_calculator (the u_dw and u_em default for this value is iuo_calculator)

Return value

Integer. Returns 1 if the function succeeds, 0 if the current DataWindow column has not been registered, and -1 if an error occurs.

Usage

The pfc_DropDown event calls this function. If the current DataWindow column is not registered, this function returns 0 and does not display the calculator.

Examples

This example is from the pfc_DropDown event:

```
Return of_DropDown()
```

of_GetInfo

Description

Retrieves object information.

Access

Public

Syntax

dwcontrol.instancename.**of_GetInfo** (*infoobject*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calculator with an EditMask control)
instancename	Instance name of u_calculator (the u_dw and u_em default for this value is iuo_calculator)
infoobject	N_cst_infoattrib instance into which the function places information (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

The DataWindow Properties window calls this function to access service information.

Examples

This example calls the of_GetInfo function:

```
n_cst_infoattrib  lnv_info

dw_1.iuo_calculator.of_GetInfo(lnv_info)
MessageBox("Info",  &
  "Description: " + lnv_info.is_description  &
    + ". Name: " + lnv_info.is_name)
```

of_GetPropertyInfo

Description

Retrieves information about the service's properties.

Access

Public

Syntax

dwcontrol.instancename.**of_GetPropertyInfo** (*propertyobject*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control
instancename	Instance name of u_calculator (the u_dw and u_em default for this value is iuo_calculator)
propertyobject	N_cst_propertyattrib instance into which the function places property information (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

The DataWindow Properties window calls this function to access service information.

Examples

This example calls the of_GetPropertyInfo function:

```
n_cst_propertyattrib  lnv_prop

dw_1.iuo_calculator.of_GetPropertyInfo(lnv_prop)
MessageBox("Info",  &
  "Description: " + lnv_prop.is_description  &
    + ". Name: " + lnv_prop.is_name  &
      + ". Property tab text: " +  &
        lnv_prop.is_propertytabtext)
```

of_GetRegisterable

Description

Retrieves an array of registerable DataWindow columns. Registerable DataWindow columns are numeric and use the DropDownListBox, Edit, or EditMask edit style.

Access

Public

Syntax

dwcontrol.instancename.**of_GetRegisterable** (*columns*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control
instancename	Instance name of u_calculator (the u_dw default for this value is iuo_calculator)
columns	Unbounded string array into which the function places the names of columns that have been registered with this instance of the dropdown calculator (passed by reference)

Return value

Integer. Returns the number of entries in the *columns* array if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_GetRegisterable function:

```
...
li_rc = &
  idw_requestor.iuo_calculator.of_GetRegisterable &
    (ls_allcols)
...
```

of_GetRegistered

Description

Retrieves an array of registered DataWindow columns.

Access

Public

Syntax

dwcontrol.instancename.**of_GetRegistered** (*columns* {, *colstyle* })

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control
instancename	Instance name of u_calculator (the u_dw default for this value is iuo_calculator)

Argument	Description
columns	Unbounded string array into which the function places the names of columns that have been registered with this instance of the dropdown calculator (passed by reference)
colstyle (optional)	Unbounded Integer array into which the function places the styles of columns that have been registered with this instance of the dropdown calculator (passed by reference)

Return value

Integer. Returns the number of entries in the *columns* array if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_GetRegistered function:

```
String ls_columns[ ], ls_display
Integer li_return, li_count

li_return = &
dw_1.iuo_calculator. of_GetRegistered &
  (ls_columns)
FOR li_count = 1 to li_return
  ls_display += ls_columns[li_count] + "~r~n"
NEXT
MessageBox("Columns", ls_display)
```

of_GetRegisteredStyle

Description

Retrieves the display style for a specified column.

Access

Public

Syntax

dwcontrol.instancename.**of_GetRegisteredStyle** (*column*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control
instancename	Instance name of u_calculator (the u_dw default for this value is iuo_calculator)
column	String containing the DataWindow column for which the function returns the display style

Return value

Integer. Returns the column style if the function succeeds, 0 if the column is not registered, and -1 if an error occurs.

Examples

This example calls the of_GetRegisteredStyle function:

```
Integer li_return

li_return = &
  dw_1.iuo_calculator.of_GetRegisteredStyle &
    ("start_date")
MessageBox("Display Style", &
"Display style is " + String(li_return))
```

of_IsCloseOnClick

Description

Reports whether the u_calculator instance closes when the user clicks the equal sign.

Access

Public

Syntax

dwcontrol.instancename.**of_IsCloseOnClick** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calculator with an EditMask control)
instancename	Instance name of u_calculator (the u_dw and u_em default for this value is iuo_calculator)

Return value

Boolean. Returns TRUE if the u_calculator instance closes when the user clicks the equal sign and FALSE if it does not.

Examples

This example calls the of_IsCloseOnClick function:

```
Boolean  lb_close
n_cst_conversion  lnv_conversion

lb_close = &
  dw_1.iuo_calculator.of_IsCloseOnClick()
MessageBox("Calculator", "Close on click is " &
  + lnv_conversion.of_String(lb_close))
```

of_IsDropDown

Description

Reports whether the u_calculator instance has enabled n_cst_dropdown.

Access

Public

Syntax

dwcontrol.instancename.**of_IsDropDown** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calculator with an EditMask control)
instancename	Instance name of u_calculator (the u_dw and u_em default for this value is iuo_calculator)

Return value

Boolean. Returns TRUE if the u_calculator instance has enabled n_cst_dropdown and FALSE if it has not.

Usage

The dropdown service is referenced by the inv_dropdown instance variable.

Examples

This example calls the of_IsDropDown function:

```
Boolean  lb_dropdown
n_cst_conversion  lnv_conversion

lb_dropdown = &
  dw_1.iuo_calculator.of_IsDropDown()
MessageBox("Calculator", &
  "DropDown behavior is " &
    + lnv_conversion.of_String(lb_dropdown))
```

of_IsInitialValue

Description

Reports whether the u_calculator instance initializes blank fields with zero when the calculator displays.

Access

Public

Syntax

dwcontrol.instancename.**of_IsInitialValue** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calculator with an EditMask control)
instancename	Instance name of u_calculator (the u_dw and u_em default for this value is iuo_calculator)

Return value

Boolean. Returns TRUE if the calculator initializes blank fields with a zero and FALSE if it does not.

Examples

This example calls the of_IsInitialValue function:

```
Boolean  lb_init
n_cst_conversion  lnv_conversion

lb_init = &
  dw_1.iuo_calculator.of_IsInitialValue()
MessageBox("Calculator", "Initialize is " &
  + lnv_conversion.of_String(lb_init))
```

of_IsNumericType

Description

Reports whether a specified DataWindow column data type is numeric (decimal, long, or number).

Access

Protected

Syntax

dwcontrol.instancename.**of_IsNumericType** (*datatype*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calculator with an EditMask control)
instancename	Instance name of u_calculator (the u_dw and u_em default for this value is iuo_calculator)
datatype	String specifying the data type

Return value

Boolean. Returns TRUE if *datatype* is numeric and FALSE if it is not.

Usage

Internal.

Examples

This example is from the of_DropDown function:

```
...
IF IsValid(idw_requestor) THEN
  ls_colname = idw_requestor.GetColumnName()
  IF NOT of_IsRegistered(ls_colname) THEN
    Return 0
  END IF
  ls_coltype = &
    idw_requestor.Describe(ls_colname+".coltype")
  IF NOT of_IsNumericType(ls_coltype) THEN
    Return -1
  END IF
END IF
...
```

of_IsOperator

Description

Reports whether a passed value is an arithmetic operator (/, *, +, or -).

Access

Protected

Syntax

dwcontrol.instancename.**of_IsOperator** (*operator*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calculator with an EditMask control)
instancename	Instance name of u_calculator (the u_dw and u_em default for this value is iuo_calculator)
operator	String containing the value to be tested

Return value

Boolean. Returns TRUE if *operator* is /, *, +, or - and FALSE if it is not.

Usage

Internal.

Examples

This example is from the of_ButtonClicked function:

```
...
CASE '/', '*', '+', '-', '='
IF of_IsOperator(as_key) AND &
  of_IsOperator(is_prevkeystroke) THEN
    is_prevkeystroke = as_key
    is_curroperator = as_key
    Return 1
END IF
...
```

of_IsRegistered

Description

Reports whether a specified DataWindow column has been registered.

Access

Public

Syntax

dwcontrol.instancename.**of_IsRegistered** (*column*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control
instancename	Instance name of u_calculator (the u_dw default for this value is iuo_calculator)

Argument	Description
column	String specifying the column to search for

Return value

Boolean. Returns TRUE if *column* has been registered and FALSE if it has not.

Examples

This example calls the of_IsRegistered function:

```
...
IF NOT this.iuo_calculator.of_IsRegistered &
  ("salary") THEN
    this.iuo_calculator.of_Register("salary")
END IF
...
```

of_PerformMath

Description

Updates the running total, using the specified operand and operator.

Access

Protected

Syntax

dwcontrol.instancename.**of_PerformMath** (*value, operator, operand*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calculator with an EditMask control)
instancename	Instance name of u_calculator (the u_dw and u_em default for this value is iuo_calculator)
value	Double into which the function places the running total (passed by reference)
operator	String specifying the operator
operand	Double containing the number to be used in the calculation

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the of_ButtonClicked function:

```
...
IF of_PerformMath(ldbl_value, &
  is_curroperator, ldbl_currvalue) < 0 THEN
    Return -1
```

```
END IF
...
```

of_RedirectFocus

Description

Prevents the u_calculator instance from getting focus while it is hidden.

Access

Protected

Syntax

dwcontrol.instancename.**of_RedirectFocus** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calculator with an EditMask control)
instancename	Instance name of u_calculator (the u_dw and u_em default for this value is iuo_calculator)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the dw_employee GetFocus event:

```
Post of_RedirectFocus()
```

of_Register

Registers columns that use the dropdown calculator. There are two syntaxes:

To register	Use
One or all eligible columns, optionally specifying a display style	Syntax 1
All eligible columns using a specified display style	Syntax 2

Syntax 1

Register columns with an optional display style

Description

Registers one or all eligible columns in a DataWindow. Eligible columns use a numeric data type and have the Edit, EditMask, or DropDownListBox edit style.

Access

Public

Syntax

dwcontrol.instancename.**of_Register** ({ *column* {, *style* } })

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control
instancename	Instance name of u_calculator (the u_dw default for this is iuo_calculator)
column	(Optional) String specifying the column to be registered. This column must have a numeric data type. If you omit this argument, the function registers all numeric columns
style	(Optional) Integer or u_calculator constant specifying the display style of registered DataWindow columns: ♦ **1 or NONE (default)** For columns that use the DropDownListBox edit style, the calculator displays automatically when the column gets focus. For columns that use the Edit and EditMask edit styles, the calculator does not display automatically; instead, you display the calculator by coding a call to the u_dw pfc_DDCalculator event ♦ **2 or DDLB** The function converts registered columns to the DropDownListBox edit style. Users display the calculator by clicking the down arrow, which *disappears* when the calculator displays ♦ **3 or DDLB_WITHARROW** The function converts registered columns to the DropDownListBox edit style. Users display the calculator by clicking the down arrow, which *remains* when the calculator displays

Return value

Integer. Returns the number of columns registered if the function succeeds, 0 if the column was already registered, and -1 if an error occurs. The number of columns registered includes hidden columns.

Usage

Register all appropriate DataWindow columns. You typically call this function in the DataWindow Constructor event.

To register all eligible columns in a DataWindow using a specified display style, see Syntax 2.

Examples

This example from a DataWindow Constructor event calls the of_Register function:

```
this.of_SetTransObject(SQLCA)
this.of_SetDropDownCalculator(TRUE)
this.iuo_calculator.of_Register ("salary", &
  this.iuo_calculator.NONE)
this.Event pfc_Retrieve()
```

Syntax 2 **Register columns with a display style**

Description

Registers all eligible columns in a DataWindow using the specified display style. Eligible columns use a numeric data type and have the Edit, EditMask, or DropDownListBox edit style.

Access

Public

Syntax

dwcontrol.instancename.**of_Register** (*style*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control
instancename	Instance name of u_calculator (the u_dw and u_em default for this is iuo_calculator)
style	Integer or u_calculator constant specifying the display style of registered DataWindow columns: ♦ **1 or NONE** For columns that use the DropDownListBox edit style, the calculator displays automatically when the column gets focus. For columns that use the Edit and EditMask edit styles, the calculator does not display automatically; instead, you display the calculator by coding a call to the u_dw pfc_DDCalculator event ♦ **2 or DDLB** The function converts registered columns to the DropDownListBox edit style. Users display the calculator by clicking the down arrow, which *disappears* when the calculator displays ♦ **3 or DDLB_WITHARROW** The function converts registered columns to the DropDownListBox edit style. Users display the calculator by clicking the down arrow, which *remains* when the calculator displays

Return value

Integer. Returns the number of columns registered if the function succeeds and -1 if an error occurs. The number of columns registered includes hidden columns.

Usage

Register all appropriate DataWindow columns. You typically call this function in the DataWindow Constructor event.

To register one or all eligible columns in a DataWindow, optionally specifying a display style, see Syntax 1.

Examples

This example from a DataWindow Constructor event calls the of_Register function:

```
this.of_SetTransObject(SQLCA)
```

```
this.of_SetDropDownCalculator(TRUE)
this.iuo_calculator. of_Register &
  (this.iuo_calculator.NONE)
this.Event pfc_Retrieve()
```

of_Reset

Description

Resets the u_calculator running total to that displayed in the DataWindow column or EditMask control.

Access

Public

Syntax

dwcontrol.instancename.**of_Reset** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calculator with an EditMask control)
instancename	Instance name of u_calculator (the u_dw and u_em default for this value is iuo_calculator)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the of_DropDown function:

```
...
of_Reset()
...
```

of_SetCloseOnClick

Description

Specifies whether to hide the u_calculator instance when the user clicks the equal sign.

Access

Public

Syntax

dwcontrol.instancename.**of_SetCloseOnClick** (*boolean*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calculator with an EditMask control)

Argument	Description
instancename	Instance name of u_calculator (the u_dw and u_em default for this value is iuo_calculator)
boolean	Boolean specifying whether to hide the dropdown calculator when the user clicks the equal sign (TRUE) or leave it displayed (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetCloseOnClick function:

```
this.of_SetTransObject(SQLCA)
this.of_SetDropDownCalculator(TRUE)
this.iuo_calculator.of_Register()
this.iuo_calculator.of_SetCloseOnClick(FALSE)
```

of_SetDropDown

Description

Enables or disables the dropdown service (n_cst_dropdown).

Access

Public

Syntax

dwcontrol.instancename.**of_SetDropDown** (*boolean*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calculator with an EditMask control)
instancename	Instance name of u_calculator (the u_dw and u_em default for this value is iuo_calculator)
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the dropdown service

Return value

Integer. Returns 1 if the function succeeds, -0 if the dropdown service already exists, and -1 if an error occurs.

Examples

This example is from the u_calculator Constructor event:

```
...
IF inv_calculatorattrib.ib_dropdown THEN
  this.Visible = FALSE
  of_SetDropDown(TRUE)
END IF
...
```

of_SetFocusOnRequestor

Description

Retrieves focus to the associated DataWindow column or EditMask control.

Access

Protected

Syntax

dwcontrol.instancename.**of_SetFocusOnRequestor** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calculator with an EditMask control)
instancename	Instance name of u_calculator (the u_dw and u_em default for this value is iuo_calculator)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the of_RedirectFocus function:

```
If this.Visible = False Then
   Return of_SetFocusOnRequestor()
End If
Return 1
```

of_SetInitialValue

Description

Specifies whether the u_calculator instance initializes blank fields with zero when the calculator displays.

Access

Public

Syntax

dwcontrol.instancename.**of_SetInitialValue** (*boolean*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calculator with an EditMask control)
instancename	Instance name of u_calculator (the u_dw and u_em default for this value is iuo_calculator)
boolean	Boolean indicating whether the calculator initializes blank fields with zero (TRUE) or not (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

For DataWindows that disable this option, if a user displays the dropdown calculator and moves directly to another column, the original column retains a status of NotModified!.

Examples

This example calls the of_SetInitialValue function:

```
this.of_SetTransObject(SQLCA)
this.of_SetDropDownCalculator(TRUE)
this.iuo_calculator.of_Register()
this.iuo_calculator.of_SetInitialValue(TRUE)
```

of_SetRequestor

Description

Associates an instance of u_calculator with a DataWindow or EditMask control.

Access

Public

Syntax

dwcontrol.instancename.**of_SetRequestor** (*requestor*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calculator with an EditMask control)
instancename	Instance name of u_calculator (the u_dw and u_em default for this value is iuo_calculator)
requestor	DragObject referencing the DataWindow or EditMask control to associate with this instance of u_calculator

Return value

Integer. Returns values as follows:

1 Success
-1 An error occurred
-2 The u_calculator instance has not enabled the dropdown service
-3 The EditMask mask type is not numeric or decimal

Usage

EditMask controls must have a mask type of numeric or decimal.

If you are not using the u_calculator object as a dropdown object (displaying it permanently on a window, user object, or tab), call this function to associate the u_calculator instance with the EditMask control that displays the results.

Examples

This example is from the u_em of_SetDropDownCalculator function:

```
...
IF ab_switch THEN
```

```
    IF NOT IsValid (iuo_calculator) THEN
      lw_parent.OpenUserObject(iuo_calculator)
      iuo_calculator.of_SetRequestor(this)
      Return 1
    END IF
  ELSE
    IF IsValid (iuo_calculator) THEN
      lw_parent.CloseUserObject(iuo_calculator)
      Return 1
    END IF
  END IF
  ...
```

of_SetValue

Description

Updates the running total with the specified value.

Access

Protected

Syntax

dwcontrol.instancename.**of_SetValue** (*value*, *sendtorequestor*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calculator with an EditMask control)
instancename	Instance name of u_calculator (the u_dw and u_em default for this value is iuo_calculator)
value	Double containing the new value
sendtorequestor	Boolean indicating whether to copy *value* to the requestor object (TRUE) or not (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the of_ButtonClicked function:

```
...
CHOOSE CASE as_key
CASE 'c'
// Clear the current variables.
is_curroperator = EMPTY
is_currvalue = EMPTY
```

```
// Clear the repeat variables.
is_repeatoperator = EMPTY
idbl_repeatvalue = 0
// Clear the Running value.
of_SetValue(0, True)
...
```

of_SetValueOnRequestor

Description

Copies the running total to the current DataWindow column or associated EditMask control.

Access

Protected

Syntax

dwcontrol.instancename.**of_SetValueOnRequestor** (*value*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calculator with an EditMask control)
instancename	Instance name of u_calculator (the u_dw and u_em default for this value is iuo_calculator)
value	String containing the value to set on the requestor object

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the of_Value function:

```
...
IF ab_setrequestor THEN
  of_SetValueOnRequestor(is_value)
END IF
...
```

of_UnRegister

Description

Removes one or all columns from the list of registered columns.

Access

Public

Syntax

dwcontrol.instancename.**of_Register** ({ *column* })

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control
instancename	Instance name of u_calculator (the u_dw default for this is iuo_calculator)
column (optional)	String specifying the column to be unregistered. If you omit this argument, the function unregisters all columns

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_UnRegister function:

```
...
dw_emp.iuo_calculator.of_UnRegister("salary")
...
```

u_calendar

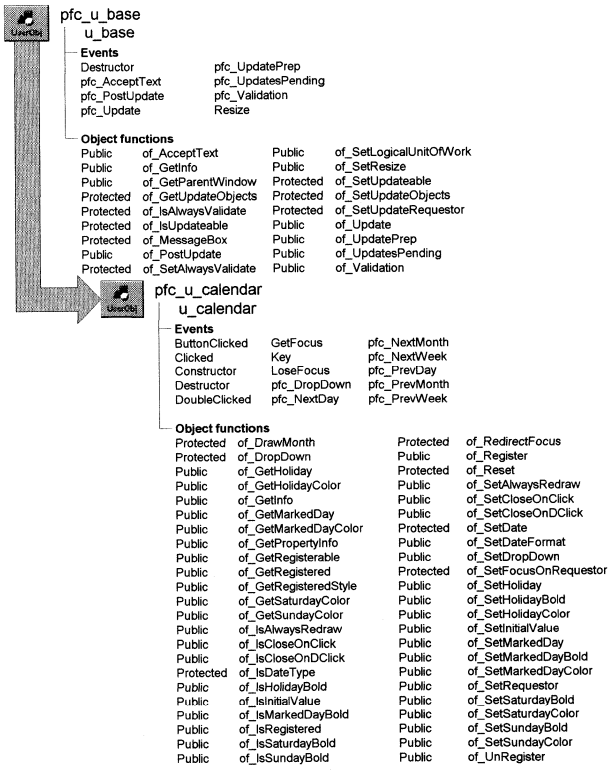
Description

Calendar object. Users enter dates by clicking on the dropdown calendar, automatically entering the selected date in the associated field. They change months by clicking the >> and << buttons and can also navigate the calendar with keyboard arrow keys.

