Icetips PowerToolbar

10/14/2018

PowerToolbar

Published: October 2018

Publisher

Icetips Creative, Inc.

Managing Editor

Arnor Baldvinsson

Table of Contents

Part I

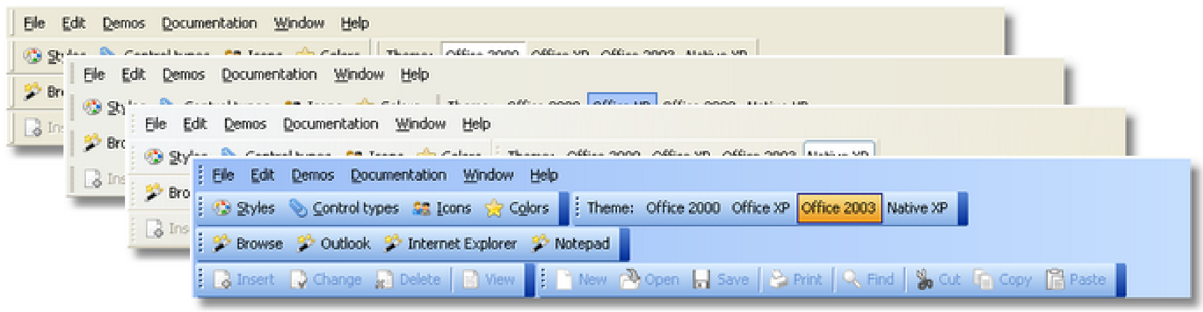
Chapter 1 - Introduction

1 Introduction

Welcome to Icetips PowerToolbar

- the most powerful Clarion toolbar ever

With PowerToolbar you can easily create Office toolbars in Clarion. Within minutes you can have Office 2000, Office XP, Office 2003 or Native XP style toolbars in your Clarion application.



Key features:

- Office style toolbar
- Office 2000, Office XP, Office 2003 and Native XP styles.
- Office style menu (supports Window list)
- AppFrame toolbar support
- Toolbar's anywhere in a window
- Toolbar controls: Button, CheckButton, RadioButton, MimicButton, DropButton, DropCombo, Entry, Text, Separator, Spacer
- Customize popup
- MDI Client template for toolbar control across threads

Please review the Limitations-section. Note updates on Clarion 7 and 8 limitations.

If you want to dive into the template right away, please read the Quick start section.

A detailed description of all template prompts can be found here.
For a complete method reference, go here.
Embed-points are described here.

Note that Icetips Alta LLC took this product over from PowerOffice in Norway in December 2008. It is possible that there are some references to PowerOffice in this text or in the source codes. If you find references to PowerOffice, please let us know.

Part II

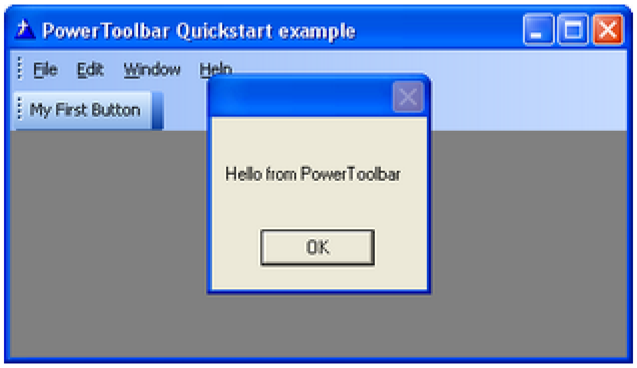
Chapter 2 - Tutorials

2 Tutorials

2.1 Quick start

This tutorial takes you through the steps to create a simple PowerToolbar application.

The finished application is installed at: Clarion6\3rdParty\Examples\PowerToolbar\Quickstart.app



Quick start tutorial

1. Create an empty application (ABC or Legacy).

2. Add the global template to your application (PowerToolbarGlobal - Icetips Power Toolbar Global)

- Press the "Global"-button
- Click "Extensions"
- Click "Insert"
- Choose "PowerToolbarGlobal - Icetips Power Toolbar Global"
- Click "Ok" to close global settings.

3. Add an AppFrame to your application

- Double-click the Main procedure
- Choose "Frame - Multiple Document Main Menu"
- Click the "Window"-button
- Choose "Application Main MDI Frame"

4. Add a toolbar to your AppFrame

- Select "Toolbar -> New Toolbar" on the menu

5. Populate the toolbar control

- Click the "Control Template" toolbutton
- Choose the template "POToolbar - Toolbar Control"
- Click somewhere inside the Clarion toolbar to position the PowerToolbar-control.

6. Open PowerToolbar settings

- Right-click the PowerToolbar-control and select "Actions"

7. Add a toolbar-band

- Press the "Insert"-button
- Enter a name for the band: "Quick start Band" (without the quotes)
- When you tab to the next field, the an ID will be generated (TB1_QuickstartBand)

8. Add a control to the band

- Press the "Insert"-button below the Controls list.
- In the control-settings dialog, enter the following Name: "My First Button" (without the quotes)
- The generated ID will be ID_MyFirstButton
- You can choose your favourite icon and enter some tooltip text if you'd like to.

9. Add some action for the button

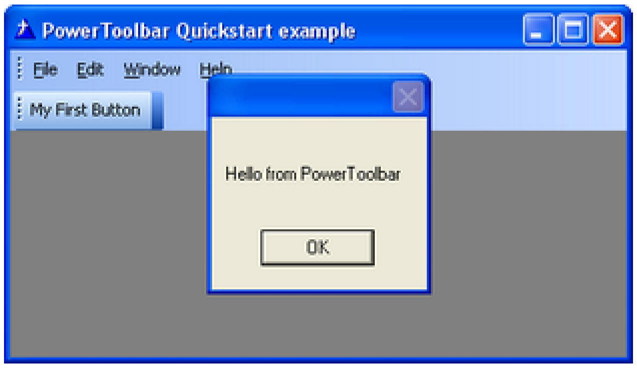
- Choose the "Action"-tab
- Press the "Embeds"-button
- Double-click the "Local Objects -> Toolbar1 -> ID_MyFirstButton -> Accepted"-event
- Choose "Source" as Embed type
- Enter the following text:

```
Message('Hello from PowerToolbar')
```

- Exit and save the source
- Press "Close" and "Ok" three times and close the window formatter to get back to the main Clarion IDE.

10. Compile and run your application

The result should be something similar to the screen shot below:

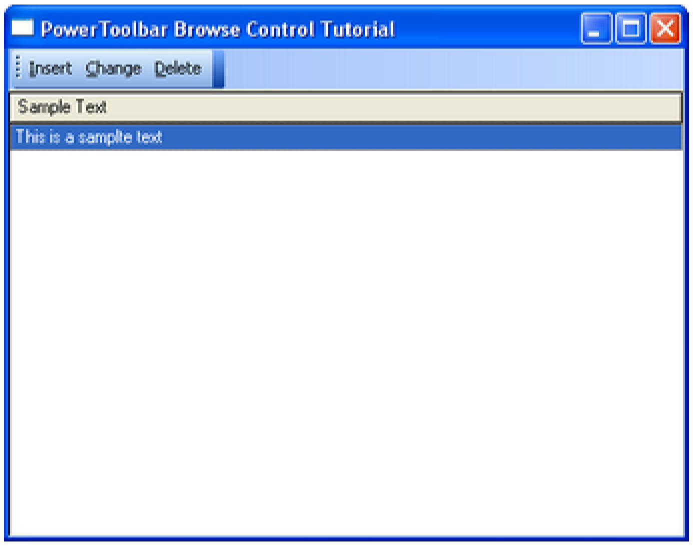


This example application can be found in the "Clarion6\3rdparty\Examples\PowerToolbar\" folder.

2.2 Simple Browse

In this tutorial you will learn how to use PowerToolbar to easily control your browses. This tutorial explains how to use a toolbar to control a browse in the same window. For advanced control across multiple threads, see the Advanced Browse tutorial.

The example below will set up a window with a browse and a toolbar to update it. You can find the finished application at: Clarion6\3rdParty\Examples\PowerToolbar\BrowseTutorial.app

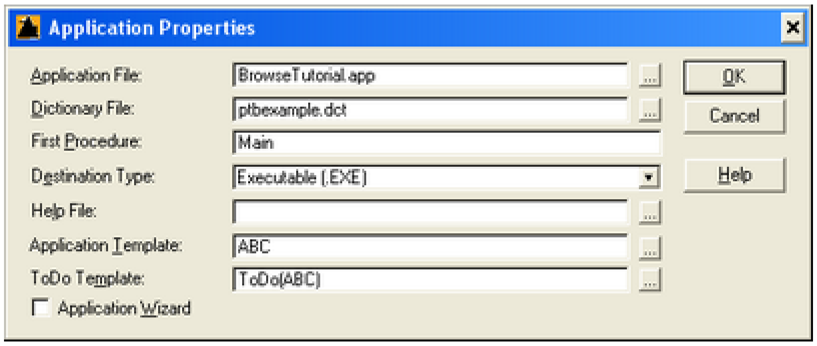


In this example you should use the dictionary named *ptbexamples.dct*

Simple Browse control tutorial

Create a new application (ABC)

- Choose the ptbexample.dct from Clarion6\3rdParty\Examples\PowerToolbar as "Dictionary File".



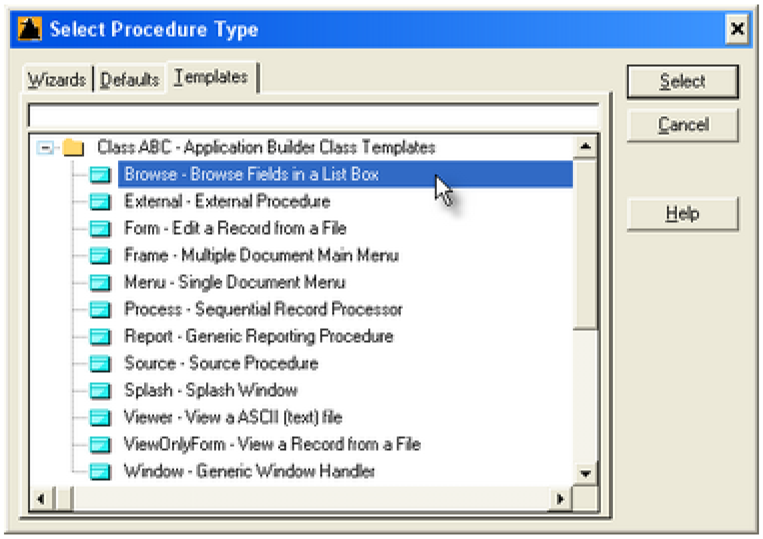
Application File:	BrowseTutorial.app
Dictionary File:	ptbexample.dct
First Procedure:	Main
Destination Type:	Executable (.EXE)
Help File:	
Application Template:	ABC
ToDo Template:	ToDo(ABC)

☐ Application Wizard

Add the global PowerToolbar template

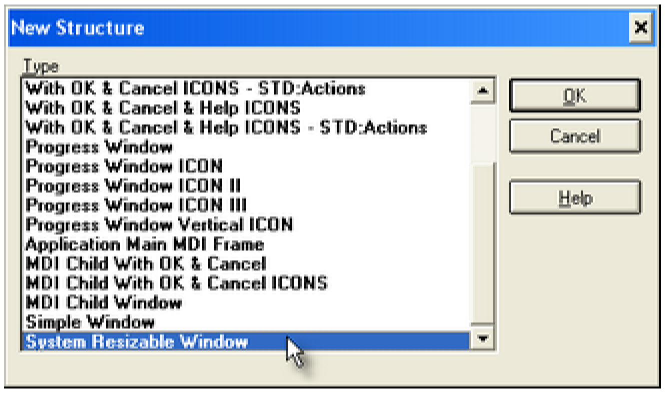
- See Adding the templates

Double click the "Main (Todo)"-procedure and choose "Browse - Browse Fields in a List Box" as template type



Class ABC - Application Builder Class Templates
Browse - Browse Fields in a List Box
External - External Procedure
Form - Edit a Record from a File
Frame - Multiple Document Main Menu
Menu - Single Document Menu
Process - Sequential Record Processor
Report - Generic Reporting Procedure
Source - Source Procedure
Splash - Splash Window
Viewer - View a ASCII (text) file
ViewOnlyForm - View a Record from a File
Window - Generic Window Handler