U_calendar provides support for highlighting:

- Saturdays
- Sundays
- Holidays
- Marked days (any other special days that might apply to your application)

Ancestry



Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Object relationships

n_cst_dropdown
u_cb
u_dw

Usage

Use this object to provide a dropdown calendar for date values in either of the following:

U_dw-based DataWindow control
U_em-based EditMask control
Standalone calendar, for use with or without an EditMask

To use this service with DataWindow columns:

1 Place a u_dw-based DataWindow control on the window or user object.

2 Enable the dropdown calendar by calling the u_dw of_SetDropDownCalendar function (this example is from a DataWindow Constructor event):

```
this.of_SetDropDownCalendar(TRUE)
```

3 Register columns one by one or all at once by calling the of_Register function. Of_Register includes an argument specifying the dropdown style:

```
this.iuo_calendar.of_Register("salary", &
  this.iuo_calendar.DDLB)
```

4 (Optional) Establish the font style and color for weekend days:

```
this.iuo_calendar.of_SetSaturdayBold(TRUE)
this.iuo_calendar.of_SetSaturdayColor &
  (RGB(0, 255, 0))
this.iuo_calendar.of_SetSundayBold(TRUE)
this.iuo_calendar.of_SetSundayColor &
  (RGB(0, 255, 0))
```

5 (Optional) Establish a list of holidays with their font style and color (this example shows holidays for one year only):

```
Date ld_holidays[11]

ld_holidays[1] = 1997-01-01
ld_holidays[2] = 1997-02-17
ld_holidays[3] = 1997-04-21
ld_holidays[4] = 1997-05-26
ld_holidays[5] = 1997-07-04
ld_holidays[6] = 1997-09-01
ld_holidays[7] = 1997-10-13
ld_holidays[8] = 1997-11-27
```

```
ld_holidays[9] = 1997-11-28
ld_holidays[10] = 1997-12-25
ld_holidays[11] = 1997-12-26
...
this.iuo_calendar.of_SetHoliday(ld_holidays)
this.iuo_calendar.of_SetHolidayBold(TRUE)
this.iuo_calendar.of_SetHolidayColor &
  (RGB(0, 255, 0))
```

6 (Optional) Establish a list of marked days with their font style and color:

```
Date ld_marked_days[12]

ld_marked_days[1] = 1996-06-13
ld_marked_days[2] = 1996-03-16
ld_marked_days[3] = 1996-09-23
ld_marked_days[4] = 1996-09-14
ld_marked_days[5] = 1997-06-13
ld_marked_days[6] = 1997-03-16
ld_marked_days[7] = 1997-09-23
ld_marked_days[8] = 1997-09-14
ld_marked_days[9] = 1998-06-13
ld_marked_days[10] = 1998-03-16
ld_marked_days[11] = 1998-09-23
ld_marked_days[12] = 1998-09-14
...
this.iuo_calendar.of_SetMarkedDay(ld_marked_days)
this.iuo_calendar.of_SetMarkedDayBold(TRUE)
this.iuo_calendar.of_SetMarkedDayColor &
  (RGB(255, 0, 0))
```

7 (Optional) Call additional functions as necessary to customize calendar behavior:

```
this.iuo_calendar.of_SetCloseOnClick(FALSE)
this.iuo_calendar.of_SetCloseOnDClick(TRUE)
this.iuo_calendar.of_SetInitialValue(TRUE)
```

For information on using u_calendar with EditMask controls and as a standalone calendar, see the *PFC User's Guide*.

See also

u_calculator
u_dw
u_em

Instance variables

U_calendar includes instance variables.

Instance variable	Description	Data type	Access	Usage
DDLB	Constant set to 2	Integer	Public	Use with of_Register
DDLB_WITHARROW	Constant set to 3	Integer	Public	Use with of_Register
ib_alwaysredraw	Controls redraw	Boolean	Protected	Internal
ib_closeonclick	Controls close on click	Boolean	Protected	Set with of_SetCloseOnClick
ib_closeondclick	Controls close on double-click	Boolean	Protected	Set with of_SetCloseOnDClick
ib_holidaybold	Controls whether holidays are bold	Boolean	Protected	Set with of_SetHolidayBold
ib_initialvalue	Controls whether blank fields are initialized	Boolean	Protected	Set with of_SetInitialValue
ib_markeddaybold	Controls whether marked days are bold	Boolean	Protected	Set with of_SetMarkedDayBold
ib_saturdaybold	Controls whether Saturdays are bold	Boolean	Protected	Set with of_SetSaturdayBold
ib_sundaybold	Controls whether Sundays are bold	Boolean	Protected	Set with of_SetSundayBold
id_date	Current date	Date	Protected	Internal
id_holiday[ ]	List of holidays	Date	Protected	Set with of_SetHoliday
id_markedday[ ]	List of marked days	Date	Protected	Set with of_SetMarkedDay
id_prevdate	Previous date	Date	Protected	Internal
id_resetdate	Date to use if invalid date is entered	Date	Protected	Internal
idrg_requestor	Generic reference	DragObject	Protected	Internal
idw_requestor	Associated DataWindow control	DataWindow	Protected	Internal

Instance variable	Description	Data type	Access	Usage
iem_requestor	Associated EditMask control	EditMask	Protected	Internal
ii_boldfontweight	Bold font weight	Integer	Protected	Internal (default is 700)
ii_dwcolumnstyle[ ]	Edit styles of registered columns	Integer	Protected	Set with of_Register
ii_normalfontweight	Normal font weight	Integer	Protected	Internal (default is 400)
il_fontcolor	Font color for normal days	Long	Protected	Internal
il_holidaycolor	Font color for holidays	Long	Protected	Set with of_SetHolidayColor
il_markeddaycolor	Font color for marked days	Long	Protected	Set with of_SetMarkedDayColor
il_saturdaycolor	Font color for Saturdays	Long	Protected	Set with of_SetSaturdayColor
il_sundaycolor	Font color for Sundays	Long	Protected	Set with of_SetSundayColor
inv_calendarattrib	Calendar attributes	n_cst_calendarattrib	Public	Internal
inv_datetime	Reference to date/time service	n_cst_datetime	Public	Internal
inv_dropdown	Reference to dropdown service	n_cst_dropdown	Public	Internal
is_dateformat	Date format	String	Protected	Set with of_SetDateFormat
is_dwcolumns[ ]	Registered columns	String	Protected	Set with of_Register
is_dwcolumnsexp[ ]	Original properties for registered columns	String	Protected	Set with of_Register
is_prevcell	Previous cell	String	Protected	Internal
NONE	Constant set to 1	Integer	Public	Use with of_Register

Events

U_calendar includes precoded events:

ButtonClicked	pfc_DropDown
Clicked	pfc_NextDay
Constructor	pfc_NextMonth
Destructor	pfc_NextWeek
DoubleClicked	pfc_PrevDay
GetFocus	pfc_PrevMonth
Key	pfc_PrevWeek
LoseFocus	

ButtonClicked

Description

Changes the month.

Applies to Dw_cal

Usage

This event executes when the user clicks the next or previous month buttons.

Clicked

Description

Changes the month or day.

Applies to Dw_cal

Usage

This event executes when the user clicks a date.

Constructor

Description

Initializes object settings.

Applies to Dw_cal and u_calendar

Usage

This event executes when the object is created.

Destructor

Description

Destroys the inv_dropdown instance variable.

Applies to U_calendar

Usage

This event executes when the u_calendar instance is destroyed.

DoubleClicked

Description

Hides the dropdown calendar if close on double-click has been set with the of_SetCloseOnDClick function.

Applies to Dw_cal

Usage

This event executes when the user double-clicks on the dw_cal DataWindow control.

GetFocus

Description

Calls the of_RedirectFocus function.

Applies to Dw_cal

Usage

This event executes when a control receives focus, typically by clicking.

Key

Description

Provides keyboard support.

Applies to Dw_cal

Usage

This event executes when the user presses one of the following keys:

ESC
TAB
ENTER
LEFT, RIGHT, UP, or DOWN ARROW
PAGE UP or PAGE DOWN

LoseFocus

Description

Hides the calendar.

Applies to Dw_cal

Usage

This event executes when the calendar loses focus.

pfc_DropDown

Description

Calls the of_DropDown function.

Applies to U_calendar

Return value

Integer. Returns 1 if the event succeeds, 0 if the DataWindow column has not been registered, and -1 if an error occurs.

Usage

This event, which calls the of_DropDown function, is called by the u_dw pfc_DDCalendar event.

pfc_NextDay

Description

Advances the current date by one day.

Applies to Dw_cal

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is called by the Key event when the user presses the RIGHT ARROW key.

pfc_NextMonth

Description

Advances the current date by one month.

Applies to Dw_cal

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is called by the Key event when the user presses the PAGE DOWN key.

pfc_NextWeek

Description

Advances the current date by one week.

Applies to Dw_cal

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is called by the Key event when the user presses the DOWN ARROW key.

pfc_PrevDay

Description

Sets the current date back by one day.

Applies to Dw_cal

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is called by the Key event when the user presses the LEFT ARROW key.

pfc_PrevMonth

Description

Sets the current date back by one month.

Applies to Dw_cal

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is called by the Key event when the user presses the PAGE DOWN key.

pfc_PrevWeek

Description

Sets the current date back by one week.

Applies to Dw_cal

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is called by the Key event when the user presses the UP ARROW key.

Functions

U_calendar contains precoded object functions:

of_DrawMonth	of_RedirectFocus
of_DropDown	of_Register
of_GetHoliday	of_Reset
of_GetHolidayColor	of_SetAlwaysRedraw
of_GetInfo	of_SetCloseOnClick
of_GetMarkedDay	of_SetCloseOnDClick
of_GetMarkedDayColor	of_SetDate

of_GetPropertyInfo	of_SetDateFormat
of_GetRegisterable	of_SetDropDown
of_GetRegistered	of_SetFocusOnRequestor
of_GetRegisteredStyle	of_SetHoliday
of_GetSaturdayColor	of_SetHolidayBold
of_GetSundayColor	of_SetHolidayColor
of_IsAlwaysRedraw	of_SetInitialValue
of_IsCloseOnClick	of_SetMarkedDay
of_IsCloseOnDClick	of_SetMarkedDayBold
of_IsDateType	of_SetMarkedDayColor
of_IsHolidayBold	of_SetRequestor
of_IsInitialValue	of_SetSaturdayBold
of_IsMarkedDayBold	of_SetSaturdayColor
of_IsRegistered	of_SetSundayBold
of_IsSaturdayBold	of_SetSundayColor
of_IsSundayBold	of_UnRegister

of_DrawMonth

Description

Draws the specified month.

Access

Protected

Syntax

dwcontrol.instancename.**of_DrawMonth** (*date*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
date	Date specifying the month to draw

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example calls the of_DrawMonth function:

```
...
IF Year(ad_date) <> Year(id_prevdate) OR &
  Month(ad_date) <> Month(id_prevdate) THEN
    of_DrawMonth(ad_date)
END IF
...
```

of_DropDown

Description

Displays the dropdown calendar in the appropriate location.

Access

Public

Syntax

dwcontrol.instancename.**of_DropDown** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)

Return value

Integer. Returns 1 if the function succeeds, 0 if the current DataWindow column has not been registered, and -1 if an error occurs.

Usage

The u_dw and u_em pfc_DDCalendar event calls this function.

Examples

This example is from the u_em pfc_DDCalendar event:

```
IF IsValid(iuo_calendar) THEN
  Return iuo_calendar.of_DropDown()
END IF
```

of_GetHoliday

Description

Retrieves an array containing all specified holidays.

Access

Public

Syntax

dwcontrol.instancename.**of_GetHoliday** (*holidays*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
holidays	Unbounded Date array into which the function places all specified holidays (passed by reference)

Return value

Integer. Returns the number of entries in the *holidays* array.

Examples

This example calls the of_GetHoliday function:

```
String ls_display
Date ldt_holidays[ ]
Integer li_return, li_count

li_return = &
  dw_caltest.iuo_calendar.of_GetHoliday &
    (ldt_holidays)
FOR li_count = 1 to li_return
  ls_display += String(ldt_holidays[li_count])
  ls_display += "~r~n"
NEXT
MessageBox("Holidays", ls_display)
```

of_GetHolidayColor

Description

Retrieves the display color for holidays.

Access

Public

Syntax

dwcontrol.instancename.**of_GetHolidayColor** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)

Return value

Long. Returns the color for holidays.

Examples

This example calls the of_GetHolidayColor function:

```
MessageBox("Holidays", "Holiday color is " + String &
  (dw_caltest.iuo_calendar.of_GetHolidayColor()))
```

of_GetInfo

Description

Retrieves object information.

Access

Public

Syntax

dwcontrol.instancename.**of_GetInfo** (*infoobject*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
infoobject	N_cst_infoattrib instance into which the function places information (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

The DataWindow Properties window calls this function to access service information.

Examples

This example calls the of_GetInfo function:

```
n_cst_infoattrib  lnv_info

dw_1.iuo_calendar.of_GetInfo(lnv_info)
MessageBox("Info",  &
  "Description: " + lnv_info.is_description  &
    + ". Name: " + lnv_info.is_name)
```

of_GetMarkedDay

Description

Retrieves an array containing all marked days.

Access

Public

Syntax

*dwcontrol.instancename.***of_GetMarkedDay** (*markeddays*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
markeddays	Unbounded Date array into which the function places all specified marked days (passed by reference)

Return value

Integer. Returns the number of entries in the *markeddays* array.

Examples

This example calls the of_GetMarkedDay function:

```
String ls_display
Date ldt_marked_days[ ]
Integer li_return, li_count

li_return = &
  dw_caltest.iuo_calendar.of_GetMarkedDay &
    (ldt_marked_days)
FOR li_count = 1 to li_return
  ls_display += String(ldt_marked_days[li_count])
  ls_display += "~r~n"
NEXT
MessageBox("Marked Days", ls_display)
```

of_GetMarkedDayColor

Description: Retrieves the display color for marked days.

Access: Public

Syntax: *dwcontrol.instancename*.**of_GetMarkedDayColor** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)

Return value: Long. Returns the color for marked days.

Examples: This example calls the of_GetMarkedDayColor function:

```
MessageBox("Marked Days", &
  "Marked day color is " + String &
    (dw_1.iuo_calendar.of_GetMarkedDayColor()))
```

of_GetPropertyInfo

Description: Retrieves information about the service's properties.

Access: Public

Syntax: *dwcontrol.instancename*.**of_GetPropertyInfo** (*propertyobject*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
propertyobject	N_cst_propertyattrib instance into which the function places property information (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

The DataWindow Properties window calls this function to access service information.

Examples

This example calls the of_GetPropertyInfo function:

```
n_cst_propertyattrib  lnv_prop

dw_1.iuo_calendar.of_GetPropertyInfo(lnv_prop)
MessageBox("Info",  &
  "Description: " + lnv_prop.is_description  &
    + ". Name: " + lnv_prop.is_name  &
      + ". Property tab text: " +  &
        lnv_prop.is_propertytabtext)
```

of_GetRegisterable

Description

Retrieves an array of registerable DataWindow columns. Registerable DataWindow columns have the date data type and use the DropDownListBox, Edit, or EditMask edit style.

Access

Public

Syntax

dwcontrol.instancename.**of_GetRegisterable** (*columns*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control
instancename	Instance name of u_calendar (the u_dw default for this value is iuo_calendar)
columns	Unbounded string array into which the function places the names of columns that have been registered with this instance of the dropdown calendar (passed by reference)

Return value

Integer. Returns the number of entries in the *columns* array if the function succeeds and -1 if an error occurs.

Usage

Call this function to determine which DataWindow columns could be registered with this instance of u_calendar.

Examples

This example calls the of_GetRegisterable function:

```
...
li_rc = &
  idw_requestor.iuo_calendar. of_GetRegisterable &
    (ls_allcols)
...
```

of_GetRegistered

Description

Retrieves an array of registered DataWindow columns.

Access

Public

Syntax

dwcontrol.instancename.**of_GetRegistered** (*columns*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control
instancename	Instance name of u_calendar (the u_dw default for this value is iuo_calendar)
columns	Unbounded string array into which the function places the names of columns that have been registered with this instance of the dropdown calendar (passed by reference)

Return value

Integer. Returns the number of entries in the *columns* array if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_GetRegistered function:

```
String ls_columns[], ls_display
Integer li_return, li_count

li_return = &
dw_1.iuo_calendar.of_GetRegistered &
(ls_columns)
FOR li_count = 1 to li_return
ls_display += ls_columns[li_count] + "~r~n"
NEXT
```

```
MessageBox("Columns", ls_display)
```

of_GetRegisteredStyle

Description

Retrieves the display style for a registered column.

Access

Public

Syntax

dwcontrol.instancename.**of_GetRegisteredStyle** (*column*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control
instancename	Instance name of u_calendar (the u_dw default for this value is iuo_calendar)
column	String containing the DataWindow column for which the function returns the display style

Return value

Integer. Returns 1 if the function succeeds, 0 if the column is not registered, and -1 if an error occurs.

Examples

This example calls the of_GetRegisteredStyle function:

```
Integer li_return

li_return = &
  dw_1.iuo_calendar.of_GetRegisteredStyle &
    ("start_date")
MessageBox("Display Style", &
  "Display style is " + String(li_return))
```

of_GetSaturdayColor

Description

Retrieves the display color for Saturdays.

Access

Public

Syntax

dwcontrol.instancename.**of_GetSaturdayColor** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)

Argument	Description
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)

Return value

Long. Returns the color for Saturdays.

Examples

This example calls the of_GetSaturdayColor function:

```
MessageBox("Saturdays", "Saturday color is " + &
  String &
    (dw_caltest.iuo_calendar.of_GetSaturdayColor()))
```

of_GetSundayColor

Description

Retrieves the display color for Sundays.

Access

Public

Syntax

dwcontrol.instancename.**of_GetSundayColor** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)

Return value

Long. Returns the color for Sundays.

Examples

This example calls the of_GetSundayColor function:

```
MessageBox("Sundays", "Sunday color is " + &
  String &
    (dw_caltest.iuo_calendar.of_GetSundayColor()))
```

of_IsAlwaysRedraw

Description

Reports whether the u_calendar instance forces calendar redraw when display properties change.

Access

Public

Syntax

dwcontrol.instancename.**of_IsAlwaysRedraw** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)

Return value

Boolean. Returns TRUE if the u_calendar instance forces redraw and FALSE if it does not.

Examples

This example calls the of_IsAlwaysRedraw function:

```
Boolean lb_redraw
n_cst_conversion lnv_conversion

lb_redraw = &
  dw_1.iuo_calendar.of_IsAlwaysRedraw()
MessageBox("Calendar", "Redraw is " &
  + lnv_conversion.of_String(lb_redraw))
```

of_IsCloseOnClick

Description

Reports whether the u_calendar instance closes when the user clicks a date.

Access

Public

Syntax

dwcontrol.instancename.**of_IsCloseOnClick** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)

Return value

Boolean. Returns TRUE if the u_calendar instance closes when the user clicks a date and FALSE if it does not.

Examples

This example calls the of_IsCloseOnClick function:

```
Boolean lb_close
n_cst_conversion lnv_conversion

lb_close = &
  dw_1.iuo_calendar.of_IsCloseOnClick()
```

```
MessageBox("Calendar", "Close on click is " &
  + lnv_conversion.of_String(lb_close))
```

of_IsCloseOnDClick

Description

Reports whether the u_calendar instance closes when the user double-clicks a date.