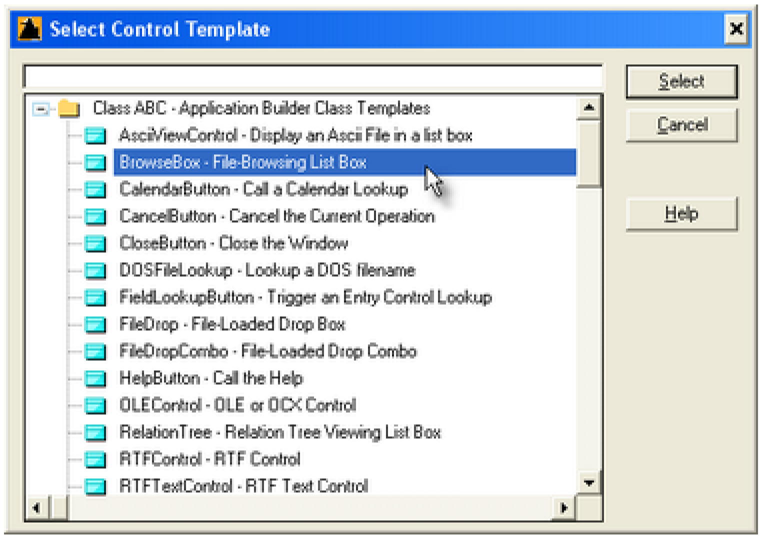
Click the "Window" button and select "System Resizable Window" as window type.



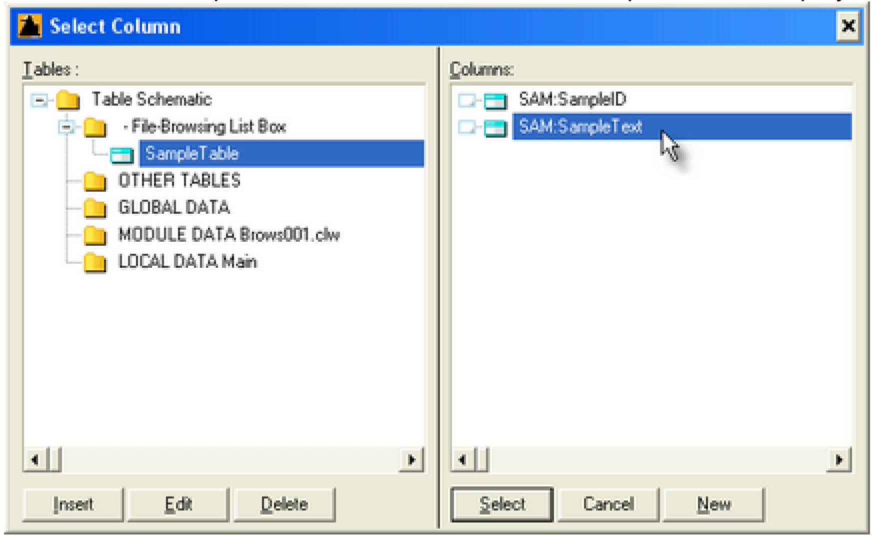
With OK & Cancel ICONS - STD:Actions
With OK & Cancel & Help ICONS
With OK & Cancel & Help ICONS - STD:Actions
Progress Window
Progress Window ICON
Progress Window ICON II
Progress Window ICON III
Progress Window Vertical ICON
Application Main MDI Frame
MDI Child With OK & Cancel
MDI Child With OK & Cancel ICONS
MDI Child Window
Simple Window
System Resizable Window

You should now be in the window formatter.

- Click the "Control template" button ( )
- Choose the "Browse Box - File-Browsing List Box"-template


Select Control Template
Select
Cancel
Help
Class ABC - Application Builder Class Templates
AsciiViewControl - Display an Ascii File in a list box
BrowseBox - File-Browsing List Box
CalendarButton - Call a Calendar Lookup
CancelButton - Cancel the Current Operation
CloseButton - Close the Window
DOSFileLookup - Lookup a DOS filename
FieldLookupButton - Trigger an Entry Control Lookup
FileDrop - File-Loaded Drop Box
FileDropCombo - File-Loaded Drop Combo
HelpButton - Call the Help
OLEControl - OLE or OCX Control
RelationTree - Relation Tree Viewing List Box
RTFControl - RTF Control
RTFTextControl - RTF Text Control

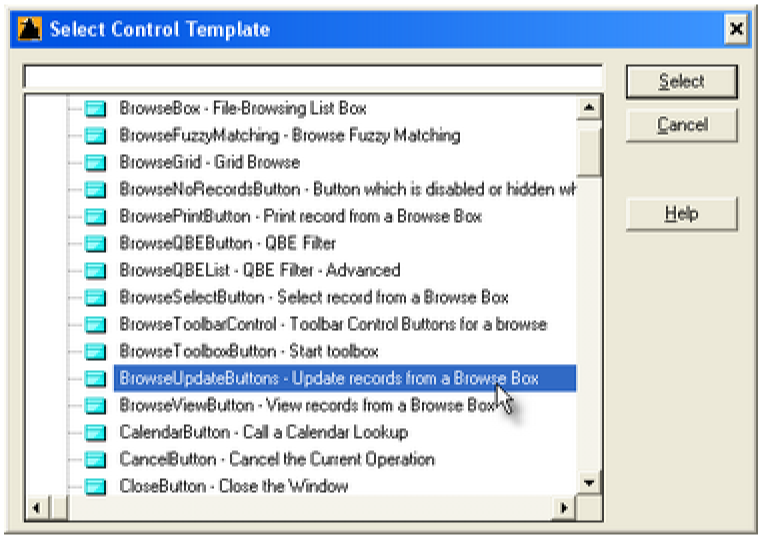
- Place it somewhere in your window
- Choose the "Sample Table" as table, and "SAM:SampleText" as display field.



- Click "Ok" in the list box formatter

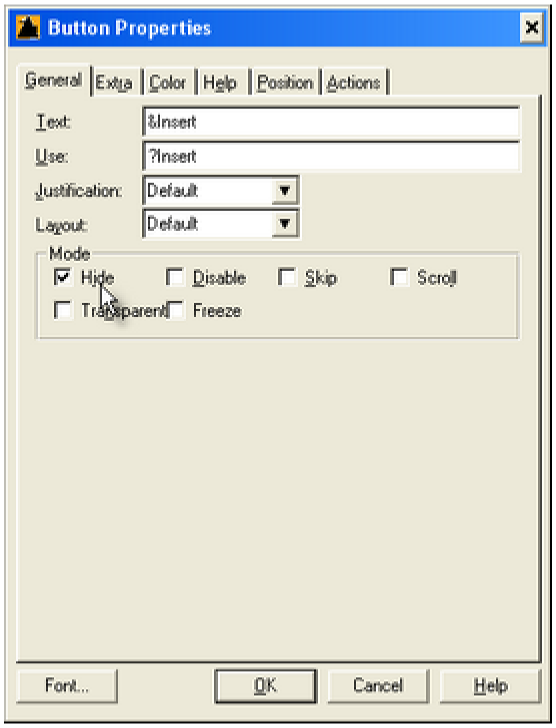
Now you should add browse update buttons

- Click the "Control template" button ( )
- Choose the "BrowseUpdateButtons - Update records from a Browse Box"-template



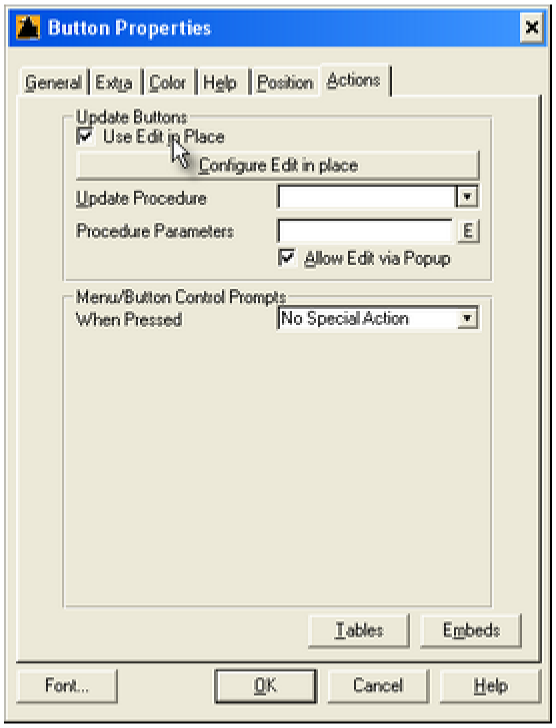
BrowseBox - File-Browsing List Box
BrowseFuzzyMatching - Browse Fuzzy Matching
BrowseGrid - Grid Browse
BrowseNoRecordsButton - Button which is disabled or hidden wh
BrowsePrintButton - Print record from a Browse Box
BrowseQBEButton - QBE Filter
BrowseQBEList - QBE Filter - Advanced
BrowseSelectButton - Select record from a Browse Box
BrowseToolbarControl - Toolbar Control Buttons for a browse
BrowseToolboxButton - Start toolbox
BrowseUpdateButtons - Update records from a Browse Box
BrowseViewButton - View records from a Browse Box
CalendarButton - Call a Calendar Lookup
CancelButton - Cancel the Current Operation
CloseButton - Close the Window

- Position the buttons over the browse box
- Set buttons to "Hide"



Text: &Insert
Use: ?Insert
Justification: Default
Layout: Default
Mode
[✓ Hide] Disable Skip Scroll
Transparent Freeze

- On one of the button's action-tab, check the "Use Edit in-Place" option



Update Buttons
☑ Use Edit in Place
Update Procedure
Procedure Parameters
☑ Allow Edit via Popup
Menu/Button Control Prompts
When Pressed: No Special Action

Add PowerToolbar to your window

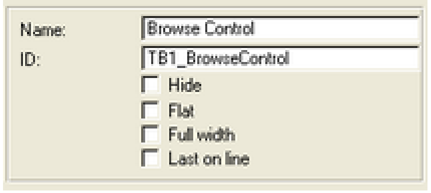
- Click the "Control template" button ( )
- Choose the "POToolbarControl - Toolbar Control"-template

Class POToolbar - PowerOffice Toolbar
POToolbarControl - Toolbar Control

- Position at the top of your window
- Right-click the toolbar control, and choose "Position"
- Set Width to "Full"

Add toolbar band

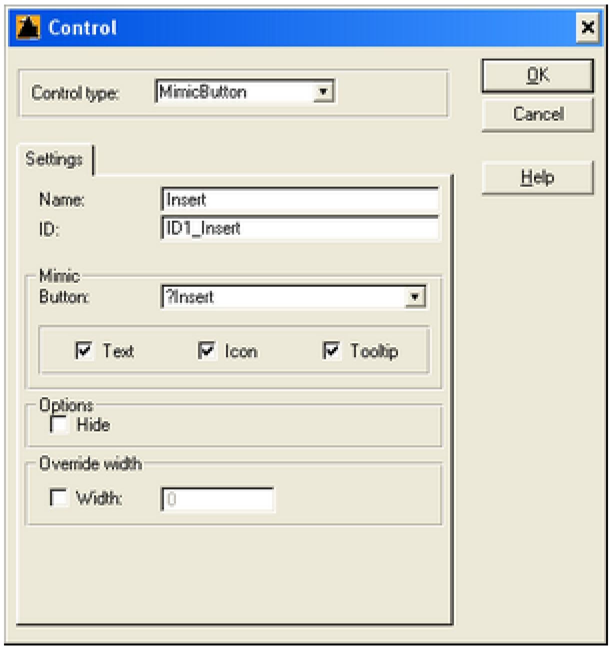
- Right-click the toolbar control, and choose "Actions"
- Click the "Insert" button to add a band
- Enter "Browse Control" as band name



Name: Browse Control
ID: TB1_BrowseControl
☐ Hide
☐ Flat
☐ Full width
☐ Last on line

Add toolbar buttons

- Click the "Insert" button to add a button
- Choose "MimicButton" as "Control type"
- Enter "Insert" as "Name"
- In the "Mimic Button"-dropdown, choose the "?Insert"-button