Access

Public

Syntax

dwcontrol.instancename.**of_IsCloseOnDClick** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)

Return value

Boolean. Returns TRUE if the u_calendar instance closes when the user double-clicks a date and FALSE if it does not.

Examples

This example calls the of_IsCloseOnDClick function:

```
Boolean lb_close
n_cst_conversion lnv_conversion

lb_close = &
  dw_1.iuo_calendar.of_IsCloseOnDClick()
MessageBox("Calendar", &
  "Close on double-click is " &
    + lnv_conversion.of_String(lb_close))
```

of_IsDateType

Description

Reports whether a specified DataWindow column uses the date data type.

Access

Protected

Syntax

dwcontrol.instancename.**of_IsDateType** (*datatype*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)

Argument	Description
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
datatype	String specifying the data type

Return value

Boolean. Returns TRUE if *datatype* is date and FALSE if it is not.

Usage

Internal.

Examples

This example is from the of_DropDown function:

```
...
IF NOT of_IsDateType(idw_requestor.Describe &
  (ls_colname+".coltype")) THEN
    Return -1
END IF
...
```

of_IsHolidayBold

Description

Reports whether holidays are bold.

Access

Public

Syntax

dwcontrol.instancename.**of_IsHolidayBold** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)

Return value

Boolean. Returns TRUE if holidays are bold and FALSE if they are not.

Examples

This example calls the of_IsHolidayBold function:

```
IF dw_caltest.iuo_calendar.of_IsHolidayBold() THEN
  MessageBox("Holidays", "Holidays are bold")
ELSE
  MessageBox("Holidays", "Holidays are not bold")
END IF
```

of_IsInitialValue

Description

Reports whether the u_calendar instance initializes blank fields with the current date when the calendar displays.

Access

Public

Syntax

dwcontrol.instancename.**of_IsInitialValue** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)

Return value

Boolean. Returns TRUE if the calendar initializes blank fields with the current date and FALSE if it does not.

Examples

This example calls the of_IsInitialValue function:

```
Boolean lb_today
n_cst_conversion lnv_conversion

lb_today = &
  dw_caltest.iuo_calendar.of_IsInitialValue()
MessageBox("Calendar", &
  "Use today's date value is " &
    + lnv_conversion.of_String(lb_today))
```

of_IsMarkedDayBold

Description

Reports whether marked days are bold.

Access

Public

Syntax

dwcontrol.instancename.**of_IsMarkedDayBold** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)

Return value

Boolean. Returns TRUE if marked days are bold and FALSE if they are not.

Examples

This example calls the of_IsMarkedDayBold function:

```
IF dw_1.iuo_calendar.of_IsMarkedDayBold() THEN
  MessageBox("Marked Days", "Marked days are bold")
ELSE
  MessageBox("Marked Days",  &
    "Marked days are not bold")
END IF
```

of_IsRegistered

Description

Reports whether a specified DataWindow column has been registered with this instance of u_calendar.

Access

Public

Syntax

dwcontrol.instancename.**of_IsRegistered** (*column*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control
instancename	Instance name of u_calendar (the u_dw default for this value is iuo_calendar)
column	String specifying the column to search for

Return value

Boolean. Returns TRUE if *column* has been registered and FALSE if it has not.

Examples

This example calls the of_IsRegistered function:

```
...
IF NOT this.iuo_calendar.of_IsRegistered &
  ("birth_date") THEN
    this.iuo_calendar.of_Register("birth_date")
END IF
...
```

of_IsSaturdayBold

Description

Reports whether Saturdays are bold.

Access

Public

Syntax

dwcontrol.instancename.**of_IsSaturdayBold** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)

Return value

Boolean. Returns TRUE if Saturdays are bold and FALSE if they are not.

Examples

This example calls the of_IsSaturdayBold function:

```
IF dw_caltest.iuo_calendar. of_IsSaturdayBold () THEN
  MessageBox("Saturdays", "Saturdays are bold")
ELSE
  MessageBox("Saturdays", "Saturdays are not bold")
END IF
```

of_IsSundayBold

Description

Reports whether Sundays are bold.

Access

Public

Syntax

dwcontrol.instancename.**of_IsSundayBold** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)

Return value

Boolean. Returns TRUE if Sundays are bold and FALSE if they are not.

Examples

This example calls the of_IsSundayBold function:

```
IF dw_caltest.iuo_calendar. of_IsSundayBold () THEN
  MessageBox("Sundays", "Sundays are bold")
ELSE
  MessageBox("Sundays", "Sundays are not bold")
END IF
```

of_RedirectFocus

Description

Prevents the u_calendar instance from getting focus while it is hidden.

Access

Protected

Syntax

dwcontrol.instancename.**of_RedirectFocus** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the dw_cal GetFocus event:

```
Post of_RedirectFocus()
```

of_Register

Registers columns that use the dropdown calendar. There are two syntaxes:

To register	Use
One or all eligible columns, optionally specifying a display style	Syntax 1
All eligible columns using a specified display style	Syntax 2

Syntax 1

Register columns with an optional display style

Description

Registers one or all eligible columns in a DataWindow. Eligible columns use the date data type and have the Edit, EditMask, or DropDownListBox edit style.

Access

Public

Syntax

dwcontrol.instancename.**of_Register** ({ *column* {, *style* } })

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control
instancename	Instance name of u_calendar (the u_dw default for this value is iuo_calendar)

Argument	Description
column	(Optional) String specifying the column to be registered. This column must have a date data type. If you omit this argument, the function registers all columns that use the date data type
style	(Optional) Integer or u_calendar constant specifying the display style of registered DataWindow columns: ♦ **1 or NONE (default)** For columns that use the DropDownListBox edit style, the calendar displays automatically when the column gets focus. For columns that use the Edit and EditMask edit styles, the calendar does not display automatically; instead, you display the calendar by coding a call to the u_dw pfc_DDCalendar event ♦ **2 or DDLB** The function converts registered columns to the DropDownListBox edit style. Users display the calendar by clicking the down arrow, which *disappears* when the calendar displays ♦ **3 or DDLB_WITHARROW** The function converts registered columns to the DropDownListBox edit style. Users display the calendar by clicking the down arrow, which *remains* when the calendar displays

Return value

Integer. Returns the number of columns registered if the function succeeds, 0 if the column was already registered, and -1 if an error occurs. The number of columns registered includes hidden columns.

Usage

Call this function to register one or all date columns in a DataWindow.

To register all eligible columns in a DataWindow using a specified display style, use Syntax 2.

Examples

This example calls the of_Register function:

```
this.of_SetTransObject(SQLCA)
this.of_SetDropDownCalendar(TRUE)
this.iuo_calendar.of_Register("start_date", &
  this.iuo_calendar.DDLB)
this.iuo_calendar.of_Register("termination_date", &
  this.iuo_calendar.NONE)
this.iuo_calendar.of_Register("birth_date", &
  this.iuo_calendar.DDLB)
this.Event pfc_Retrieve()
```

Syntax 2 **Register columns with a display style**

Description

Registers all eligible columns in a DataWindow using the specified display style. Eligible columns use the date data type and have the Edit, EditMask, or DropDownListBox edit style.

Access

Public

Syntax

dwcontrol.instancename.**of_Register** (*style*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control
instancename	Instance name of u_calendar. The u_dw and u_em default for this is iuo_calendar
style	Integer or u_calendar constant specifying the display style of registered DataWindow columns: ♦ **1 or NONE** For columns that use the DropDownListBox edit style, the calendar displays automatically when the column gets focus. For columns that use the Edit and EditMask edit styles, the calendar does not display automatically; instead, you display the calendar by coding a call to the u_dw pfc_DDCalendar event ♦ **2 or DDLB** The function converts registered columns to the DropDownListBox edit style. Users display the calendar by clicking the down arrow, which *disappears* when the calendar displays ♦ **3 or DDLB_WITHARROW** The function converts registered columns to the DropDownListBox edit style. Users display the calendar by clicking the down arrow, which *remains* when the calendar displays

Return value

Integer. Returns the number of columns registered if the function succeeds and -1 if an error occurs. The number of columns registered includes hidden columns.

Usage

Call this function to register all Date columns in a DataWindow, using a specified display style.

To register one or all eligible columns in a DataWindow, optionally specifying a display style, use Syntax 1.

Examples

This example calls the of_Register function:

```
this.of_SetTransObject(SQLCA)
this.of_SetDropDownCalendar(TRUE)
```

```
this.iuo_calendar.of_Register &
  (this.iuo_calendar.NONE)
this.Event pfc_Retrieve()
```

of_Reset

Description

Resets the u_calendar date to that displayed in the DataWindow column or EditMask control.

Access

Protected

Syntax

dwcontrol.instancename.**of_Reset** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the of_DropDown function:

```
...
li_rc = inv_dropdown.of_Position &
  (idrg_requestor, FALSE)
IF li_rc < 0 THEN Return -1
  of_Reset ()
  This.Visible = TRUE
  Return 1
...
```

of_SetAlwaysRedraw

Description

Specifies whether the u_calendar instance forces calendar redraw when display properties change.

Access

Public

Syntax

dwcontrol.instancename.**of_SetAlwaysRedraw** (*boolean*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
boolean	Boolean indicating whether the u_calendar instance forces a redraw when display properties change (TRUE) or not (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

You should enable automatic redraw if your application programmatically changes a calendar's holidays, special days, or other display properties.

Examples

This example calls the of_SetAlwaysRedraw function:

```
dw_1.iuo_calendar.of_SetAlwaysRedraw(TRUE)
```

of_SetCloseOnClick

Description

Specifies whether to hide the u_calendar instance when the user clicks a date.

Access

Public

Syntax

dwcontrol.instancename.**of_SetCloseOnClick** (*boolean*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
boolean	Boolean specifying whether to hide the dropdown calendar when the user clicks a date (TRUE) or leave it displayed (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetCloseOnClick function:

```
this.of_SetTransObject(SQLCA)
this.of_SetDropDownCalendar(TRUE)
this.iuo_calendar.of_Register("start_date", &
  this.iuo_calendar.DDLB)
this.iuo_calendar.of_Register("termination_date")
```

```
this.iuo_calendar.of_Register("birth_date", &
  this.iuo_calendar.DDLB)
this.iuo_calendar.of_SetCloseOnClick(FALSE)
this.iuo_calendar.of_SetInitialValue(TRUE)
```

of_SetCloseOnDClick

Description

Specifies whether to hide the u_calendar instance when the user double-clicks a date.

Access

Public

Syntax

dwcontrol.instancename.**of_SetCloseOnDClick** (*boolean*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
boolean	Boolean specifying whether to hide the dropdown calendar when the user double-clicks a date (TRUE) or leave it displayed (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetCloseOnDClick function:

```
this.of_SetTransObject(SQLCA)
this.of_SetDropDownCalendar(TRUE)
this.iuo_calendar.of_Register("start_date")
this.iuo_calendar.of_Register("termination_date")
this.iuo_calendar.of_Register("birth_date")
this.iuo_calendar.of_SetCloseOnDClick(TRUE)
```

of_SetDate

Description

Sets the current date, optionally updating the date in the associated DataWindow column or EditMask control.

Access

Protected

Syntax

dwcontrol.instancename.**of_SetDate** (*date, sendtorequestor*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
date	Date variable containing the new date
sendtorequestor	Boolean indicating whether to copy *date* to the requestor object (TRUE) or not (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the of_Reset function:

```
...
id_resetdate = date(ls_date)
IF of_IsValid(id_resetdate) THEN
  of_SetDate(id_resetdate, FALSE)
ELSE
  of_SetDate(Today(), ib_InitialValue)
END IF
Return 1
```

of_SetDateFormat

Description

Sets the format for dates converted to strings.

Access

Public

Syntax

dwcontrol.instancename.**of_SetDateFormat** (*format*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
format	String specifying the format for dates converted to strings. Sample date formats include mm/dd/yy; m-d-yy; and mmm dd, yyyy

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

When using a calendar with an EditMask, this format must match the date mask.

Examples

This example calls the of_SetDateFormat function:

```
...
this.iuo_calendar.of_SetInitialValue(TRUE)
this.iuo_calendar.of_SetDateFormat("dd/mm/yy")
```

of_SetDropDown

Description

Enables or disables the dropdown service (n_cst_dropdown).

Access

Public

Syntax

dwcontrol.instancename.**of_SetDropDown** (*boolean*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
boolean	Boolean specifying whether to enable (TRUE) or disable (TRUE) the dropdown service

Return value

Integer. Returns 1 if the function succeeds, 0 if the dropdown service already exists, and -1 if an error occurs.

Examples

This example is from the u_calendar Constructor event:

```
...
this.Visible = FALSE
of_SetDropDown(TRUE)
...
```

of_SetFocusOnRequestor

Description

Returns focus to the associated DataWindow column or EditMask control.

Access

Protected

Syntax

dwcontrol.instancename.**of_SetFocusOnRequestor** ( )

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the dw_cal DoubleClicked event:

```
...
IF IsValid(inv_dropdown) THEN
  IF ib_closeondclick THEN
    of_SetFocusOnRequestor()
  END IF
END IF
```

of_SetHoliday

Description

Specifies the holidays displayed on the calendar.

Access

Public

Syntax

dwcontrol.instancename.**of_SetHoliday** (*holidays*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
holidays	Date array containing holidays

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetHoliday function:

```
Date ld_holidays[11]

ld_holidays[1] = 1997-01-01
ld_holidays[2] = 1997-02-17
```

```
ld_holidays[3] = 1997-04-21
ld_holidays[4] = 1997-05-26
ld_holidays[5] = 1997-07-04
ld_holidays[6] = 1997-09-01
ld_holidays[7] = 1997-10-13
ld_holidays[8] = 1997-11-27
ld_holidays[9] = 1997-11-28
ld_holidays[10] = 1997-12-25
ld_holidays[11] = 1997-12-26
this.iuo_calendar.of_SetHoliday(ld_holidays)
```

of_SetHolidayBold

Description

Sets the Bold property for holidays.

Access

Public

Syntax

dwcontrol.instancename.**of_SetHolidayBold** (*boolean*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
boolean	Boolean indicating whether holidays are bold (TRUE) or not (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetHolidayBold function:

```
...
this.iuo_calendar.of_SetHoliday(ld_holidays)
this.iuo_calendar.of_SetHolidayBold (TRUE)
this.iuo_calendar.of_SetHolidayColor(RGB(0,255,0))
```

of_SetHolidayColor

Description

Sets the display color for holidays.

Access

Public

Syntax

dwcontrol.instancename.**of_SetHolidayColor** (*color*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
color	Long specifying the color for holidays

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetHolidayColor function:

```
...
this.iuo_calendar.of_SetHoliday(ld_holidays)
this.iuo_calendar.of_SetHolidayBold(TRUE)
this.iuo_calendar.of_SetHolidayColor(RGB(0,255,0))
```

of_SetInitialValue

Description

Specifies whether the u_calendar instance initializes blank fields with the current date when the calendar displays.

Access

Public

Syntax

dwcontrol.instancename.**of_SetInitialValue** (*boolean*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
boolean	Boolean indicating whether the calendar initializes the blank fields with the current date when the calendar displays (TRUE) or not (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetInitialValue function:

```
...
this.iuo_calendar.of_SetInitialValue(TRUE)
...
```

of_SetMarkedDay

Description

Specifies a set of marked days. Marked days are not holidays but still require special display on the calendar.

Access

Public

Syntax

dwcontrol.instancename.**of_SetMarkedDay** (*markeddays*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
markeddays	Date array containing marked days

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetMarkedDay function:

```
Date ld_marked_days[12]

ld_marked_days[1] = 1996-06-13
ld_marked_days[2] = 1996-03-16
ld_marked_days[3] = 1996-09-23
ld_marked_days[4] = 1996-09-14
ld_marked_days[5] = 1997-06-13
ld_marked_days[6] = 1997-03-16
ld_marked_days[7] = 1997-09-23
ld_marked_days[8] = 1997-09-14
ld_marked_days[9] = 1998-06-13
ld_marked_days[10] = 1998-03-16
ld_marked_days[11] = 1998-09-23
ld_marked_days[12] = 1998-09-14

this.iuo_calendar.of_SetMarkedDay(ld_marked_days)
```

of_SetMarkedDayBold

Description

Sets the Bold property for marked days.

Access

Public

Syntax

dwcontrol.instancename.**of_SetMarkedDayBold** (*boolean*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
boolean	Boolean indicating whether marked days are bold (TRUE) or not (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetMarkedDayBold function:

```
...
this.iuo_calendar.of_SetMarkedDayBold(TRUE)
this.iuo_calendar.of_SetMarkedDayColor &
   (RGB(255, 0, 0))
```

of_SetMarkedDayColor

Description

Sets the display color for marked days.

Access

Public

Syntax

dwcontrol.instancename.**of_SetMarkedDayColor** (*color*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
color	Long specifying the color for marked days

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetMarkedDayColor function:

```
...
this.iuo_calendar.of_SetMarkedDayBold(TRUE)
this.iuo_calendar.of_SetMarkedDayColor &
  (RGB(255, 0, 0))
```

of_SetRequestor

Description

Associates an instance of u_calendar with a DataWindow or EditMask control.

Access

Public

Syntax

dwcontrol.instancename.**of_SetRequestor** (*requestor*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calculator with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
requestor	DragObject referencing the DataWindow or EditMask control to associate with this instance of u_calendar

Return value

Integer. Returns values as follows:

- 1 Success
- -1 An error occurred
- -2 The u_calendar instance has not enabled n_cst_dropdown
- -3 The EditMask mask type is not date

Usage

EditMask controls must have a mask type of date.

If you are not using the u_calendar object as a dropdown object (that is, displaying it permanently on a window, user object, or tab), call this function to associate the u_calendar instance with the EditMask control that displays the results.

Examples

This example is from the u_em of_SetDropDownCalendar function:

```
...
IF ab_switch THEN
  IF NOT IsValid (iuo_calendar) THEN
    lw_parent.OpenUserObject(iuo_calendar)
    iuo_calendar.of_SetRequestor(this)
    Return 1
  END IF
ELSE
  IF IsValid (iuo_calendar) THEN
    lw_parent.CloseUserObject(iuo_calendar)
    Return 1
  END IF
```

```
    END IF
    ...
```

of_SetSaturdayBold

Description — Sets the Bold property for Saturdays.

Access — Public

Syntax — *dwcontrol.instancename*.**of_SetSaturdayBold** (*boolean*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
boolean	Boolean indicating whether Saturdays are bold (TRUE) or not (FALSE)

Return value — Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples — This example calls the of_SetSaturdayBold function:

```
this.iuo_calendar.of_SetSaturdayBold TRUE)
this.iuo_calendar.of_SetSaturdayColor &
   (RGB(0, 255, 0))
```

of_SetSaturdayColor

Description — Sets the display color for Saturdays.

Access — Public

Syntax — *dwcontrol.instancename*.**of_SetSaturdayColor** (*color*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
color	Long specifying the color for Saturdays

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetSaturdayColor function:

```
this.iuo_calendar.of_SetSaturdayBold(TRUE)
this.iuo_calendar.of_SetSaturdayColor &
   (RGB(0, 255, 0))
```

of_SetSundayBold

Description

Sets the Bold property for Sundays.

Access

Public

Syntax

dwcontrol.instancename.**of_SetSundayBold** (*boolean*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
boolean	Boolean indicating whether Sundays are bold (TRUE) or not (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetSundayBold function:

```
this.iuo_calendar.of_SetSundayBold(TRUE)
this.iuo_calendar.of_SetSundayColor(RGB(0, 255, 0))
```

of_SetSundayColor

Description

Sets the display color for Sundays.

Access

Public

Syntax

dwcontrol.instancename.**of_SetSundayColor** (*color*)

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control (not required when using u_calendar with an EditMask control)

Argument	Description
instancename	Instance name of u_calendar (the u_dw and u_em default for this value is iuo_calendar)
color	Long specifying the color for Sundays

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetSundayColor function:

```
this.iuo_calendar.of_SetSundayBold(TRUE)
this.iuo_calendar.of_SetSundayColor(RGB(0, 255, 0))
```

of_UnRegister

Description

Removes one or all columns from the list of registered columns.