Control

Control type: MimicButton

Settings

Name: Insert

ID: ID1_Insert

Mimic Button: ?Insert

[✓] Text [✓] Icon [✓] Tooltip

Options

[ ] Hide

Override width

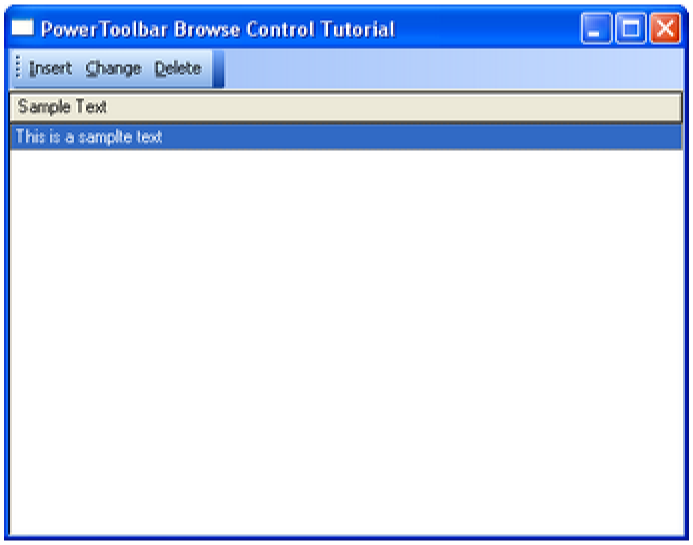
[ ] Width: 0

- Click "Ok"
- **Now, repeat this step for the ?Change and ?Delete button**

Save and compile your application

After starting your application, you can click the "insert" button to add a new record.

In this tutorial we used "Edit in-Place", but this method works just as well if you use an update form instead.



PowerToolbar Browse Control Tutorial

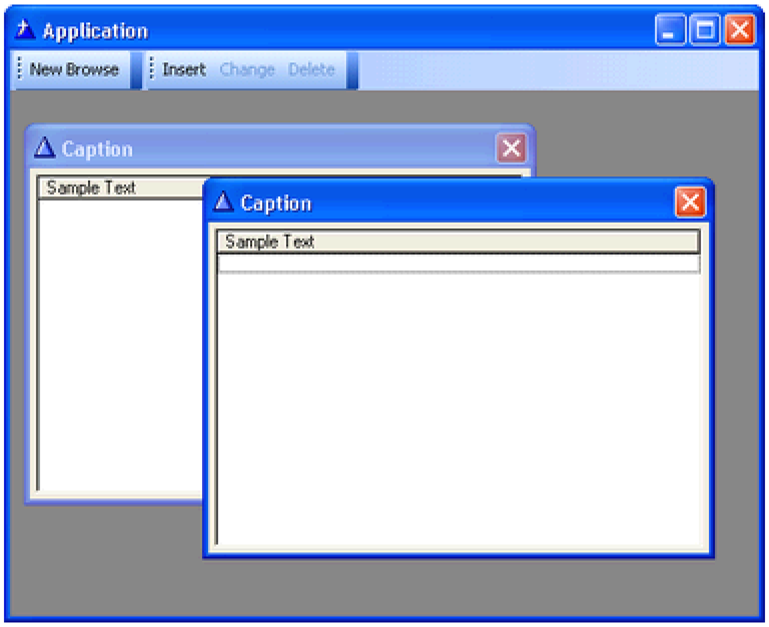
Insert Change Delete

Sample Text
This is a samplte text

2.3 Advanced Browse

This tutorial shows how to create a browse and control it from the AppFrame toolbar. For a tutorial on how to control a browse with a toolbar in the same window as the browse, click here.

You can find the finished application can be found at:
Clarion6\3rdParty\Examples\PowerToolbar\AdvancedBrowseTutorial.app

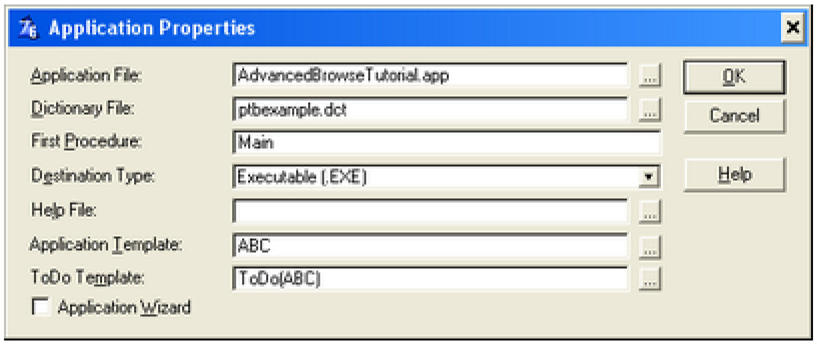


In this example you should use the dictionary named *ptbexamples.dct*

Part 1 - Populate templates

Create a new application (ABC)

- Choose the ptbexample.dct from Clarion6\3rdParty\Examples\PowerToolbar as "Dictionary File".


Application Properties

Application File:	AdvancedBrowseTutorial.app
Dictionary File:	ptbexample.dct
First Procedure:	Main
Destination Type:	Executable (.EXE)
Help File:	
Application Template:	ABC
ToDo Template:	ToDo(ABC)

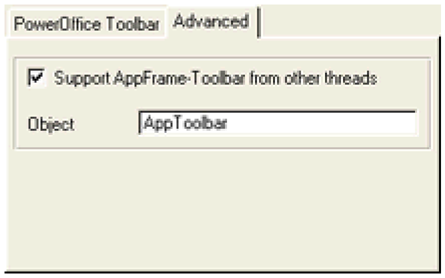
☐ Application Wizard

Add the global PowerToolbar template

- See Adding the templates

Select the "Advanced"-tab of the global template

- Make sure "Support AppFrame-Toolbar from other threads" is checked.