Access

Public

Syntax

dwcontrol.instancename.**of_Register** ({ *column* })

Argument	Description
dwcontrol	Instance name of the u_dw-based DataWindow control
instancename	Instance name of u_calendar (the u_dw default for this is iuo_calendar)
column	(Optional) String specifying the column to be unregistered. If you omit this argument, the function unregisters all columns

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_UnRegister function:

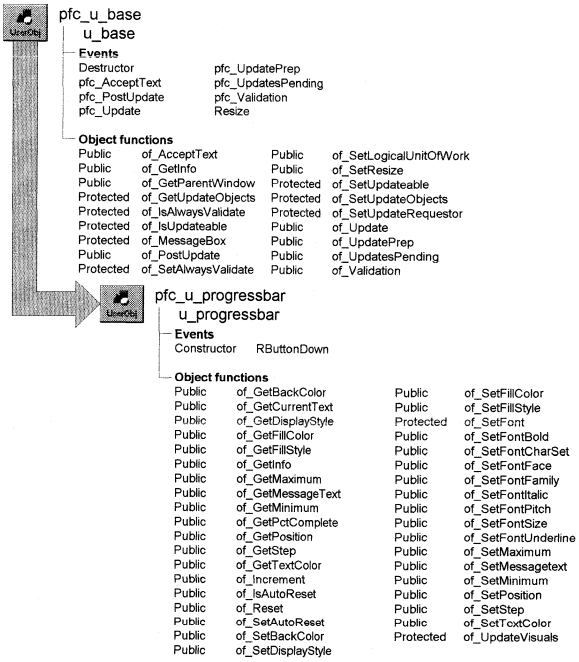
```
...
dw_emp.iuo_calendar.of_UnRegister("start_date")
...
```

u_progressbar

Description

Progress bar control. Use u_progressbar to provide users with a visual representation of percentage complete for long-running operations. U_progressbar can be vertical or horizontal and can display either percent complete or programmatically specified text.

Ancestry



Library

PFCAPSRV.PBL
PFEAPSRV.PBL

Usage

Use this control to display a progress bar for long-running operations.

To use u_progressbar:

1 Place an instance of the u_progressbar object on a window or user object.

2 Size the object to create an appropriately sized vertical or horizontal progress bar.

3 Define defaults for the progress bar instance. This example defines defaults in the Constructor event:

```
of_SetFillStyle(LEFTRIGHT)
```

```
of_SetDisplayStyle(PCTCOMPLETE)
of_SetFillColor(RGB(128, 128, 128))
of_SetFontSize(10)
of_SetAutoReset(FALSE)
```

4 In the process to be tracked, set the initial progress bar position, set the maximum, and call the of_Increment function at regular intervals to update the progress bar:

```
Integer li_count

uo_progress.of_SetMaximum(100)
uo_progress.of_SetPosition(0)
FOR li_count = 1 TO &
  uo_progress.of_GetMaximum() STEP 1
    uo_progress.of_Increment(1)
// Additional code here to perform your process
NEXT
```

See also n_cst_winsrv_statusbar

Instance variables

U_progressbar includes instance variables:

Instance variable	Description	Data type	Access	Usage
BAR	Constant set to 0	Integer	Public	Use with of_SetDisplayStyle
BOTTOMTOP	Constant set to 3	Integer	Public	Use with of_SetFillStyle
ib_autoreset	Controls autoreset	Boolean	Protected	Set with of_SetAutoReset. Default is TRUE
ii_displaystyle	Controls test displayed on progress bar	Integer	Protected	Set with of_DisplayStyle. Default is BAR
ii_fillstyle	Controls the direction in which the progress bar fills	Integer	Protected	Set with of_SetFillStyle. Default is LEFTRIGHT

Instance variable	Description	Data type	Access	Usage
ii_maximum	Value at which the incrementation process ends	Integer	Protected	Set with of_SetMaximum. Default is 100
ii_minimum	Value at which the incrementation process starts	Integer	Protected	Set with of_SetMinimum. Default is 0
ii_msgtextcount	Number of items in os_msgtext[] array	Integer	Protected	Internal
ii_percentcomplete	Current percent complete	Integer	Protected	Access with of_GetPctComplete
ii_position	Current increment position	Integer	Protected	Set with of_SetPosition. Updated internally
ii_step	Default increment value	Integer	Protected	Set with of_SetStep. Default is 10
il_backcolor	Background color	Long	Protected	Set with of_SetBackColor. Default is 78682240
il_fillcolor	Fill color	Long	Protected	Set with of_SetFillColor. Default is 16668075
il_textcolor	Text color	Long	Protected	Set with of_SetTextColor. Default is 0
is_msgtext[ ]	Array of text to display on progress bar	String	Protected	Set with of_SetMessageText
LEFTRIGHT	Constant set to 0	Integer	Public	Use with of_SetFillStyle
MSGTEXT	Constant set to 3	Integer	Public	Use with of_SetDisplayStyle
PCTCOMPLETE	Constant set to 1	Integer	Public	Use with of_SetDisplayStyle
POSITION	Constant set to 2	Integer	Public	Use with of_SetDisplayStyle
RIGHTLEFT	Constant set to 1	Integer	Public	Use with of_SetFillStyle

Instance variable	Description	Data type	Access	Usage
TOPDOWN	Constant set to 2	Integer	Public	Use with of_SetFillStyle

Events

U_progressbar includes precoded events:

Constructor
RButtonDown

Constructor

Description

Initializes progress bar settings.

Applies to U_progressbar

Usage

This event executes when the progress bar object is created.

RButtonDown

Description

Calls the parent's RButtonDown event.

Applies to Dw_progress

Usage

This event executes when the user right-clicks over the progress bar.

Functions

U_progressbar contains precoded object functions:

of_GetBackColor	of_SetFillColor
of_GetCurrentText	of_SetFillStyle
of_GetDisplayStyle	of_SetFont
of_GetFillColor	of_SetFontBold
of_GetFillStyle	of_SetFontCharSet
of_GetInfo	of_SetFontFace
of_GetMaximum	of_SetFontFamily
of_GetMessageText	of_SetFontItalic

of_GetMinimum	of_SetFontPitch
of_GetPctComplete	of_SetFontSize
of_GetPosition	of_SetFontUnderline
of_GetStep	of_SetMaximum
of_GetTextColor	of_SetMessageText
of_Increment	of_SetMinimum
of_IsAutoReset	of_SetPosition
of_Reset	of_SetStep
of_SetAutoReset	of_SetTextColor
of_SetBackColor	of_UpdateVisuals
of_SetDisplayStyle	

of_GetBackColor

Description

Retrieves the progress bar background color.

Access

Public

Syntax

instancename.**of_GetBackColor** ( )

Argument	Description
instancename	Instance name of u_progressbar

Return value

Long. Returns the background color.

Examples

This example calls the of_GetBackColor function:

```
Long ll_backcolor

ll_backcolor = uo_progress.of_GetBackColor()
MessageBox("Progress Bar", &
  "Background color property is " &
    + String(ll_backcolor))
```

of_GetCurrentText

Description

Retrieves the text currently displayed on the progress bar. This function retrieves any text displayed on the progress bar, not just the display text specified by the of_SetMessageText function.

Access

Public

Syntax

instancename.**of_GetCurrentText** ( )

Argument	Description
instancename	Instance name of u_progressbar

Return value

String. Returns the current progress bar text.

Examples

This example calls the of_GetCurrentText function:

```
String ls_text

ls_text = uo_progress.of_GetCurrentText()
MessageBox("Progress Bar", &
  "Current text is " + ls_text)
```

of_GetDisplayStyle

Description

Retrieves the progress bar display style. The display style controls the information that displays within the bar while tracking progress.

Access

Public

Syntax

instancename.**of_GetDisplayStyle ( )**

Argument	Description
instancename	Instance name of u_progressbar

Return value

Integer. Returns values as follows:

- **0 (BAR)** Progress bar only
- **1 (PCTCOMPLETE)** Progress bar and percent complete, including the percent sign
- **2 (POSITION)** Current increment value
- **3 (MSGTEXT)** Progress bar and text as specified by the of_SetMessageText function

Examples

This example calls the of_GetDisplayStyle function:

```
Integer li_displaystyle

li_displaystyle = uo_progress.of_GetDisplayStyle()
MessageBox("Progress Bar", &
  "DisplayStyle property is " &
    + String(li_displaystyle))
```

of_GetFillColor

Description

Retrieves the progress bar fill color.

Access

Public

Syntax

instancename.**of_GetFillColor** ( )

Argument	Description
instancename	Instance name of u_progressbar

Return value

Long. Returns the fill color.

Examples

This example calls the of_GetFillColor function:

```
Long ll_fillcolor

ll_fillcolor = uo_progress.of_GetFillColor()
MessageBox("Progress Bar", &
  "Fill color property is " &
    + String(ll_fillcolor))
```

of_GetFillStyle

Description

Retrieves the progress bar fill style.

Access

Public

Syntax

instancename.**of_GetFillStyle** ( )

Argument	Description
instancename	Instance name of u_progressbar

Return value

Integer. Returns values as follows:

- **0 (LEFTRIGHT)** The progress bar fills from left to right
- **1 (RIGHTLEFT)** The progress bar fills from right to left
- **2 (TOPDOWN)** The progress bar fills from top to bottom
- **3 (BOTTOMUP)** The progress bar fills from bottom to top

Examples

This example calls the of_GetFillStyle function:

```
Integer li_fillstyle
```

```
li_fillstyle = uo_progress.of_GetFillStyle()
MessageBox("Progress Bar", &
  "Fill style is "  + String(li_fillstyle))
```

of_GetInfo

Description

Retrieves object information.

Access

Public

Syntax

dwcontrol.instancename.**of_GetInfo** (*infoobject*)

Argument	Description
instancename	Instance name of u_progressbar
infoobject	N_cst_infoattrib instance into which the function places information (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

The DataWindow Properties window calls this function to access service information:

Examples

This example calls the of_GetInfo function:

```
n_cst_infoattrib  lnv_info

uo_progress.of_GetInfo(lnv_info)
MessageBox("Info",  &
  "Description: " + lnv_info.is_description  &
    + ". Name: " + lnv_info.is_name)
```

of_GetMaximum

Description

Retrieves the maximum value used by the progress bar incrementation process.

Access

Public

Syntax

instancename.**of_GetMaximum** ( )

Argument	Description
instancename	Instance name of u_progressbar

Return value

Integer. Returns the maximum increment value for the progress bar.

Examples

This example calls the of_GetMaximum function:

```
uo_progress.of_SetPosition(0)
FOR li_count = 1 TO &
  uo_progress.of_GetMaximum() STEP 1
  uo_progress.of_Increment(1)
  // Additional code here to perform your process
NEXT
```

of_GetMessageText

Description

Retrieves the progress messages specified in the of_SetMessageText function.

Access

Public

Syntax

instancename.**of_GetMessageText** (*text*)

Argument	Description
instancename	Instance name of u_progressbar
text	String array into which the function places the progress messages (passed by reference)

Return value

Integer. Returns the number of elements in the *text* array.

Examples

This example calls the of_GetMessageText function:

```
String ls_text[], ls_all
Integer li_size, li_count

li_size = uo_progress.of_GetMessageText(ls_text)

FOR li_count = 1 to li_size STEP 1
  ls_all += ls_text[li_count] + "~r~n"
NEXT
MessageBox("Progress Bar", &
  "Progress messages are: " + ls_all)
```

of_GetMinimum

Description

Retrieves the minimum value used by the incrementation process. This value is the baseline from which the incrementation process begins.

Access

Public

Syntax

instancename.**of_GetMinimum ( )**

Argument	Description
instancename	Instance name of u_progressbar

Return value

Integer. Returns the minimum value for the incrementation process.

Examples

This example calls the of_GetMinimum function:

```
Integer li_min

li_min = uo_progress.of_GetMinimum )
MessageBox("Progress Bar", &
  "Minimum value is " &
    + String(li_min))
```

of_GetPctComplete

Description

Retrieves the progress bar's current percent complete.

Access

Public

Syntax

instancename.**of_GetPctComplete ( )**

Argument	Description
instancename	Instance name of u_progressbar

Return value

Integer. Returns the current percent complete for the progress bar.

Examples

This example calls the of_GetPctComplete function:

```
Integer li_count

FOR li_count = 1 TO &
  uo_progress.of_GetMaximum() STEP 1
  uo_progress.of_Increment(1)
  gnv_app.of_GetFrame().SetMicroHelp &
  ("Count is: " + String(li_count) + &
    " Pct Complete is: " + &
      String(uo_progress.of_GetPctComplete()))
  // Additional code here to perform your process
NEXT
```

of_GetPosition

Description

Retrieves the progress bar's current increment value.

Access

Public

Syntax

instancename.**of_GetPosition** ( )

Argument	Description
instancename	Instance name of u_progressbar

Return value

Integer. Returns the current increment value for the progress bar.

Examples

This example calls the of_GetPosition function:

```
Integer li_count

FOR li_count = 1 TO &
  uo_progress.of_GetMaximum() STEP 1
  uo_progress.of_Increment(1)
  gnv_app.of_GetFrame().SetMicroHelp &
    ("Count is: " + &
      String(uo_progress.of_GetPosition()) + &
        " Pct Complete is: " + &
          String(uo_progress.of_GetPctComplete()))
  // Additional code here to perform your process
NEXT
```

of_GetStep

Description

Retrieves the step value the progress bar uses during a default increment process.

Access

Public

Syntax

instancename.**of_GetStep** ( )

Argument	Description
instancename	Instance name of u_progressbar

Return value

Integer. Returns the step value.

Examples

This example calls the of_GetStep function:

```
Integer li_step
```

```
li_step = uo_progress.of_GetStep()
MessageBox("Progress Bar", &
  "Step value is " + String(li_step))
```

of_GetTextColor

Description: Retrieves the text color for the progress bar.

Access: Public

Syntax: *instancename*.**of_GetTextColor** ( )

Argument	Description
instancename	Instance name of u_progressbar

Return value: Long. Returns the progress bar text color.

Examples: This example calls the of_GetTextColor function:

```
Long ll_color

ll_color = uo_progress.of_GetTextColor()
MessageBox("Progress Bar", &
  "Current text color is " + String(ll_color))
```

of_Increment

Description: Increments progress by either the default increment value or by a specified value.

Access: Public

Syntax: *instancename*.**of_Increment** ({ *incrementvalue* })

Argument	Description
instancename	Instance name of u_progressbar
incrementvalue (optional)	Integer specifying the increment value. The default increment value is 10, which you can modify by calling the of_SetStep function

Return value: Integer. Returns the new increment value.

Examples

This example calls the of_Increment function:

```
Integer li_count

uo_progress.of_SetPosition(0)
FOR li_count = 1 TO &
uo_progress.of_GetMaximum() STEP 1
uo_progress.of_Increment(1)
// Additional code here to perform your process
NEXT
```

of_IsAutoReset

Description

Reports whether the progress bar returns to zero after it reaches 100%.

Access

Public

Syntax

instancename.**of_IsAutoReset ( )**

Argument	Description
instancename	Instance name of u_progressbar

Return value

Boolean. Returns TRUE if u_progressbar clears the progress bar when it reaches 100% and FALSE if it does not.

Examples

This example calls the of_IsAutoReset function:

```
n_cst_conversion lnv_conversion
Boolean lb_autoreset

lb_autoreset = uo_progress.of_IsAutoReset()
MessageBox("Progress Bar", &
  "Autoreset property is " &
    + lnv_conversion.of_String(lb_autoreset))
```

of_Reset

Description

Clears the progress bar, setting the position to the minimum and percent complete to zero.

Access

Public

Syntax

instancename.**of_Reset ( )**

Argument	Description
instancename	Instance name of u_progressbar

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example is from the of_SetPosition function:

```
...
IF ldc_completion >= 1 and ib_autoreset THEN
this.Post Function of_Reset()
END IF
...
```

of_SetAutoReset

Description

Specifies whether to clear the progress bar when it reaches 100%. The progress bar reaches 100% when the incrementation process reaches the maximum value.

Access

Public

Syntax

instancename.**of_SetAutoReset** (*boolean*)

Argument	Description
instancename	Instance name of u_progressbar
boolean	Boolean indicating whether to clear the status bar when it reaches 100% (TRUE) or leave it visible (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetAutoReset function:

```
of_SetFillStyle(BOTTOMUP)
of_SetDisplayStyle(PCTCOMPLETE)
of_SetAutoReset(FALSE)
```

of_SetBackColor

Description

Sets the background color of the progress bar.

Access

Public

Syntax

instancename.**of_SetBackColor** (*color*)

Argument	Description
instancename	Instance name of u_progressbar
color	Long specifying the progress bar fill color. The default is 78682240 (light gray)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example from the Constructor event of a u_progressbar instance calls the of_SetBackColor function:

```
of_SetFillStyle(BOTTOMUP)
of_SetDisplayStyle(PCTCOMPLETE)
of_SetBackColor (RGB(255, 0, 0)) // Red
of_SetAutoReset(FALSE)
```

of_SetDisplayStyle

Description

Sets a display style for the progress bar. The display style controls the information that displays within the bar while tracking progress.

Access

Public

Syntax

instancename.**of_SetDisplayStyle** (*displaystyle*)

Argument	Description
instancename	Instance name of u_progressbar
displaystyle	Integer or u_progressbar constant specifying the progress bar display style: ♦ **0 or BAR (default)** Bar only, no text ♦ **1 or PCTCOMPLETE** Bar displays percent complete, including the percent sign ♦ **2 or POSITION** Bar displays the current increment value ♦ **3 or MSGTEXT** Bar displays text, as specified by the of_SetMessageText function

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

If you use MSGTEXT, you must also call the of_SetMessageText function to specify the text displayed in the status bar.

Examples

This example from the Constructor event of a u_progressbar instance calls the of_SetDisplayStyle function:

```
of_SetFillStyle(BOTTOMUP)
of_SetDisplayStyle(PCTCOMPLETE)
```

of_SetFillColor

Description

Sets the fill color of the progress bar.

Access

Public

Syntax

instancename.**of_SetFillColor** (*color*)

Argument	Description
instancename	Instance name of u_progressbar
color	Long specifying the progress bar fill color. The default is 16668075 (blue)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example from the Constructor event of a u_progressbar instance calls the of_SetFillColor function:

```
of_SetFillStyle(BOTTOMUP)
of_SetDisplayStyle(PCTCOMPLETE)
of_SetFillColor (RGB(128, 128, 128))
```

of_SetFillStyle

Description

Sets a fill style for the progress bar.

Access

Public

Syntax

instancename.**of_SetFillStyle** (*style*)

Argument	Description
instancename	Instance name of u_progressbar

Argument	Description
style	Integer or u_progressbar constant specifying the fill style: ♦ **0 or TOPDOWN** The bar fills from top to bottom ♦ **1 or BOTTOMUP** The bar fills from bottom to top ♦ **2 or LEFTRIGHT (default)** The bar fills from left to right ♦ **3 or RIGHTLEFT** The bar fills from right to left

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetFillStyle function:

```
of_SetFillStyle (BOTTOMUP)
of_SetDisplayStyle(PCTCOMPLETE)
of_SetFillColor(RGB(128, 128, 128))
```

of_SetFont

Description

Changes the font displayed in the progress bar.

Access

Protected

Syntax

instancename.**of_SetFont** (*fontface, fontsize, fontcharset, fontfamily, fontpitch, bold, italic, underline*)

Argument	Description
instancename	Instance name of u_progressbar
fontface	String specifying the new font. If you pass a NULL, the function does not change the font
fontsize	Integer specifying the point size of the new font. If you pass a NULL, the function does not change the font size
fontcharset	Integer specifying the character set: ♦ 0–ANSI ♦ 1–The default character set for the specified font ♦ 2–Symbol ♦ 3–Shift JIS ♦ 4–OEM If you pass a NULL, the function does not change the font character set

Argument	Description
fontfamily	FontFamily enumerated data type specifying the font family: ♦ AnyFont! ♦ Roman! ♦ Swiss! ♦ Modern! ♦ Script! ♦ Decorative! If you pass a NULL, the function does not change the font family
fontpitch	FontPitch enumerated data type specifying the pitch of the new font: ♦ Default! ♦ Fixed! ♦ Variable! If you pass a NULL, the function does not change the pitch
bold	Boolean specifying whether text is bold: ♦ TRUE—Bold ♦ FALSE—Not bold (default) If you pass a NULL, the function does not change the Bold property
italic	Boolean specifying whether text is italic: ♦ TRUE—Italic ♦ FALSE—Not italic (default) If you pass a NULL, the function does not change the italic property
underline	Boolean specifying whether text is underlined: ♦ TRUE—Underlined ♦ FALSE—Not underlined (default) If you pass a NULL, the function does not change the underline property

Return value

String. Returns the result of the Modify PowerScript function that changes the font; an empty string indicates success.

Usage

Internal.

Examples

This example is from the of_SetFontBold function:

```
String ls_fontface
Long ll_fontsize
FontFamilylff_fontfamily
FontPitchlfp_fontpitch
Boolean lb_bold, lb_italic, lb_underline

SetNull(ls_fontface)
SetNull(ll_fontsize)
SetNull(lff_fontfamily)
SetNull(lfp_fontpitch)
SetNull(lb_bold)
SetNull(lb_italic)
SetNull(lb_underline)

// ab_bold is the argument to of_SetBold
Return of_SetFont(ls_fontface, ll_fontsize, &
  lff_fontfamily, lfp_fontpitch, ab_bold, &
    lb_italic, lb_underline)
```

of_SetFontBold

Description

Sets the Bold property for progress bar text.

Access

Public

Syntax

instancename.**of_SetFontBold** (*boolean*)

Argument	Description
instancename	Instance name of u_progressbar
boolean	Boolean specifying whether progress bar text is bold (TRUE) or not (FALSE)

Return value

String. Returns the result of the Modify PowerScript function that changes the Bold property; an empty string indicates success.

Examples

This example calls the of_SetFontBold function:

```
of_SetFillStyle(BOTTOMUP)
of_SetDisplayStyle(PCTCOMPLETE)
of_SetFillColor(RGB(128, 128, 128))
of_SetFontBold(TRUE)
```

of_SetFontCharSet

Description

Sets the CharSet property for progress bar text.

Access

Public

Syntax

instancename.**of_SetFontCharSet** (*fontcharset*)

Argument	Description
instancename	Instance name of u_progressbar
fontcharset	Integer specifying the character set: ♦ 0–ANSI ♦ 1–The default character set for the specified font ♦ 2–Symbol ♦ 3–Shift JIS ♦ 4–OEM

Return value

String. Returns the result of the Modify PowerScript function that changes the font face; an empty string indicates success.

Examples

This example calls the of_SetFontCharSet function:

```
of_SetFillStyle(BOTTOMUP)
of_SetDisplayStyle(PCTCOMPLETE)
of_SetFillColor(RGB(128, 128, 128))
of_SetFontCharSet(1)
of_SetFontFamily(Script!)
of_SetFontSize(12)
```

of_SetFontFace

Description

Sets the Face (typeface) property for progress bar text.

Access

Public

Syntax

instancename.**of_SetFontFace** (*fontface*)

Argument	Description
instancename	Instance name of u_progressbar
fontface	String specifying the font face (such as Arial)

Return value

String. Returns the result of the Modify PowerScript function that changes the font face; an empty string indicates success.

Examples

This example from the Constructor event of a u_progressbar instance calls the of_SetFontFace function:

```
of_SetFillStyle(BOTTOMUP)
of_SetDisplayStyle(PCTCOMPLETE)
of_SetFillColor(RGB(128, 128, 128))
of_SetFontFace("Monotype Corsiva")
of_SetFontFamily(Script!)
```

of_SetFontFamily

Description

Sets the font Family property (such as Roman) for progress bar text.

Access

Public

Syntax

instancename.**of_SetFontFamily** (*fontfamily*)

Argument	Description
instancename	Instance name of u_progressbar
fontfamily	FontFamily enumerated data type specifying the font family: ♦ AnyFont! ♦ Roman! ♦ Swiss! ♦ Modern! ♦ Script! ♦ Decorative!

Return value

String. Returns the result of the Modify PowerScript function that changes the font family; an empty string indicates success.

Examples

This example calls the of_SetFontFamily function:

```
of_SetFillStyle(BOTTOMUP)
of_SetDisplayStyle(PCTCOMPLETE)
of_SetFillColor(RGB(128, 128, 128))
of_SetFontFace("Monotype Corsiva")
of_SetFontFamily(Script!)
```

of_SetFontItalic

Description

Sets the Italic property for progress bar text.

Access

Public

Syntax

instancename.**of_SetFontItalic** (*boolean*)

Argument	Description
instancename	Instance name of u_progressbar
boolean	Boolean specifying whether progress bar text is italic (TRUE) or not (FALSE)

Return value

String. Returns the result of the Modify PowerScript function that changes the Italic property; an empty string indicates success.

Examples

This example calls the of_SetFontItalic function:

```
of_SetFillStyle(BOTTOMUP)
of_SetDisplayStyle(PCTCOMPLETE)
of_SetFillColor(RGB(128, 128, 128))
of_SetFontItalic (TRUE)
```

of_SetFontPitch

Description

Sets the Pitch property for progress bar text.

Access

Public

Syntax

instancename.**of_SetFontPitch** (*fontpitch*)

Argument	Description
instancename	Instance name of u_progressbar
fontpitch	FontPitch enumerated data type specifying the font pitch: ♦ Default! ♦ Fixed! ♦ Variable!

Return value

String. Returns the result of the Modify PowerScript function that changes the font pitch; an empty string indicates success.

Examples

This example calls the of_SetFontPitch function:

```
of_SetFillStyle(BOTTOMUP)
```

```
of_SetDisplayStyle(PCTCOMPLETE)
of_SetFillColor(RGB(128, 128, 128))
of_SetFontPitch(Variable!)
```

of_SetFontSize

Description: Sets the Size property for progress bar text.

Access: Public

Syntax: *instancename*.**of_SetFontSize** (*fontsize*)

Argument	Description
instancename	Instance name of u_progressbar
fontsize	Long specifying the font size, in points, for the progress bar text. Do not specify a negative number for this value

Return value: String. Returns the result of the Modify PowerScript function that changes the font size; an empty string indicates success.

Examples: This example calls the of_SetFontSize function:

```
of_SetFillStyle(BOTTOMUP)
of_SetDisplayStyle(PCTCOMPLETE)
of_SetFontSize(10)
```

of_SetFontUnderline

Description: Sets the Underline property for progress bar text.

Access: Public

Syntax: *instancename*.**of_SetFontUnderline** (*boolean*)

Argument	Description
instancename	Instance name of u_progressbar
boolean	Boolean specifying whether progress bar text is underlined (TRUE) or not (FALSE)

Return value: String. Returns the result of the Modify PowerScript function that changes the underline property; an empty string indicates success.

Examples: This example calls the of_SetFontUnderline function:

```
of_SetFillStyle(BOTTOMUP)
of_SetDisplayStyle(PCTCOMPLETE)
of_SetFontUnderline (TRUE)
```

of_SetMaximum

Description

Sets the maximum increment value for the progress bar (also used as the basis for the progress bar's percent complete calculation).

Access

Public

Syntax

instancename.**of_SetMaximum** (*maximum*)

Argument	Description
instancename	Instance name of u_progressbar
maximum	Integer specifying the maximum increment value.

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

The default for this value is 100. The of_SetPosition function uses this value as the dividend in its percent complete calculation (percent complete = current position/maximum * 100).

Examples

This example calls the of_SetMaximum function:

```
SELECT COUNT(emp_id)
  INTO :il_max
  FROM Employee
  USING SQLCA ;
IF il_max > 0 THEN
  uo_progress.of_SetMaximum(il_max)
ELSE
  MessageBox("SQL error", &
    "SELECT COUNT failed~r~n" + SQLCA.SQLERRTEXT)
END IF
```

of_SetMessageText

Description

Sets the text strings that display at regular intervals when the progress bar uses the message text display style.

Access

Public

Syntax

instancename.**of_SetMessageText** (*text*)

Argument	Description
instancename	Instance name of u_progressbar
text	String array whose elements contain the values to display on the progress bar

Return value

Integer. Returns the number of elements in *text* if the function succeeds and -1 if an error occurs.

Usage

You typically call this function after calling of_SetDisplayStyle(MSGTEXT).

Examples

This example calls the of_SetMessageText function:

```
String ls_msgtext[ ] = {"Ten", "Twenty", &
  "Thirty", "Forty", "Fifty", "Sixty", &
    "Seventy", "Eighty", "Ninety", "One Hundred"}

of_SetDisplayStyle(MSGTEXT)
of_SetMessageText(ls_msgtext)
```

of_SetMinimum

Description

Sets the initial increment value for the progress bar.

Access

Public

Syntax

instancename.**of_SetMinimum** (*minimum*)

Argument	Description
instancename	Instance name of u_progressbar
minimum	Integer specifying the initial increment value for the progress bar

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

The of_SetPosition and of_Reset functions use the minimum to determine the initial position value.

Examples

This example calls the of_SetMinimum function:

```
of_SetFillStyle(BOTTOMUP)
of_SetDisplayStyle(PCTCOMPLETE)
of_SetMinimum(0)
```

of_SetPosition

Description — Sets the progress bar's current position and updates percent complete.

Access — Public

Syntax — *instancename*.**of_SetPosition** (*position*)

Argument	Description
instancename	Instance name of u_progressbar
position	Integer specifying the progress bar's position

Return value — Integer. Returns the current increment value if the function succeeds and -1 if an error occurs.

Usage — Call this function to establish an initial position for the progress bar before beginning an incrementation process.

If *position* is less than the minimum (the default minimum is 0), this function sets the position to the minimum. If *position* is greater than the maximum (the default maximum is 100), this function sets the position to the maximum. You modify the default minimum and maximum by calling the of_SetMinimum and of_SetMaximum functions.

The of_Increment function calls this function to update the current position and percent complete.

Examples — This example calls the of_SetPosition function:

```
Integer li_count

uo_progress.of_SetPosition(0)
For li_count = 1 to 100 STEP 1
  uo_progress.of_Increment(1)
NEXT
```

of_SetStep

Description — Sets the step value the progress bar uses during a default increment process.

Access — Public

Syntax — *instancename*.**of_SetStep** (*stepvalue*)

Argument	Description
instancename	Instance name of u_progressbar
stepvalue	Integer specifying the default increment value

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

The progress bar uses this value as the default when you call of_Increment with no arguments.

Examples

This example calls the of_SetStep function:

```
of_SetFillStyle(BOTTOMUP)
of_SetDisplayStyle(PCTCOMPLETE)
of_SetStep(5)
```

of_SetTextColor

Description

Sets the color of text in the progress bar.

Access

Public

Syntax

instancename.**of_SetTextColor** (*color*)

Argument	Description
instancename	Instance name of u_progressbar
color	Long specifying the progress bar text color. The default is 0 (black)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example from the Constructor event of a u_progressbar instance calls the of_SetTextColor function:

```
of_SetFillStyle(BOTTOMUP)
of_SetDisplayStyle(PCTCOMPLETE)
of_SetTextColor(RGB(255, 255, 255))
```

of_UpdateVisuals

Description

Resizes the progress bar and updates the display text.

Access

Protected

Syntax

instancename.**of_UpdateVisuals** (*completionvalue*)

Argument	Description
instancename	Instance name of u_progressbar
completionvalue	Decimal specifying the completion value of the progress bar

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the of_SetPosition function:

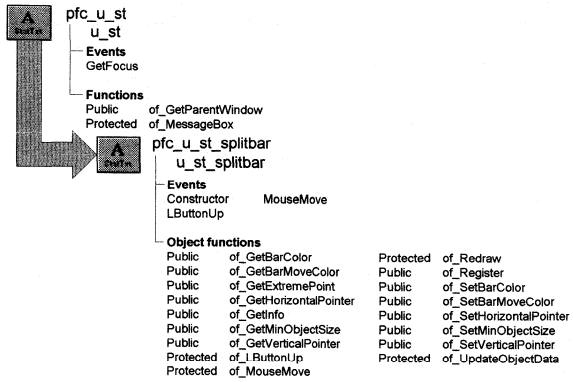
```
...
li_rc = of_UpdateVisuals (ldc_completion)
dw_progress.SetRedraw(true)
...
```

u_st_splitbar

Description

Splitbar service. Use this control to add splitbar functionality to two or more controls in a window.

Ancestry



Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use this object to add a splitbar to a window. To use this object:

1 Place the object on a window between two or more controls.

2 (Optional) If you are using the window resize service, register the splitbar:

```
this.of_SetResize(TRUE)
this.inv_resize.of_Register &
  (tv_deptemp, 0, 0, 0, 100)
this.inv_resize.of_Register &
  (st_splitbar, 0, 0, 0, 100)
this.inv_resize.of_Register &
  (lv_empdetail, 0, 0, 0, 100)
```

3 Move and resize the splitbar object and the surrounding objects until they relate appropriately. For example, a vertical splitbar between two objects should have the same height as the surrounding objects.

4 Add code to the splitbar control's Constructor event to register the controls that resize when the user moves the splitbar:

```
this.of_Register(tv_1, LEFT
this.of_Register(lv_1, RIGHT)
this.of_SetBarColor(RGB(192, 192, 192))
```

See also n_cst_resize

Instance variables

U_st_splitbar has instance variables:

Name	Description	Data type	Access	Usage
ABOVE	Constant set to 3	Integer	Public	Use with of_Register
BELOW	Constant set to 4	Integer	Public	Use with of_Register
BOTTOMMOST	Constant set to 4	Integer	Protected	Internal
HORIZONTAL	Constant set to 2	Integer	Public	Internal
ib_performredraw	Controls redraw	Boolean	Protected	Internal
idrg_lefttop[ ]	Upper-left coordinate for all registered controls	DragObject	Protected	Internal
idwg_rightbottom[ ]	Lower-right coordinate for all registered controls	DragObject	Protected	Internal
ii_barwidth	Thickness of splitbar	Integer	Protected	Internal
ii_lefttopbound	Upper-left coordinate	Integer	Protected	Internal
ii_minobjectsize	Minimum object size	Integer	Protected	Set with of_SetMinObject Size
ii_prevpositionx	Previous x coordinate	Integer	Protected	Internal
ii_prevpositiony	Previous y coordinate	Integer	Protected	Internal
ii_bottomrightbound	Bottom right coordinate	Integer	Protected	Internal
ii_rounding	Rounding factor	Integer	Protected	Internal
ii_style	Splitbar style (horizontal or vertical)	Integer	Protected	Set with of_SetStyle
il_barcolor	Splitbar color	Long	Protected	Set with of_SetBarColor

Name	Description	Data type	Access	Usage
il_barmovecolor	Splitbar color when it is being moved	Long	Protected	Set with of_SetBarMoveColor
ir_lefttopheight[ ]	Left top height for all registered controls	Real	Protected	Internal
ir_lefttopwidth[ ]	Left top width for all registered controls	Real	Protected	Internal
ir_lefttopx[ ]	Top left x coordinate for all registered controls	Real	Protected	Internal
ir_lefttopy[ ]	Top left y coordinate for all registered controls	Real	Protected	Internal
ir_rightbottomheight[ ]	Bottom right height for all registered controls	Real	Protected	Internal
ir_rightbottomwidth[ ]	Bottom right width for all registered controls	Real	Protected	Internal
ir_rightbottomx[ ]	Bottom right x coordinate for all registered controls	Real	Protected	Internal
ir_rightbottomy[ ]	Bottom right y coordinate for all registered controls	Real	Protected	Internal
is_horizontalpointer	Icon name for the pointer that displays when over a horizontal splitbar	String	Protected	Set with of_SetHorizontalPointer (default is SizeNS!)
is_verticalpointer	Icon name for the pointer that displays when over a vertical splitbar	String	Protected	Set with of_SetHorizontalPointer (default is SizeWE!)
itab_parent	References the Tab control that contains the splitbar	Tab	Protected	Internal
iuo_parent	References the custom standard user object that contains the splitbar	UserObject	Protected	Internal
iw_parent	References the window that contains the splitbar	Window	Protected	Internal
LEFT	Constant set to 1	Integer	Public	Use with of_Register

Name	Description	Data type	Access	Usage
LEFTMOST	Constant set to 1	Integer	Protected	Internal
RIGHT	Constant set to 2	Integer	Public	Use with of_Register
RIGHTMOST	Constant set to 2	Integer	Protected	Internal
TOPMOST	Constant set to 3	Integer	Protected	Internal
VERTICAL	Constant set to 1	Integer	Public	Vertical

Events

U_st_splitbar includes precoded events:

Constructor	MouseMove
LButtonUp	

Constructor

Description

Automatically calculates the bar orientation (vertical or horizontal) and initializes other status-bar settings.

Usage

Add code to this event to register the controls that resize when the user moves the splitbar.

Examples

This example shows code you might add to the Constructor event:

```
this.of_Register(tv_1, LEFT)
this.of_Register(lv_1, RIGHT)
this.of_SetBarColor(RGB(192, 192, 192))
```

LButtonUp

Description

Calls the of_LButtonUp function.

Usage

PFC uses this event to resize objects when the user moves the splitbar.

MouseMove

Description

Calls the of_MouseMove function.

Usage

If the user presses the left mouse button, PFC uses this event to move the splitbar, refresh the location, and redraw if necessary.

Functions

U_st_splitbar contains precoded object functions:

of_GetBarColor	of_Redraw
of_GetBarMoveColor	of_Register
of_GetExtremePoint	of_SetBarColor
of_GetHorizontalPointer	of_SetBarMoveColor
of_GetInfo	of_SetHorizontalPointer
of_GetMinObjectSize	of_SetMinObjectSize
of_GetVerticalPointer	of_SetVerticalPointer
of_LButtonUp	of_UpdateObjectData
of_MouseMove	

of_GetBarColor

Description

Retrieves the splitbar color.

Access

Public

Syntax

instancename.**of_GetBarColor** ( )

Argument	Description
instancename	Instance name of u_st_splitbar

Return value

Long. Returns the splitbar color.

Examples

This example calls the of_GetBarColor function:

```
Long ll_color

ll_color = st_splitbar.of_GetBarColor()
```

of_GetBarMoveColor

Description: Retrieves the color that displays when the splitbar is being moved.

Access: Public

Syntax: *instancename*.**of_GetBarMoveColor** ( )

Argument	Description
instancename	Instance name of u_st_splitbar

Return value: Long. Returns the splitbar move color.

Examples: This example calls the of_GetBarMoveColor function:

```
Long ll_color

ll_color = st_splitbar.of_GetBarMoveColor()
```

of_GetExtremePoint

Description: Retrieves the specified extreme point coordinate.

Access: Public

Syntax: *instancename*.**of_GetExtremePoint** (*extremetype*)

Argument	Description
instancename	Instance name of u_st_splitbar
extremetype	Integer or u_st_splitbar constant that specifies the extreme point to be returned: ♦ **1 or LEFTMOST** The extreme point to the left ♦ **2 or RIGHTMOST** The extreme point to the right ♦ **3 or TOPMOST** The extreme point at the top ♦ **4 or BOTTOMMOST** The extreme point at the bottom

Return value: Integer. Returns the coordinate of the specified extreme point.

Usage: Internal.

Examples: This example is from the of_LButtonUp function:

```
...
IF ii_style = HORIZONTAL THEN
```

```
li_miny = of_GetExtremePoint(TOPMOST)
li_maxy = of_GetExtremePoint(BOTTOMMOST)
...
```

of_GetHorizontalPointer

Description

Retrieves the name of the pointer that displays when the cursor is over a horizontal splitbar.

Access

Public

Syntax

instancename.**of_GetHorizontalPointer** ( )

Argument	Description
instancename	Instance name of u_st_splitbar

Return value

String. Returns the name of the pointer that displays when the cursor is over a horizontal splitbar.

Examples

This example calls the of_GetHorizontalPointer function:

```
String ls_pointer

ls_pointer = st_1.of_GetHorizontalPointer()
MessageBox("Splitbar", &
  "Horizontal pointer is " + ls_pointer)
```

of_GetInfo

Description

Retrieves object information.