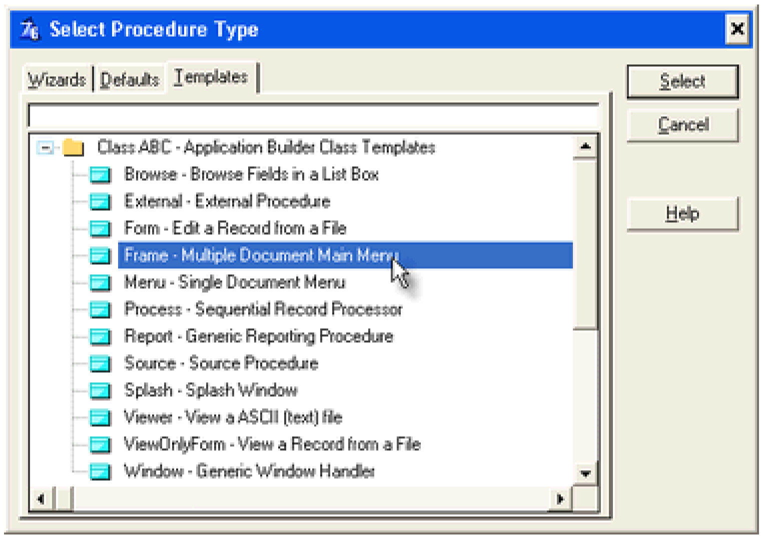
☑ Support AppFrame-Toolbar from other threads

Object: AppToolbar

- Click "Ok" twice to get back to the Application Tree.

Add AppFrame

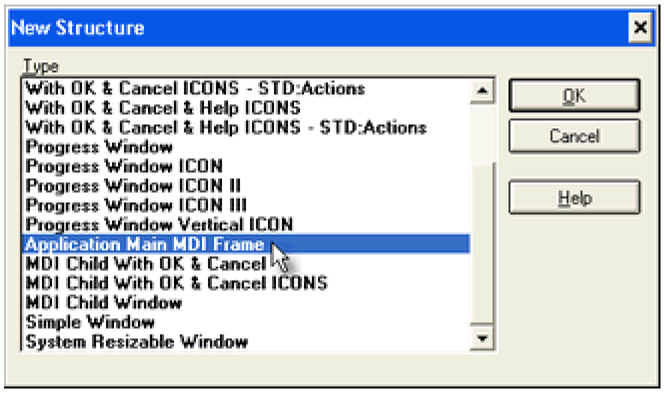
- Double-click the Main(Todo) procedure to bring up the "Select Procedure Type" dialog.
- Select "Frame - Multiple Document Main Menu"



Class ABC - Application Builder Class Templates
- Browse - Browse Fields in a List Box
- External - External Procedure
- Form - Edit a Record from a File
- Frame - Multiple Document Main Menu
- Menu - Single Document Menu
- Process - Sequential Record Processor
- Report - Generic Reporting Procedure
- Source - Source Procedure
- Splash - Splash Window
- Viewer - View a ASCII (text) file
- ViewOnlyForm - View a Record from a File
- Window - Generic Window Handler

Create the window

- Click the Window button
- Select "Application Main MDI Frame" and click OK. This will bring up the window formatter.



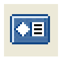
With OK & Cancel ICONS - STD:Actions
With OK & Cancel & Help ICONS
With OK & Cancel & Help ICONS - STD:Actions
Progress Window
Progress Window ICON
Progress Window ICON II
Progress Window ICON III
Progress Window Vertical ICON
Application Main MDI Frame
MDI Child With OK & Cancel
MDI Child With OK & Cancel ICONS
MDI Child Window
Simple Window
System Resizable Window

Delete the menubar (We do not want a menu for this tutorial)

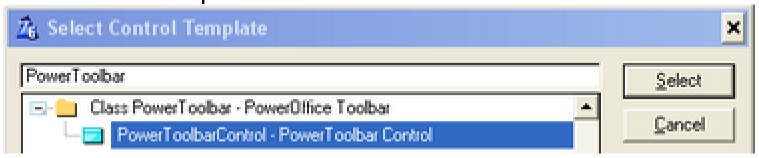
- Select the menu-item "Menu -> Delete Menu"

Create the toolbar
- Select the menu-item "Toolbar -> New Toolbar"

Populate PowerToolbar control
- Click the "Control Template" toolbutton



- Choose the template "POToolbar - Toolbar Control"



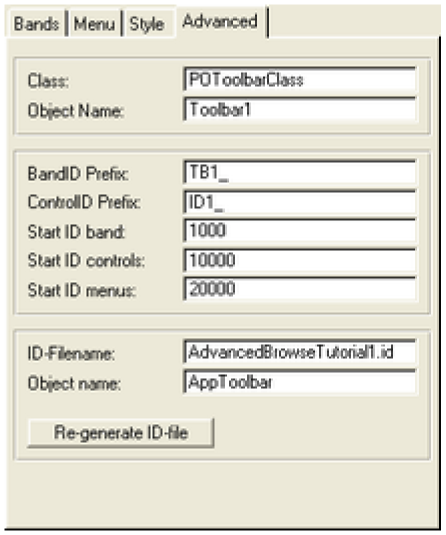
PowerToolbar
Class PowerToolbar - PowerOffice Toolbar
PowerToolbarControl - PowerToolbar Control

- Click somewhere inside the Clarion toolbar or Window to place the PowerToolbar-control.

Part 2 - Toolbar settings

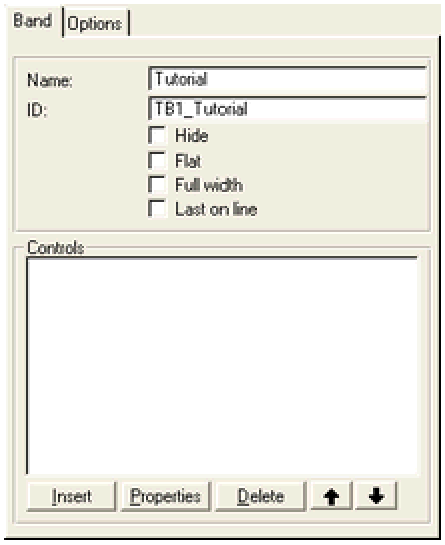
Right-click the PowerToolbar control, and select "Actions"

Select the "Advanced" tab
- Notice "ID-Filename". All ID's for this toolbar will be exported to this file. You will need to select this file in the MDI Client template later in this tutorial.
- "Object name" must be the same as global template setting.