Access

Public

Syntax

dwcontrol.instancename.**of_GetInfo** (*infoobject*)

Argument	Description
instancename	Instance name of u_st_splitbar
infoobject	N_cst_infoattrib instance into which the function places information (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

The DataWindow Properties window calls this function to access service information:

Examples

This example calls the of_GetInfo function:

```
n_cst_infoattrib  lnv_info

uo_splitbar.of_GetInfo(lnv_info)
MessageBox("Info",  &
  "Description: " + lnv_info.is_description  &
    + ". Name: " + lnv_info.is_name)
```

of_GetMinObjectSize

Description

Retrieves the minimum size (in PBUs) for registered objects.

Access

Public

Syntax

instancename.**of_GetMinObjectSize** ( )

Argument	Description
instancename	Instance name of u_st_splitbar

Return value

Integer. Returns the minimum size for registered objects.

Examples

This example calls the of_GetMinObjectSize function:

```
Integer li_minsize

li_minsize = st_1.of_GetMinObjectSize()
MessageBox("Splitbar", &
  "Minimum size is " + String(li_minsize))
```

of_GetVerticalPointer

Description

Retrieves the name of the pointer that displays when the cursor is over a vertical splitbar.

Access

Public

Syntax

instancename.**of_GetVerticalPointer** ( )

Argument	Description
instancename	Instance name of u_st_splitbar

Return value

String. Returns the name of the pointer that displays when the cursor is over a vertical splitbar.

Examples

This example calls the of_GetVerticalPointer function:

```
String ls_pointer

ls_pointer = st_1.of_GetVerticalPointer()
MessageBox("Splitbar", &
  "Vertical pointer is " + ls_pointer)
```

of_LButtonUp

Description

Resizes registered objects when the user releases the left mouse button.

Access

Protected

Syntax

instancename.**of_LButtonUp** (*flags*, *x*, *y*)

Argument	Description
instancename	Instance name of u_st_splitbar
flags	UnsignedLong indicating the modifier keys and mouse buttons that were pressed
x	Integer indicating the x position of the mouse click
y	Integer indicating the y position of the mouse click

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the LButtonUp event:

```
of_LButtonUp(flags, xpos, ypos)
```

of_MouseMove

Description

Refreshes the screen as the user moves the splitbar.

Access

Protected

Syntax

instancename.**of_MouseMove** (*flags*, *x*, *y*)

Argument	Description
instancename	Instance name of u_st_splitbar
flags	UnsignedLong indicating the modifier keys and mouse buttons that are currently pressed
x	Integer indicating the x position of the mouse
y	Integer indicating the y position of the mouse

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the MouseMove event:

```
IF NOT KeyDown(keyLeftButton!) THEN
  Return
END IF
of_MouseMove(flags, xpos, ypos)
```

of_Redraw

Description

Redraws objects adjacent to the splitbar.

Access

Protected

Syntax

instancename.**of_Redraw** (*x*, *width*, *y*, *height*)

Argument	Description
instancename	Instance name of u_st_splitbar
x	Integer specifying the splitbar's previous x coordinate
width	Integer specifying the splitbar's previous width
y	Integer specifying the splitbar's previous y coordinate
height	Integer specifying the splitbar's previous height

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the of_MouseMove function:

```
. . .
```

```
IF ii_style = HORIZONTAL THEN
  this.Y = li_pointery
  this.Height = ii_barwidth
  IF ib_performredraw THEN
    IF (li_prevy <> this.Y) THEN
      of_Redraw(li_prevx, li_prevwidth, &
        li_prevy, li_prevheight)
    END IF
  END IF
...
```

of_Register

Description

Registers the controls to be moved and resized when the user moves the splitbar.

Access

Public

Syntax

instancename.**of_Register** (*control*, *relativeposition*)

Argument	Description
instancename	Instance name of u_st_splitbar
control	DragObject containing the control to be registered
relativeposition	Integer or u_st_splitbar constant specified how the service positions *control* relative to the splitbar: ♦ **1 or LEFT** Position *control* to the left of the splitbar ♦ **2 or RIGHT** Position *control* to the right of the splitbar ♦ **3 or ABOVE** Position *control* above the splitbar ♦ **4 or BELOW** Position *control* below the splitbar

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function in the splitbar's Constructor event to specify the controls affected by the splitbar.

Examples

This example calls the of_Register function:

```
this.of_Register (tv_1, LEFT)
this.of_Register (lv_1, RIGHT)
this.of_SetBarColor(RGB(192, 192, 192))
```

of_SetBarColor

Description: Specifies the splitbar color.

Access: Public

Syntax: *instancename*.**of_SetBarColor** (*color*)

Argument	Description
instancename	Instance name of u_st_splitbar
color	Long specifying the splitbar color. The default bar color is maintained in the il_barcolor instance variable

Return value: Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage: Call this function in the splitbar's Constructor event.

Examples: This example calls the of_SetBarColor function:

```
this.of_SetBarColor(RGB(192, 192, 192))
```

of_SetBarMoveColor

Description: Specifies the color that displays when the splitbar is being moved.

Access: Public

Syntax: *instancename*.**of_SetBarMoveColor** (*color*)

Argument	Description
instancename	Instance name of u_st_splitbar
color	Long specifying the splitbar color when the user moves it. The default bar move color is maintained in the il_barmovecolor instance variable

Return value: Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage: Call this function in the splitbar's Constructor event.

Examples: This example calls the of_SetBarMoveColor function:

```
this.of_Register(tv_1, LEFT)
this.of_Register(lv_1, RIGHT)
this.of_SetBarColor(RGB(192, 192, 192))
this.of_SetBarMoveColor(RGB(0, 0, 0))
```

of_SetHorizontalPointer

Description

Specifies the name of the pointer that displays when the cursor is over a horizontal splitbar.

Access

Public

Syntax

instancename.**of_SetHorizontalPointer** (*pointername*)

Argument	Description
instancename	Instance name of u_st_splitbar
pointername	String specifying the pointer that displays when the cursor is over a horizontal splitbar. This can be one of the following: ♦ Arrow! ♦ Beam! ♦ Cross! ♦ Hourglass! ♦ SizeNESW! ♦ SizeNS! (default) ♦ SizeNWSE! ♦ SizeWE! ♦ Uparrow! ♦ Name of a customized cursor file (CUR extension). This file must be accessible at execution time

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function in the splitbar's Constructor event.

Examples

This example calls the of_SetHorizontalPointer function:

```
this.of_Register(tv_1, LEFT)
this.of_Register(lv_1, RIGHT)
this.of_SetHorizontalPointer ("SizeNS!")
```

of_SetMinObjectSize

Description

Specifies the minimum size (in PBUs) for registered objects.

Access

Public

Syntax

instancename.**of_SetMinObjectSize** (*minimumsize*)

Argument	Description
instancename	Instance name of u_st_splitbar
minimumsize	Integer specifying the minimum size (in PBUs) for registered objects

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function in the splitbar's Constructor event.

Examples

This example calls the of_SetMinObjectSize function:

```
this.of_Register(tv_1, LEFT)
this.of_Register(lv_1, RIGHT)
this.of_SetBarColor(RGB(192,192,192))
this.of_SetBarMoveColor(RGB(128, 128, 128))
this.of_SetMinObjectSize(100)
```

of_SetVerticalPointer

Description

Specifies the name of the pointer that displays when the cursor is over a vertical splitbar.

Access

Public

Syntax

instancename.**of_SetVerticalPointer** (*pointername*)

Argument	Description
instancename	Instance name of u_st_splitbar

Argument	Description
pointername	String specifying the pointer that displays when the cursor is over a vertical splitbar. This can be one of the following: ♦ Arrow! ♦ Beam! ♦ Cross! ♦ Hourglass! ♦ SizeNESW! ♦ SizeNS! (default) ♦ SizeNWSE! ♦ SizeWE! ♦ Uparrow! ♦ Filename of a customized cursor file (CUR extension). This file must be accessible at execution time

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function in the splitbar's Constructor event.

Examples

This example calls the of_SetVerticalPointer function:

```
this.of_Register(tv_1, LEFT)
this.of_Register(lv_1, RIGHT)
this.of_SetBarColor(RGB(192,192,192))
this.of_SetVerticalPointer("SizeWE!")
```

of_UpdateObjectData

Description

Updates the instance variables that maintain object coordinates.

Access

Protected

Syntax

instancename.**of_UpdateObjectData** ( )

Argument	Description
instancename	Instance name of u_st_splitbar

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the of_LButtonUp function:

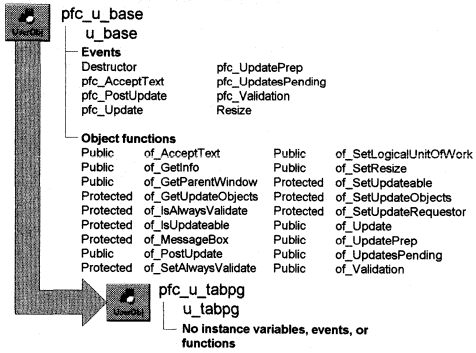
```
...
li_deltax = li_pointerx - ii_prevpositionx
li_deltay = li_pointery - ii_prevpositiony
of_UpdateObjectData()
...
```

u_tabpg

Description

Tab page visual user object ancestor.

Ancestry



Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use this visual object in the User Object painter as the ancestor for all tab pages. After defining the tab page's appearance and functionality, you then place these tab pages on tab user objects in the User Object painter. To insert a tab page object, click the right mouse button on a tab control, select Insert UserObject, and double-click on a u_tabpg descendant.

See also

u_tab

CHAPTER 8

Standard Class User Objects

About this chapter

This chapter describes the standard class user objects in PFC.

Contents

The standard class user objects are listed in alphabetical order. Each object's discussion includes alphabetical listings of instance variables, events, and object functions.

n_cn

Description

Connection object for use with all distributed PFC applications. You can either use this object as is or create descendants that include customized or extended functionality.

Ancestry

Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use n_cn whenever you would normally create a Connection object. By calling the of_Init function, you can easily initialize the Connection object.

To use n_cn:

1 Declare a variable of type n_cn (this example declares an instance variable):

```
n_cn   icn_connection
```

2 Create the Connection object:

```
icn_connection = CREATE n_cn
```

3 Call the of_Init function to initialize Connection object fields:

```
icn_connection.of_Init &
   (gnv_app.of_GetAppINIFile(),"Connect")
```

4 Continue the distributed application as appropriate.

See also

n_trp

Functions

N_cn includes precoded object functions:

of_Init
of_MessageBox

of_Init

Description

Initializes a Connection object's properties using values from an INI file section or registry key.

Access

Public

Syntax

instancename.**of_Init** (*fileorkey*, *section*)

Argument	Description
instancename	Instance name of n_cn
fileorkey	String specifying either the INI file or the registry key containing connection information. If you specify the INI file, you must also specify *section*
section	String specifying the INI file section containing connection information. If *fileorkey* specifies an INI file, you must specify this value; if *fileorkey* specifies a registry key, you cannot specify this value

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

The INI file section or registry key must contain the following keywords:

Application
ConnectString
Driver
Location
Options
Password
Trace
UserID

If the INI file section or registry key doesn't have a value, the function defaults the value to an empty string.

FOR INFO For complete information on distributed computing, see *Application Techniques*.

Examples

This example calls the of_Init function (assuming an icn_connection instance variable):

```
Integer   li_return

icn_connection = CREATE n_cn
li_return = icn_connection.of_Init &
    ("c:\HKEY_CURRENT_USER\Software\" &
    + "Powersoft\EISUSER\Connect")
```

```
IF li_return = -1 THEN
   MessageBox("Connect", &
      "Remote connection error")
ELSE
   gnv_app.of_GetFrame().SetMicroHelp &
      ("Connection succeeded")
END IF
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of n_cn
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!

Argument	Description
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in Connection objects.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('pfc_cn_dberror','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

n_cxinfo

Description

ContextInformation nonvisual user object ancestor.

Ancestry

pfc_n_cxinfo
n_cxinfo
No instance variables, events or functions

Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use this nonvisual object instead of the standard PowerBuilder ContextInformation object.

See also

n_cxk
n_inet
n_ir
n_srv

n_cxk

Description

ContextKeyword nonvisual user object ancestor.

Ancestry

pfc_n_cxkey
n_cxkey
No instance variables, events or functions

Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use this nonvisual object instead of the standard PowerBuilder ContextKeyword object.

See also

n_cxinfo
n_inet
n_ir
n_srv

n_dda

Description

DynamicDescriptionArea nonvisual user object ancestor.

Ancestry

pfc_n_dda
n_dda
No instance variables, events or functions

Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use this nonvisual object instead of the standard PowerBuilder DynamicDescriptionArea object.

See also

n_dsa
n_err
n_msg
n_tr

n_ds

Description

DataStore object for use with all PFC applications. This user object includes:

- Functions to enable and disable DataStore services
- Events and functions that call DataStore services
- Events and functions that are called by the logical unit of work service

You can either use this object asis or create descendants that include customized or extended functionality.

N_ds is a self-updating object.

Ancestry

pfc_n_ds
n_ds

Events

DBError	pfc_ResetUpdate
Destructor	pfc_Retrieve
pfc_PageSetup	pfc_Ruler
pfc_PageSetupDlg	pfc_Update
pfc_PostUpdate	pfc_UpdatePrep
pfc_PrePageSetupDlg	pfc_UpdatesPending
pfc_PrePrintDlg	pfc_Validation
pfc_Print	pfc_Zoom
pfc_PrintDlg	RetrieveStart
pfc_PrintImmediate	SQLPreview
pfc_PrintPreview	

Object functions

Public	of_AcceptText	Public	of_SetParentWindow
Public	of_CheckRequired	Public	of_SetPrintPreview
Public	of_GetAppend	Public	of_SetReport
Public	of_GetParentWindow	Public	of_SetTransObject
Public	of_IsUpdateable	Public	of_SetUpdateable
Protected	of_MessageBox	Public	of_SetUpdateRequestor
Public	of_PostUpdate	Public	of_Update
Public	of_Retrieve	Public	of_UpdatePrep
Public	of_SetAppend	Public	of_UpdatesPending
Public	of_SetBase	Public	of_Validation
Public	of_SetMultiTable		

Library

PFCMAIN.PBL
PFEAPSRV.PBL

Object relationships

n_cst_conversion
n_cst_dssrv
n_cst_dssrv_multitable
n_cst_dssrv_printpreview
n_cst_dssrv_report
n_cst_platform
n_tr
s_pagesetupattrib

s_printdlgattrib
w_master

Usage

Use n_ds whenever you would normally create a DataStore. N_ds has the same relationship to n_cst_dssrv as u_dw has to n_cst_dwsrv.

To use n_ds:

1 Declare a variable of type n_ds (this example declares an instance variable):

```
n_ds  ids_datastore
```

2 Create the DataStore:

```
ids_datastore = CREATE n_ds
```

3 Associate a DataWindow object with the DataStore:

```
ids_datastore.DataObject = "d_emplist"
```

4 Set the Transaction object by calling the of_SetTransObject function:

```
ids_datastore.of_SetTransObject(SQLCA)
```

5 (Optional) Enable DataStore services:

```
inv_datastore.of_SetPrintPreview(TRUE)
inv_datastore.of_SetReport(TRUE)
```

6 Add code to the pfc_Retrieve event to retrieve rows:

```
Return this.Retrieve()
```

7 Access data and perform other DataStore functions as needed. This example retrieves data:

```
inv_datastore.of_Retrieve()
```

See also

n_cst_dwsrv
u_dw

Instance variables

N_ds contains instance variables.

Instance variable	Description	Data type	Access	Usage
ib_append	Indicates whether the Retrieve function appends	Boolean	Protected	Set with of_SetAppend
ib_isupdateable	Indicates whether the DataStore is updatable	Boolean	Protected	Set with of_SetUpdateable
inv_base	Reference variable for basic DataStore service	n_cst_dssrv	Public	Use in dot notation to access n_cst_dssrv functions and events
inv_multitable	Reference variable for multitable update service	n_cst_dssrv_multitable	Public	Use in dot notation to access n_cst_dssrv_multitable functions and events
inv_printpreview	Reference variable for print preview service	n_cst_dssrv_printpreview	Public	Use in dot notation to access n_cst_dssrv_printpreview functions and events
inv_report	Reference variable for report service	n_cst_dssrv_report	Public	Use in dot notation to access n_cst_dssrv_report functions and events
ipo_updaterequestor	Update requestor	PowerObject	Protected	Set with of_SetUpdateRequestor
is_updatesallowed	Specifies allowable updates	String	Protected	Internal
itr_object	Transaction object used by the DataStore	n_tr	Public	Set with of_SetTransObject

Events

N_ds includes precoded event scripts:

DBError	pfc_ResetUpdate
Destructor	pfc_Retrieve
pfc_PageSetup	pfc_Ruler
pfc_PageSetupDlg	pfc_Update
pfc_PostUpdate	pfc_UpdatePrep

pfc_PrePageSetupDlg	pfc_UpdatesPending
pfc_PrePrintDlg	pfc_Validation
pfc_Print	pfc_Zoom
pfc_PrintDlg	RetrieveStart
pfc_PrintImmediate	SQLPreview
pfc_PrintPreview	

DBError

Description — Displays a message box informing the user that a database error occurred.

Usage — This event is called when there is a database error.

Destructor

Description — Destroys all existing DataStore service objects.

Usage — This event executes when the DataStore is destroyed.

pfc_PageSetup

Description — Calls the pfc_PageSetupDlg event to display the Page Setup dialog box.

Return value — Integer. Returns 1 if the event succeeds, 0 if the user cancels out of the dialog box, and -1 if an error occurs.

pfc_PageSetupDlg

Description — Displays the Page Setup dialog box by calling the n_cst_platform of_PageSetupDlg function, passing the DataWindow's page display properties.

Syntax — *instancename.Event* **pfc_PageSetupDlg** (*attributes*)

Argument	Description
instancename	Instance name of n_ds
attributes	S_pagesetupattrib variable into which the event places page setup information. This argument is accessed through the *astr_pagesetup* argument (passed by reference)

Return value — Integer. Returns 1 if the event succeeds, 0 if the user cancels out of the dialog box, and -1 if an error occurs.

Usage

This event is called by the pfc_PageSetup event.

pfc_PostUpdate

Description

Clears the DataStore update flags.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

You can extend this event to code additional post update processing.

Examples

This example is from the of_PostUpdate function:

```
Return this.Event pfc_PostUpdate()
```

pfc_PrePageSetupDlg

Description

Empty user event allowing you to modify the properties passed to the n_cst_platform of_PageSetupDlg function.

Syntax

instancename.Event **pfc_PrePageSetupDlg** (*attributes*)

Argument	Description
instancename	Instance name of n_ds
attributes	S_pagesetupattrib variable into which the event places additional page setup information. This argument is accessed through the *astr_pagesetup* argument (passed by reference)

Usage

This event is called by the pfc_PageSetupDlg event before calling the n_cst_platform of_PageSetupDlg function.

You can use this event to modify or extend the information passed in the s_pagesettupattrib structure.

Examples

This example contains code you might add to the pfc_PrePageSetupDlg event:

```
// Sets page setup to portrait
astr_pagesetup.b_portraitorientation = TRUE
```

pfc_PrePrintDlg

Description

Empty user event allowing you to modify the properties passed to the n_cst_platform of_PrintDlg function.

Syntax

instancename.Event **pfc_PrePrintDlg** (*attributes*)

Argument	Description
instancename	Instance name of n_ds
attributes	S_printdlgattrib variable into which the event places additional printing information. This argument is accessed through the *astr_printdlg* argument (passed by reference)

Return value

None

Usage

Use this event to modify or extend the information passed in the s_printdlgattrib structure.

Examples

This example contains code you might add to the pfc_PrePrintDlg event:

```
// Default copies to 1
astr_printdlg.l_copies = 1
```

pfc_Print

Description

Calls the pfc_PrintDlg function and prints the DataStore, as specified in the Print dialog box.

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

pfc_PrintDlg

Description

Initializes the s_printdlgattrib structure with the DataStore's current settings, displays the Print dialog box by calling the n_cst_platform of_PrintDlg function, and resets the DataWindow's settings as specified by the user.

Syntax

instancename.Event **pfc_PrintDlg** (*attributes*)

Argument	Description
instancename	Instance name of n_ds
attributes	S_printdlgattrib variable into which the event places printing information. This argument is accessed through the *astr_printdlg* argument (passed by reference)

Return value

Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage

This event is called by the pfc_Print event.

pfc_PrintImmediate

Description	Prints the current DataStore without displaying the Print dialog box.
Return value	Integer. Returns 1 if the event succeeds and -1 if an error occurs.

pfc_PrintPreview

Description	Toggles the DataStore between preview and edit modes.
Return value	Boolean. Returns TRUE if the DataStore is placed in preview mode and FALSE if it is placed in edit mode.
Usage	To use this event, you must enable the print preview service by calling the of_SetPrintPreview function.

pfc_ResetUpdate

Description	Clears the DataStore's update flags.
Return value	Integer. Returns 1 if the event succeeds and -1 if an error occurs.
Usage	The pfc_PostUpdate event calls this event.

pfc_Retrieve

Description	Empty user event to contain all database retrieval logic.
Return value	Long. Returns the result of the Retrieve function.
Usage	You typically call the of_Retrieve function, which calls this event.
Examples	This example calls the pfc_Retrieve event:

```
ids_data = CREATE n_ds
ids_data.DataObject = "d_empall"
ids_data.of_SetTransObject(SQLCA)
IF ids_data.Event pfc_Retrieve() = -1 THEN
   SQLCA.of_Rollback()
ELSE
   SQLCA.of_Commit( )
END IF
```

This example shows the code you add to the pfc_Retrieve event to retrieve rows:

```
Return this.Retrieve( )
```

pfc_Ruler

Description

Toggles the DataStore between displaying and hiding rulers in print preview mode.

Return value

Boolean. Returns TRUE if print preview rulers are displayed and FALSE if they are hidden.

Usage

To use this event, you must enable the print preview service by calling the of_SetPrintPreview function.

pfc_Update

Description

Updates the DataStore. If the multitable update service is enabled, this event updates all specified tables.

Syntax

instancename.Event **pfc_Update** (*accepttext, resetflags*)

Argument	Description
instancename	Instance name of n_ds
accepttext	Boolean specifying whether the DataStore control should automatically perform an AcceptText before performing the Update (TRUE) or not (FALSE)
resetflags	Boolean specifying whether the DataStore control should automatically reset the update flags (TRUE) or not (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this event to update a DataStore.

pfc_UpdatePrep

Description

Empty user event to which you can add code that prepares for update.

Return value

Long. Return 1 if the update preparation succeeds and -1 to halt the update process.

Usage — The of_UpdatePrep function calls this function.

pfc_UpdatesPending

Description — Determines if there are pending updates for the DataStore.

Return value — Integer. Returns 1 if there are pending updates for the DataStore and 0 if there are no pending updates.

Usage — This event is called by the of_UpdatesPending function.

pfc_Validation

Description — Validates the DataStore, including checks for required fields.

Return value — Integer. Returns 1 if the event succeeds and -1 if an error occurs.

Usage — Extend this event to perform DataStore-specific validation.

pfc_Zoom

Description — Displays the w_zoom dialog box, allowing the user to control DataStore display while in print preview mode.

Return value — Integer. Returns the zoom level chosen by the user if the event succeeds, 0 if the user cancels out of the w_zoom dialog box, and -1 if an error occurs.

Usage — To use this event, you must enable the print preview service by calling the of_SetPrintPreview function.

RetrieveStart

Description — Determines whether PowerBuilder should append data to the end of the DataStore when retrieving.

Usage — This event executes just before the actual database retrieval. You can control whether PowerBuilder appends to the end of the DataStore when retrieving using the of_SetAppend function.

SQLPreview

Description

Invokes the SQL Spy service, if enabled and controls which SQL statements are sent to the database.

Usage

This event executes just before any database access.

Functions

N_ds includes precoded object functions:

of_AcceptText	of_SetParentWindow
of_CheckRequired	of_SetPrintPreview
of_GetAppend	of_SetReport
of_GetParentWindow	of_SetTransObject
of_IsUpdateable	of_SetUpdateable
of_MessageBox	of_SetUpdateRequestor
of_PostUpdate	of_Update
of_Retrieve	of_UpdatePrep
of_SetAppend	of_UpdatesPending
of_SetBase	of_Validation
of_SetMultiTable	

of_AcceptText

Description

Performs an AcceptText function for the DataStore.

Access

Public

Syntax

instancename.**of_AcceptText** (*focusonerror*)

Argument	Description
instancename	Instance name of n_ds
focusonerror	Boolean indicating whether PFC sets focus to the DataStore when an error occurs

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize accept text processing, extend the pfc_AcceptText event.

Examples

This example is from the n_cst_luw of_AcceptText function:

```
...
If lb_defined Then
   li_rc =  &
      lpo_tocheck.Function Dynamic of_AcceptText &
         (ab_focusonerror)
   If li_rc < 0 Then Return -1

...
```

of_CheckRequired

Description

Determines if any of required columns contain NULL values.

Access

Public

Syntax

instancename.**of_CheckRequired** (*buffer*, *row*, *column*, *colname*, *updateonly*)

Argument	Description
instancename	Instance name of n_ds
buffer	DWBuffer enumerated data type specifying the DataStore buffer to check
row	Long specifying the first row to check and into which the function places the number of the row in error (passed by reference)
column	Integer specifying the first column to check and into which the function places the number of the column in error (passed by reference)
colname	String specifying the first column to check and into which the function places the name of the column in error (passed by reference)
updateonly	Boolean indicating whether to validate only those rows and columns that have changed (TRUE) or validate all rows and columns

Return value

Integer. Returns values as follows:

- **1** The required fields check was successful but errors were found; check the arguments for missing required fields
- **0** The required fields check was successful and no errors were found

- **-1** An error occurred

Usage

The pfc_Validation event calls this function.

Examples

This example is from the n_ds pfc_Validation event:

```
...
li_rc = of_CheckRequired (primary!, ll_checkrow, &
   li_checkcolumn, ls_checkcolname, ib_updateonly)
if (li_rc < 0) or (ll_checkrow > 0) then return -1
...
```

of_GetAppend

Description

Reports whether DataStore retrieves append to the end of the DataStore.

Access

Public

Syntax

instancename.**of_GetAppend** ( )

Argument	Description
instancename	Instance name of n_ds

Return value

Boolean. Returns TRUE if retrieves append to the end of the DataStore and FALSE if retrieves reset the DataStore before the retrieve.

Examples

This example calls the of_GetAppend function:

```
Boolean  lb_append

lb_append = ids_datasource.of_GetAppend()
...
```

of_GetParentWindow

Description

Retrieves a reference to the window containing this instance of n_ds.

Access

Public

Syntax

instancename.**of_GetParentWindow** (*window*)

Argument	Description
instancename	Instance name of n_ds

Argument	Description
window	Window variable into which the function places a reference to the parent window (passed by reference)

Return value

Integer. Returns 1 if the function succeeds and -1 if there is no parent window.

Usage

Internal.

Examples

This example is from the pfc_PrintDlg event:

```
...
f_setplatform (lnv_platform, true)
this.of_GetParentWindow(lw_parent)
ll_rc = lnv_platform.of_PrintDlg(astr_printdlg,  &
   lw_parent)
...
```

of_IsUpdateable

Description

Reports whether the DataStore is updatable.

Access

Public

Syntax

instancename.**of_IsUpdateable ( )**

Argument	Description
instancename	Instance name of n_ds

Return value

Boolean. Returns TRUE if the DataStore is updatable and FALSE if it is not.

Examples

This example is from the pfc_UpdatesPending event:

```
...
if not of_IsUpdateable() then
   return 0
end if
...
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of n_ds
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in DataStore objects.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('ds_dberror','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

of_PostUpdate

Description

Calls the pfc_PostUpdate event, which clears update flags and allows you to code additional post-update processing.

Access

Public

Syntax

instancename.**of_PostUpdate** ( )

Argument	Description
instancename	Instance name of n_ds

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize post-update processing, extend the pfc_PostUpdate event.

Examples

This example is from the n_cst_luw of_PostUpdate function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_PostUpdate()
...
```

of_Retrieve

Description

Calls the pfc_Retrieve event.

Access

Public

Syntax

instancename.**of_Retrieve** ( )

Argument	Description
instancename	Instance name of n_ds

Return value

Integer. Returns the number of rows retrieved if the function succeeds and -1 if an error occurs.

Usage

Call this function to perform DataStore retrieves. You must also code retrieve logic in the pfc_Retrieve event.

Examples

This example calls the of_Retrieve function:

```
ids_data.of_Retrieve()
```

of_SetAppend

Description

Specifies whether DataStore retrieves append to the end of the DataStore.

Access

Public

Syntax

instancename.**of_SetAppend** (*boolean*)

Argument	Description
instancename	Instance name of n_ds
boolean	Boolean specifying whether DataStore retrieves append to the end of existing rows (TRUE) or reset the DataStore before the retrieve (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetAppend function:

```
Integer  li_return

li_return = ids_datasource.of_SetAppend(FALSE)
```

of_SetBase

Description

Enables or disables n_cst_dssrv, which provides basic DataStore services.

Access

Public

Syntax

instancename.**of_SetBase** (*boolean*)

Argument	Description
instancename	Instance name of n_ds
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) an instance of the n_cst_dssrv object

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function to create or destroy an instance of n_cst_dssrv.

Examples

This example calls the of_SetBase function:

```
Integer  li_return

li_return = ids_datastore.of_SetBase(TRUE)
IF li_return = -1 THEN
```

```
    MessageBox("DataStore", "Error with of_SetBase")
END IF
```

of_SetMultiTable

Description

Enables or disables n_cst_dssrv_multitable, which provides multitable update services.

Access

Public

Syntax

instancename.**of_SetMultiTable** (*boolean*)

Argument	Description
instancename	Instance name of n_ds
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the multitable update service

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_dssrv_multitable.

Examples

This example calls the of_SetMultiTable function:

```
ids_data.of_SetMultiTable(TRUE)
```

of_SetParentWindow

Description

Sets a reference to the parent window.

Access

Public

Syntax

instancename.**of_SetParentWindow** (*window*)

Argument	Description
instancename	Instance name of n_ds
window	Window variable containing the parent window

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Some n_ds print functionality requires a reference to a parent window.

Examples

This example calls the of_SetParentWindow function in a window Open event:

```
ids_data = CREATE n_ds
```

```
ids_data.DataObject = "d_empall"
ids_data.of_SetTransObject(SQLCA)
ids_data.of_SetParentWindow(this)
...
```

of_SetPrintPreview

Description

Enables or disables n_cst_dwsrv_printpreview, which provides the print preview service.

Access

Public

Syntax

instancename.**of_SetPrintPreview** (*boolean*)

Argument	Description
instancename	Instance name of n_ds
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the print preview service

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_dwsrv_printpreview.

Examples

This example calls the of_SetPrintPreview function:

```
ids_data.of_SetPrintPreview(TRUE)
```

of_SetReport

Description

Enables or disables n_cst_dwsrv_report, which provides the reporting service.

Access

Public

Syntax

instancename.**of_SetReport** (*boolean*)

Argument	Description
instancename	Instance name of n_ds
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) the reporting service

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Use this function to create or destroy an instance of n_cst_dwsrv_report.

Examples

This example calls the of_SetReport function:

```
ids_data.of_SetReport(TRUE)
```

of_SetTransObject

Description

Sets the Transaction object for the DataStore.

Access

Public

Syntax

instancename.**of_SetTransObject** (*transaction*)

Argument	Description
instancename	Instance name of n_ds
transaction	N_tr variable specifying the Transaction object to use for the DataStore

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetTransObject function (it assumes you have associated n_tr with SQLCA in the Application painter):

```
Integer   li_return

li_return = ids_datastore.of_SetTransObject(SQLCA)
IF li_return = -1 THEN
   MessageBox("DataStore", &
      "Error with of_SetTransObject")
END IF
```

of_SetUpdateable

Description

Specifies whether the DataStore is updatable.

Access

Public

Syntax

instancename.**of_SetUpdateable** (*boolean*)

Argument	Description
instancename	Instance name of n_ds
boolean	Boolean indicating whether the DataStore is updatable (TRUE) or not (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function to disable default save processing for DataStores that are not updatable.

Examples

This example calls the of_SetUpdateable function:

```
ids_data.of_SetUpdateable(FALSE)
```

of_SetUpdateRequestor

Description

Creates a reference to the object requesting an update within a logical unit of work.

Access

Public

Syntax

instancename.**of_SetUpdateRequestor** (*requestor*)

Argument	Description
instancename	Instance name of n_ds
requestor	PowerObject containing the object requesting the update

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Internal.

Examples

This example is from the of_Update function:

```
...
If this.of_SetUpdateRequestor(apo_requestor) <0 Then
   Return -1
END IF
li_rc = this.of_Update(ab_accepttext, ab_resetflag)
this.of_SetUpdateRequestor(lpo_notvalid)
...
```

of_Update

Updates the DataStore. There are two syntaxes:

To	Use
Update data, optionally controlling update type	Syntax 1
Update data passing the requestor object	Syntax 2

Syntax 1 **Update data, optionally controlling update type**

Description

Updates rows in the DataStore. With this syntax, you can optionally control the update types:

Insert
Update
Delete

Access

Public

Syntax

instancename.**of_Update** (*accepttext*, *resetflags* {, *insert*, *update*, *delete* })

Argument	Description
instancename	Instance name of n_ds
accepttext	Boolean indicating whether the DataStore should automatically perform an AcceptText before performing the update: TRUE—Perform AcceptText (default) FALSE—Do not perform AcceptText
resetflags	Boolean indicating whether *instancename* should automatically reset the update flags: TRUE—Reset the flags (default) FALSE—Do not reset the flags
insert (optional)	Boolean indicating whether to insert rows: TRUE—Insert rows (default) FALSE—Do not insert rows
update (optional)	Boolean indicating whether to modify changed rows: TRUE—Modify rows (default) FALSE—Do not modify rows
delete (optional)	Boolean indicating whether to delete rows: TRUE—Delete rows (default) FALSE—Do not delete rows

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize update processing, extend the pfc_Update event.

FOR INFO For more information on update flags, see the Update function in the *PowerScript Reference*.

Examples

This example calls the of_Update function:

```
Integer  li_return

li_return = ids_data.of_Update(TRUE, TRUE)
```

Syntax 2 **Update data passing the requestor object**

Description Updates rows in the DataStore, passing a reference to the requestor object.

Access Public

Syntax *instancename*.**of_Update** (*accepttext*, *resetflags*, *requestor*)

Argument	Description
instancename	Instance name of n_ds
accepttext	Boolean indicating whether the DataStore should automatically perform an AcceptText before performing the update: TRUE—Perform AcceptText (default) FALSE—Do not perform AcceptText
resetflags	Boolean indicating whether *instancename* should automatically reset the update flags: TRUE—Reset the flags (default) FALSE—Do not reset the flags
requestor	PowerObject containing the requestor object

Return value Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage Internal.

Examples This example is from the n_cst_luw of_Update function:

```
...
If lb_defined Then
  li_rc = lpo_tocheck.Function Dynamic of_Update &
    (ab_accepttext,  ab_resetflag, &
      lpo_updaterequestor)
  If li_rc < 0 Then Return -1
  Continue
End If
...
```

of_UpdatePrep

Description

Calls the pfc_UpdatePrep event, which allows you to code additional update preparation logic.

Access

Public

Syntax

instancename.**of_UpdatePrep** ( )

Argument	Description
instancename	Instance name of n_ds

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize update preparation processing, extend the pfc_UpdatePrep event.

Examples

This example is from the n_cst_luw of_UpdatePrep function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_UpdatePrep ()
  If li_rc < 0 Then Return -1
  Continue
End If
...
```

of_UpdatesPending

Description

Determines if there are updates pending in the DataStore.

Access

Public

Syntax

instancename.**of_UpdatesPending** ( )

Argument	Description
instancename	Instance name of n_ds

Return value

Integer. Returns values as follows:

- **1** Updates are pending
- **0** No updates pending

- **-1** An error occurred

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize pending updates processing, extend the pfc_UpdatesPending event.

Examples

This example calls the of_UpdatesPending function:

```
...
If lb_defined Then
   la_rc = lpo_tocheck.Dynamic of_UpdatesPending ()
...
```

of_Validation

Description

Performs validation on the DataStore.

Access

Public

Syntax

instancename.**of_Validation ( )**

Argument	Description
instancename	Instance name of n_ds

Return value

Integer. Returns 1 if the function succeeds and -1 if a validation error occurs.

Usage

N_cst_luw calls this function as part of the default save process. This function is part of the self-updating object API. To customize validation processing, extend the pfc_Validation event.

Examples

This example calls the of_Validation function:

```
...
If lb_defined Then
  li_rc =  &
    lpo_tocheck.Function Dynamic of_Validation()
...
```

n_dsa

Description

DynamicStagingArea nonvisual user object ancestor.

Ancestry

Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use this nonvisual object instead of the standard PowerBuilder DynamicStagingArea object.

See also

n_dda
n_err
n_msg
n_tr

n_err

Description

Error nonvisual user object ancestor.

Ancestry

pfc_n_err
n_err
Object functions
Protected of_MessageBox

Library

PFCMAIN.PBL
PFEAPSRV.PBL

Usage

Use this nonvisual object instead of the standard PowerBuilder error object.

See also

n_dda
n_dsa
n_msg
n_tr

Functions

N_err contains one precoded function:

of_MessageBox

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id*, *title*, *message*, *icon*, *button*, *default*)

Argument	Description
instancename	Instance name of n_err
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text

Argument	Description
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in Error objects.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('err_dberror','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

n_inet

Description

Inet nonvisual user object ancestor.

Ancestry

pfc_n_inet
n_inet
No instance variables, events or functions

Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use this nonvisual object instead of the standard PowerBuilder Inet object.

See also

n_cxinfo
n_cxk
n_ir
n_srv

n_ir

Description

InternetResult nonvisual user object ancestor.

Ancestry

pfc_n_ir
n_ir
No instance variables, events or functions

Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use this nonvisual object instead of the standard PowerBuilder InternetResult object.

See also

n_cxinfo
n_cxk
n_inet
n_srv

n_ms

Description

MailSession nonvisual user object ancestor.

Ancestry

pfc_n_ms
n_ms
Object functions
Protected of_MessageBox

Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use this nonvisual object instead of the standard PowerBuilder MailSession object.

See also

n_ds
n_err
n_msg
n_pl
n_tr

Functions

N_ms contains one precoded function:

of_MessageBox

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id*, *title*, *message*, *icon*, *button*, *default*)

Argument	Description
instancename	Instance name of n_ms
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text

Argument	Description
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in MailSession objects.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('ms_dberror','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

n_msg

Description

Message nonvisual user object ancestor.

Ancestry

Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use this nonvisual object instead of the standard PowerBuilder Message object.

See also

n_ds
n_err
n_ms
n_pl
n_tr

Functions

N_msg contains one precoded function:

of_MessageBox

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id*, *title*, *message*, *icon*, *button*, *default*)

Argument	Description
instancename	Instance name of n_msg
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text

Argument	Description
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in Message objects.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('msg_dberror','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

n_oo

Description

OLEObject nonvisual user object ancestor.

Ancestry

Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use this nonvisual object instead of the standard PowerBuilder OLEObject object.

See also

n_ostg
n_ostm

n_ostg

Description

OLEStorage nonvisual user object ancestor.

Ancestry

pfc_n_ostg
n_ostg
No instance variables, events or functions

Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use this nonvisual object instead of the standard PowerBuilder OLEStorage object.

See also

n_oo
n_ostm

n_ostm

Description

OLEStream nonvisual user object ancestor.

Ancestry

pfc_n_ostm
n_ostm
No instance variables, events or functions

Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use this nonvisual object instead of the standard PowerBuilder OLEStream object.

See also

n_oo
n_ostg

n_pl

Description

Pipeline nonvisual user object ancestor.

Ancestry

pfc_n_pl
n_pl
Object functions
Protected of_MessageBox

Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use this nonvisual object instead of the standard PowerBuilder pipeline object.