Class:	POToolbarClass
Object Name:	Toolbar1
BandID Prefix:	TB1_
ControlID Prefix:	ID1_
Start ID band:	1000
Start ID controls:	10000
Start ID menus:	20000
ID-Filename:	AdvancedBrowseTutorial1.id
Object name:	AppToolbar

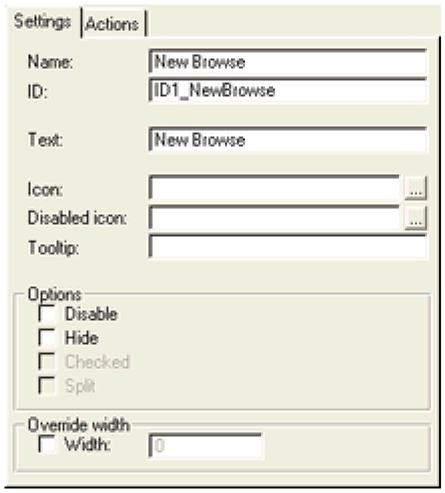
Add a band
- Select the "Bands"-tab
- Click the "Insert"-button
- As "Name", type "Tutorial"



Name: Tutorial
ID: TB1_Tutorial
☐ Hide
☐ Flat
☐ Full width
☐ Last on line

Add a button to start a browse

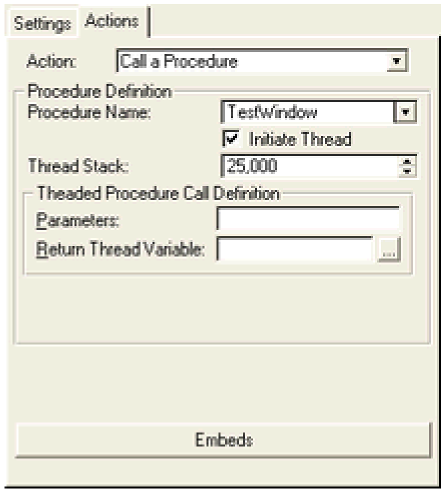
- Click the "Insert"-button
- Type "New Browse" in the Name field.



Name: New Browse
ID: ID1_NewBrowse
Text: New Browse
Icon:
Disabled icon:
Tooltip:
☐ Disable
☐ Hide
☐ Checked
☐ Split
☐ Width: 0

Set button action

- Select the "Actions"-tab
- As "Action" select "Call a Procedure"
- Enter "TestWindow" as procedure name
- Check the "Initiate thread" checkbox



Action: Call a Procedure

Procedure Definition

Procedure Name: TestWindow

[✓] Initiate Thread

Thread Stack: 25,000

Theaded Procedure Call Definition

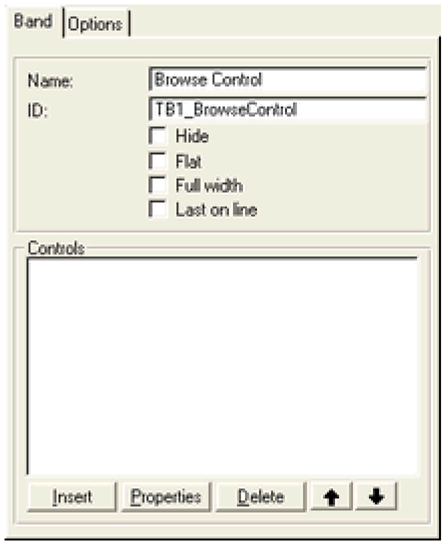
Parameters:

Return Thread Variable:

- Click "Ok" two times, to get back to "Bands"

Add new band

- Click the "Insert"-button
- As "Name", type "Browse Control"



Name: Browse Control

ID: TB1_BrowseControl

[ ] Hide

[ ] Flat

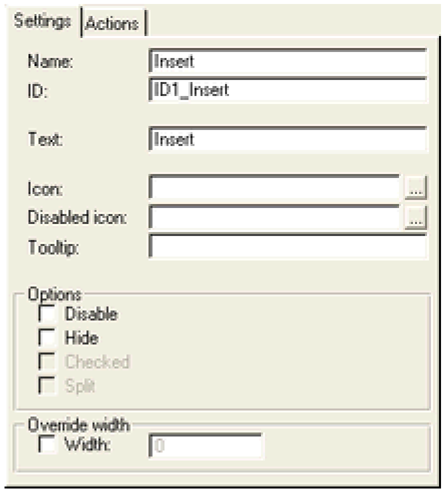
[ ] Full width

[ ] Last on line

Controls

Add browse "Insert" button

- Click the "Insert"-button
- Type "Insert" in the Name field.
- Notice the ID generated (ID1_Insert)
- Check "Disable" to make the button disabled by default


Settings | Actions

Name: Insert
ID: ID1_Insert

Text: Insert

Icon:
Disabled icon:
Tooltip:

Options
- [ ] Disable
- [ ] Hide
- [ ] Checked
- [ ] Split

Override width
- [ ] Width: 0

- Click "Ok" to save the new button

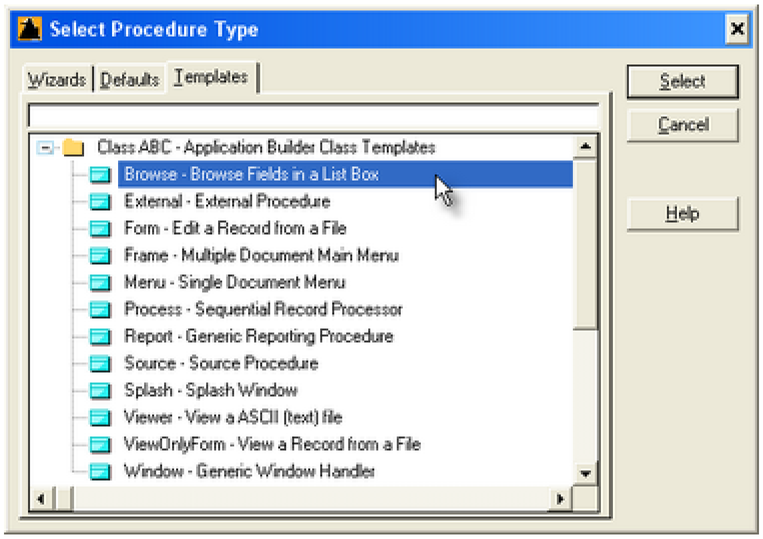
Add browse "Change" button
- Click the "Insert"-button (below the list)
- Type "Change" in the Name field.
- Notice the ID generated (ID1_Change)
- Check "Disable" to make the button disabled by default
- Click "Ok" to save the new button

Add browse "Delete" button
- Click the "Insert"-button (below the list)
- Type "Delete" in the Name field.
- Notice the ID generated (ID1_Delete)
- Check "Disable" to make the button disabled by default
- Click "Ok" to save the new button
- Click "Ok" 2 times to get back to the window formatter
- Save and exit the window