See also

n_ds
n_err
n_ms
n_msg
n_tr

Functions

N_pl contains one precoded function:

of_MessageBox

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id*, *title*, *message*, *icon*, *button*, *default*)

Argument	Description
instancename	Instance name of n_pl
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text

Argument	Description
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in Pipeline objects.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('pl_dberror','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

n_srv

Description

Service nonvisual user object ancestor.

Ancestry

pfc_n_srv
n_srv
No instance variables, events or functions

Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use this nonvisual object instead of the standard PowerBuilder Service object. The Service object is used in implementing extensions to the PowerBuilder context feature.

See also

n_cxinfo
n_cxk
n_inet
n_ir

n_tmg

Description: Timing nonvisual user object ancestor.

Ancestry:

pfc_n_ir
n_ir
No instance variables, events or functions

Library: PFCMAIN.PBL
PFEMAIN.PBL

Object relationships: n_cst_tmgmultiple
n_cst_tmgsingle

Usage: Use this nonvisual object instead of the standard PowerBuilder Timing object.

See also: n_tr

Instance variables

N_tmg includes instance variables:

Instance variable	Description	Data type	Access	Usage
inv_multiple	Reference variable for multiple notify service	N_cst_tmgmultiple	Public	Call of_SetMultiple to enable
inv_single	Reference variable for single notify service	N_cst_tmgsingle	Public	Call of_SetSingle to enable

Functions

N_tmg includes precoded object functions:

of_SetTmgMultiple
of_SetTmgSingle

of_SetTmgMultiple

Description

Enables or disables n_cst_tmgmultiple, which provides the multiple notify service.

Access

Public

Syntax

instancename.**of_SetTmgMultiple** (*boolean*)

Argument	Description
instancename	Instance name of n_tmg
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) an instance of the n_cst_tmgmultiple object

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

You cannot use both n_cst_tmgsingle and n_cst_tmgmultiple at the same time.

Examples

This example calls the of_SetTmgMultiple function:

```
Integer  li_return

li_return = itmg_timer.of_SetTmgMultiple(TRUE)
IF li_return = -1 THEN
   MessageBox("Timer", "Error with of_SetTmgMultiple")
END IF
```

of_SetTmgSingle

Description

Enables or disables n_cst_tmgsingle, which provides the single notify service.

Access

Public

Syntax

instancename.**of_SetTmgSingle** (*boolean*)

Argument	Description
instancename	Instance name of n_tmg
boolean	Boolean specifying whether to enable (TRUE) or disable (FALSE) an instance of the n_cst_tmgsingle object

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

You cannot use both n_cst_tmgsingle and n_cst_tmgmultiple at the same time.

Examples

This example calls the of_SetTmgSingle function:

```
Integer  li_return

li_return = itmg_timer.of_SetTmgSingle(TRUE)
IF li_return = -1 THEN
   MessageBox("Timer", "Error with of_SetTmgSingle")
END IF
```

n_tr

Description

Transaction object for use with all PFC applications. You can use this object as is, or create DBMS- or application-specific descendants that include customized or extended functionality.

N_tr includes integration with the SQL Spy debugging service.

FOR INFO For more information on using SQL Spy, see the *PFC User's Guide*.

Ancestry

pfc_n_tr
n_tr

Events
Destructor

Object functions

Public	of_Begin	Public	of_GetSqlState
Public	of_Commit	Public	of_GetTrace
Public	of_Connect	Public	of_Init
Public	of_CopyTo	Public	of_IsConnected
Public	of_Disconnect	Protected	of_MessageBox
Public	of_DistinctValues	Public	of_Rollback
Public	of_End	Public	of_SetAutoRollback
Public	of_Execute	Public	of_SetName
Public	of_GetAutoRollback	Public	of_SetTrace
Public	of_GetName	Public	of_SetUser

Library

PFCMAIN.PBL
PFEMAIN.PBL

Object relationships

n_cst_trregistration

Usage

Use n_tr functions instead of native SQL transaction management statements. For example, to connect to the database, use of_Connect instead of the CONNECT statement.

To register an n_tr-based Transaction object with the transaction registration service, call the n_cst_trregistration of_Register function.

To use n_tr:

1 If the Transaction object is not SQLCA, create it:

```
// Assumes an itr_security instance variable
itr_security = CREATE n_tr
```

2 (Optional) If you are using the transaction registration service:

- Register all Transaction objects:

```
gnv_app.inv_trregistration.of_Register(SQLCA)
// Assumes an itr_security instance variable
```

```
gnv_app.inv_trregistration.of_Register &
   (itr_security)
```

- Establish a transaction name for use with the transaction registration service:

```
// Assumes an itr_security instance variable
SQLCA.of_SetName("SQLCA")
itr_security.of_SetName("Security")
```

3 Initialize the ib_autorollback instance variable, which specifies what to do if the application closes (or the object is otherwise destroyed) while the transaction is still connected:

```
SQLCA.of_SetAutoRollback(TRUE)
itr_security.of_SetAutoRollback(FALSE)
```

4 Call n_tr functions as needed.

See also n_cst_trregistration

Instance variables

N_tr includes instance variables:

Instance variable	Description	Data type	Access	Usage
ib_autorollback	Specifies whether the Destructor event issues a COMMIT or a ROLLBACK if the transaction is still connected	Boolean	Protected	The Destructor event uses this event to determine whether to call of_Commit or of_Rollback: ♦ TRUE — Call of_Rollback ♦ FALSE (default) — Call of_Commit Set with of_SetAutoRollback. This controls how n_tr handles transactions if the application closes (or the object is otherwise destroyed) while the transaction is still connected
ib_trace	Specifies whether trace is enabled	Boolean	Protected	Call of_GetTrace to access this value (default is FALSE)

Instance variable	Description	Data type	Access	Usage
is_name	A name used to identify the transaction	String	Protected	You can reuse transactions between windows by passing this value (using OpenWithParm) when opening a new window. The new window can use the passed value with the n_cst_trregistration of_GetByName function to access the Transaction object Call of_GetName and of_SetName to access this value

Events

N_tr includes a precoded event:

Destructor

Destructor

Description

If the instance is still connected to a database, this event commits or rolls back the transaction, depending on the ib_autorollback instance variable. It then disconnects the transaction.

Usage

This event's functionality is meant to ensure efficient use of system resources. However, it's best to commit, rollback, and disconnect explicitly using this object's of_Commit, of_Rollback, and of_Disconnect functions.

Functions

N_tr includes precoded object functions:

of_Begin	of_GetSqlState
of_Commit	of_GetTrace
of_Connect	of_Init
of_CopyTo	of_IsConnected
of_Disconnect	of_MessageBox
of_DistinctValues	of_Rollback
of_End	of_SetAutoRollback

of_Execute	of_SetName
of_GetAutoRollback	of_SetTrace
of_GetName	of_SetUser

of_Begin

Description

Empty function into which you can code DBMS-specific BEGIN TRANSACTION statements in n_tr descendants.

Access

Public

Syntax

instancename.**of_Begin** ( )

Argument	Description
instancename	Instance name of n_tr

Return value

Long. Always returns -1. Add DBMS-specific returns values in your n_tr descendant.

Usage

Call this function to begin a database transaction.

You should connect to the database before issuing this function. Use of_Connect to connect to the database.

of_Commit

Description

Issues a COMMIT statement.

Access

Public

Syntax

instancename.**of_Commit** ( )

Argument	Description
instancename	Instance name of n_tr

Return value

Long. Returns the SQLCODE value from the COMMIT statement.

Usage

Call this function instead of the COMMIT statement to make any changes to the database permanent and end a logical unit of work.

Examples

This example calls the of_Commit function:

```
IF dw_sheet.Update( ) = -1 THEN
   itr_trans.of_Rollback( )
```

```
    MessageBox("Update", "Update failed")
ELSE
    itr_trans.of_Commit( )
    gnv_app.of_GetFrame( ).SetMicroHelp &
        ("Update succeeded")
END IF
```

of_Connect

Description

Connects to the database.

Access

Public

Syntax

instancename.**of_Connect ( )**

Argument	Description
instancename	Instance name of n_tr

Return value

Long. Returns the SQLCODE value from the CONNECT statement.

Usage

Call this function instead of the CONNECT statement to connect the transaction to the database.

You must initialize the Transaction object before calling this function. The fields initialized vary, depending on your DBMS. You can call the of_Init function to initialize the Transaction object.

Examples

This example initializes itr_security fields and connects to the database:

```
String  ls_dbms, ls_database, ls_logid, ls_logpass
String  ls_servername
Long    ll_return

itr_security.DBMS = "Syb"
itr_security.Database = "agora"
itr_security.LogID = "williamk"
itr_security.LogPass = "billpass"
itr_security.ServerName = "MUSTANG"

ll_return = itr_security.of_Connect()
IF ll_return <> 0 THEN
    MessageBox("Connect", "Connect error")
END IF
```

of_CopyTo

Description

Copies the contents of the Transaction object to the specified Transaction object.

Access

Public

Syntax

instancename.**of_CopyTo** (*targettrans*)

Argument	Description
instancename	Instance name of n_tr
targettrans	N_tr-based Transaction object to which the function copies values

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function to copy contents from one Transaction object to another. Call this function instead of simply assigning one object to another (which copies a reference, not the values).

Examples

This example calls the of_CopyTo function (it assumes an itr_back instance variable):

```
Integer  li_return

li_return = SQLCA.of_CopyTo(itr_back)
...
```

of_Disconnect

Description

Disconnects from the database.

Access

Public

Syntax

instancename.**of_Disconnect** ( )

Argument	Description
instancename	Instance name of n_tr

Return value

Long. Returns the SQLCODE value from the DISCONNECT statement.

Usage

Call this function instead of the DISCONNECT statement to disconnect the transaction from the database.

Be sure to check the return value to ensure that the function succeeded.

Examples

This example disconnects from the database:

```
Long  ll_return

ll_return = itr_security.of_Disconnect()
IF ll_return <> 0 THEN
   MessageBox("Disconnect", "Disconnect error")
END IF
```

of_DistinctValues

Description

Retrieves distinct values for the specified database column.

Access

Public

Syntax

instancename.**of_DistinctValues** (*table*, *column*, *values*)

Argument	Description
instancename	Instance name of n_tr
table	String specifying the database table
column	String specifying the column for which the function returns distinct values
values	Unbounded string array into which the function places distinct values for *column*

Return value

Long. Returns the number of elements in the *values* array.

Examples

This calls the of_DistinctValues function:

```
Long  ll_return
String  ls_values[ ]

ll_return = SQLCA.of_DistinctValues &
   ("employee", "dept_id", ls_values)
...
```

of_End

Description

Empty function into which you can code DBMS-specific END TRANSACTION statements in n_tr descendants.

Access

Public

Syntax

instancename.**of_End** ( )

Argument	Description
instancename	Instance name of n_tr

Return value

Long. Always returns -1. Add DBMS-specific returns values in your n_tr descendant.

Usage

Call this function to end a database transaction.

of_Execute

Description

Executes a specified SQL statement.

Access

Public

Syntax

instancename.**of_Execute** (*sqlstatement*)

Argument	Description
instancename	Instance name of n_tr
sqlstatement	String containing the SQL statement to execute

Return value

Long. Returns the SQLCODE returned by the EXECUTE IMMEDIATE statement. A value of -10 indicates that there is no database connection.

Examples

This example calls the of_Execute function:

```
Long  ll_return
String  ls_sql

ls_sql = "DELETE from employee WHERE dept_id = 100"
ll_return = SQLCA.of_Execute(ls_sql)
...
```

of_GetAutoRollback

Description

Reports whether autorollback is enabled.

Access

Public

Syntax

instancename.**of_GetAutoRollback** ( )

Argument	Description
instancename	Instance name of n_tr

Return value

Boolean. Returns TRUE if autorollback is enabled and FALSE if it is not.

Examples

This example calls the of_GetAutoRollback function:

```
IF SQLCA.of_GetAutoRollback() THEN
   MessageBox("Transaction", &
      "Autorollback is enabled")
ELSE
   MessageBox("Transaction", &
      "Autorollback is disabled")
END IF
```

of_GetName

Description

Retrieves the transaction name, as specified by the of_SetName function.

Access

Public

Syntax

instancename.**of_GetName** ( )

Argument	Description
instancename	Instance name of n_tr

Return value

String. Returns the transaction name.

Usage

N_cst_trregistration functions call this function to access the Transaction object name.

Examples

This example is from the n_cst_trregistration of_GetByName function:

```
...
FOR li_cnt = 1 to li_upperbound
  IF istr_trans[li_cnt].b_used AND &
    IsValid (istr_trans[li_cnt].tr_object) THEN
      ls_name = &
        istr_trans[li_cnt].tr_object.of_GetName()
        IF ls_name = as_name THEN
          atr_obj = istr_trans[li_cnt].tr_object
          lb_found = TRUE
          EXIT
```

```
        END IF
    END IF
NEXT
...
```

of_GetSqlState

Description: Retrieves the SQL state string returned by the DBMS.

Access: Public

Syntax: *instancename*.**of_GetSqlState** ( )

Argument	Description
instancename	Instance name of n_tr

Return value: String. Returns the SQL state string returned by the DBMS.

Examples: This example calls the of_GetSqlState function:

```
String  ls_sqlstate

ls_sqlstate = SQLCA.of_GetSqlState()
MessageBox("SQL State", &
    "SQL state is: " + ls_sqlstate)
```

of_GetTrace

Description: Reports whether the PowerBuilder DBMS trace is enabled.

Access: Public

Syntax: *instancename*.**of_GetTrace** ( )

Argument	Description
instancename	Instance name of n_tr

Return value: Boolean. Returns TRUE if trace is enabled and FALSE if it is not.

Examples: This example calls the of_GetTrace function:

```
IF SQLCA.of_GetTrace() THEN
   MessageBox("Transaction", "Trace is enabled")
ELSE
```

```
        MessageBox("Transaction", "Trace is disabled")
    END IF
```

of_Init

Description

Initializes the n_tr-based Transaction object with the passed values.

Access

Public

Syntax

instancename.**of_Init** (*fileorkey*, *section*)

Argument	Description
instancename	Instance name of n_tr
fileorkey	String specifying the name of the ini file or registry key containing connection information. If you specify an INI file, you must also specify *section*
section	String specifying the INI file section containing connection information. If *fileorkey* specifies a registry key, you cannot specify this value

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function to initialize n_tr before connecting to the database. The INI-file section or registry key must include the following keywords (with DBMS-specific values as appropriate):

DBMS
Database
LogID
LogPass
ServerName
UserID
DBPass
Lock
DbParm

Examples

This example calls the of_Init function:

```
String  ls_file, ls_section
Long    ll_return

ls_file = "c:\pbapp\pbapp.ini"
ls_section = "securityDB"
```

```
itr_security.of_Init &
   (ls_file, ls_section)
ll_return = itr_security.of_Connect()
IF ll_return <> 0 THEN
   MessageBox("Connect", "Connect error")
END IF
```

of_IsConnected

Description

Determines whether the n_tr instance is connected to a database.

Access

Public

Syntax

instancename.**of_IsConnected** ( )

Argument	Description
instancename	Instance name of n_tr

Return value

Boolean. Returns TRUE if the n_tr instance is connected to a database and FALSE if it is not.

Examples

This example calls the of_IsConnected function:

```
Long  ll_return

IF NOT itr_security.of_IsConnected() THEN
   ll_return = itr_security.of_Connect( )
   IF ll_return <> 0 THEN
      MessageBox("Connect", "Connect error")
   END IF
END IF
```

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id, title, message, icon, button, default*)

Argument	Description
instancename	Instance name of n_tr
id	String specifying an ID for the message

Argument	Description
title	String specifying the title of the MessageBox
message	String specifying the message text
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in Transaction objects.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('tr_dberror','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

of_Rollback

Description

Rolls back the database transaction.

Access

Public

Syntax

instancename.**of_Rollback** ( )

Argument	Description
instancename	Instance name of n_tr

Return value

Long. Returns 0 if the function succeeds, -10 if the transaction is not connected and any other nonzero value if an error occurs.

Usage

Call this function instead of the ROLLBACK statement to reverse all uncommitted changes.

Examples

This example calls the of_Rollback function:

```
IF dw_sheet.Update() = -1 THEN
   itr_trans.of_Rollback()
   MessageBox("Update", "Update failed")
ELSE
   itr_trans.of_Commit()
   gnv_app.of_GetFrame().SetMicroHelp &
      ("Update succeeded")
END IF
```

of_SetAutoRollback

Description

Controls whether autorollback is enabled.

If autorollback is enabled, the n_tr Destructor event issues a ROLLBACK for a connected transaction. If autorollback is disabled, the n_tr Destructor event issues a COMMIT.

Access

Public

Syntax

instancename.**of_SetAutoRollback** (*boolean*)

Argument	Description
instancename	Instance name of n_tr
boolean	Boolean indicating whether autorollback is enabled (TRUE) or not (FALSE)

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Examples

This example calls the of_SetAutoRollback function:

```
SQLCA.of_SetAutoRollback(TRUE)
```

of_SetName

Description

Specifies the transaction name, which is used by the transaction registration service.

Access

Public

Syntax

instancename.**of_SetName** (*name*)

Argument	Description
instancename	Instance name of n_tr
name	String specifying a name for the transaction

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Call this function to specify a transaction name when you are using the transaction registration service.

Examples

This example calls the of_SetName function:

```
SQLCA.of_SetName("SQLCA")
itr_security.of_SetName("Security")
```

of_SetTrace

Description

Specifies whether the PowerBuilder DBMS trace is enabled.

Access

Public

Syntax

instancename.**of_SetTrace** (*boolean*)

Argument	Description
instancename	Instance name of n_tr
boolean	Boolean specifying whether trace is enabled (TRUE) or not (FALSE)

Return value

Integer. Returns 1 if the function succeeds, 0 if trace is already in the specified state, and -1 if an error occurs.

Examples

This example calls the of_SetTrace function to enable trace:

```
SQLCA.of_SetTrace(TRUE)
```

of_SetUser

Description

Specifies the DBMS user ID and password (DBPass).

Access

Public

Syntax

instancename.**of_SetUser** (*userid, password*)

Argument	Description
instancename	Instance name of n_tr
userid	String specifying the user ID (Transaction object UserID property)
password	String specifying the password (Transaction object DBPass property)

Return value

None

Examples

This example, which is from an n_cst_appmanager pfc_Logon event, calls the of_SetUser function:

```
SQLCA.of_SetUser(as_userid, as_password)
IF SQLCA.of_Connect() >= 0 THEN
   Return 1
ELSE
   MessageBox (iapp_object.DisplayName, &
      "Connect failed")
   Return -1
END IF
```

n_trp

Description

Transport nonvisual user object ancestor.

Ancestry

pfc_n_trp
n_trp
Object functions
Protected of_MessageBox

Library

PFCMAIN.PBL
PFEMAIN.PBL

Usage

Use this nonvisual object instead of the standard PowerBuilder Transport object.

See also

n_cn

Functions

N_trp contains one precoded function:

of_MessageBox

of_MessageBox

Description

Displays a MessageBox.

Access

Protected

Syntax

instancename.**of_MessageBox** (*id*, *title*, *message*, *icon*, *button*, *default*)

Argument	Description
instancename	Instance name of n_trp
id	String specifying an ID for the message
title	String specifying the title of the MessageBox
message	String specifying the message text

Argument	Description
icon	Icon enumerated data type indicating the icon you want to display on the left side of the message box. Values are: ♦ Information! ♦ StopSign! ♦ Exclamation! ♦ Question! ♦ None!
button	Button enumerated data type indicating the set of CommandButtons you want to display at the bottom of the message box. The buttons are numbered in the order listed in the enumerated data type. Values are: ♦ OK! ♦ OKCancel! ♦ YesNo! ♦ YesNoCancel! ♦ RetryCancel! ♦ AbortRetryIgnore!
default	Integer specifying the number of the button you want to be the default button

Return value

Integer. Returns 1 if the function succeeds and -1 if an error occurs.

Usage

Override this function to control MessageBox behavior in Transport objects.

The *id* argument is not used in the default implementation.

Examples

This example calls the of_MessageBox function:

```
of_Messagebox('trp_dberror','Save', &
   as_error, StopSign!, Ok!, 1)
...
```

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z