Compile the application
- We need to compile (or at least generate) the application to generate the ID-file (AdvancedBrowseTutorial1.id)
- Compile the application (just quit it right away, we got more work to do :-)

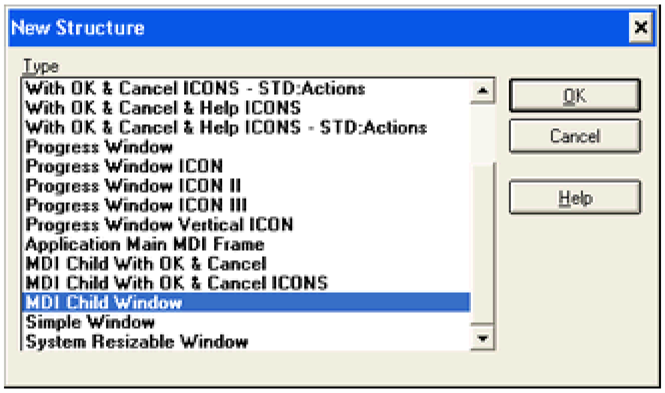
Part 3 - Create the browse

Double click the "TestWindow (Todo)"-procedure and choose "Browse - Browse Fields in a List Box" as template type



Class ABC - Application Builder Class Templates
- Browse - Browse Fields in a List Box
- External - External Procedure
- Form - Edit a Record from a File
- Frame - Multiple Document Main Menu
- Menu - Single Document Menu
- Process - Sequential Record Processor
- Report - Generic Reporting Procedure
- Source - Source Procedure
- Splash - Splash Window
- Viewer - View a ASCII (text) file
- ViewOnlyForm - View a Record from a File
- Window - Generic Window Handler

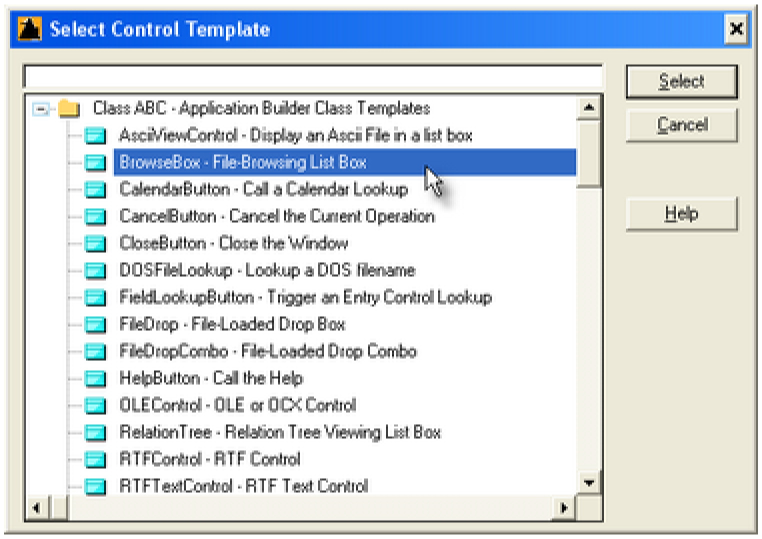
Click the "Window" button and select "MDI Child Window" as window type.



With OK & Cancel ICONS - STD:Actions
With OK & Cancel & Help ICONS
With OK & Cancel & Help ICONS - STD:Actions
Progress Window
Progress Window ICON
Progress Window ICON II
Progress Window ICON III
Progress Window Vertical ICON
Application Main MDI Frame
MDI Child With OK & Cancel
MDI Child With OK & Cancel ICONS
MDI Child Window
Simple Window
System Resizable Window

You should now be in the window formatter.

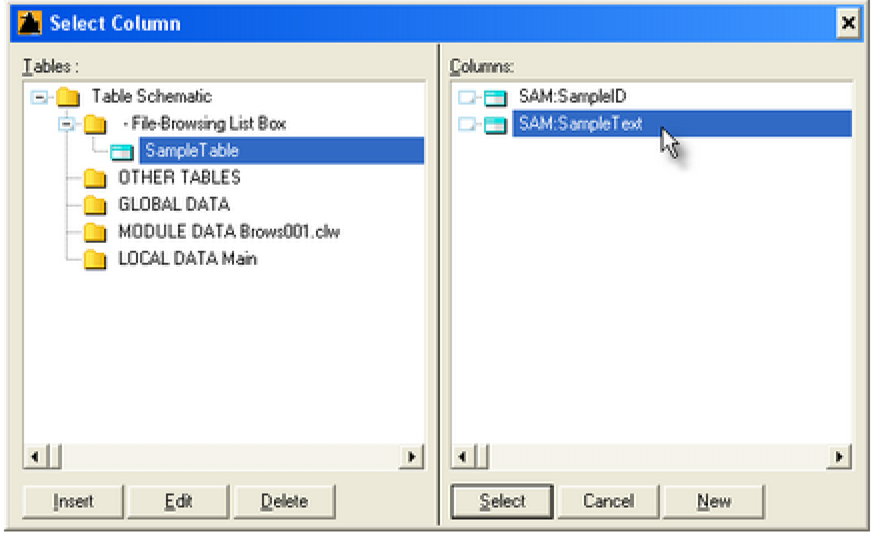
- Click the "Control template" button ( )
- Choose the "Browse Box - File-Browsing List Box"-template



Select Control Template

Class ABC - Application Builder Class Templates
- AsciiViewControl - Display an Ascii File in a list box
- BrowseBox - File-Browsing List Box
- CalendarButton - Call a Calendar Lookup
- CancelButton - Cancel the Current Operation
- CloseButton - Close the Window
- DOSFileLookup - Lookup a DOS filename
- FieldLookupButton - Trigger an Entry Control Lookup
- FileDrop - File-Loaded Drop Box
- FileDropCombo - File-Loaded Drop Combo
- HelpButton - Call the Help
- OLEControl - OLE or OCX Control
- RelationTree - Relation Tree Viewing List Box
- RTFControl - RTF Control
- RTFTextControl - RTF Text Control

■ Place it somewhere in your window
■ Choose the "Sample Table" as table, and "SAM:SampleText" as display field.



Select Column

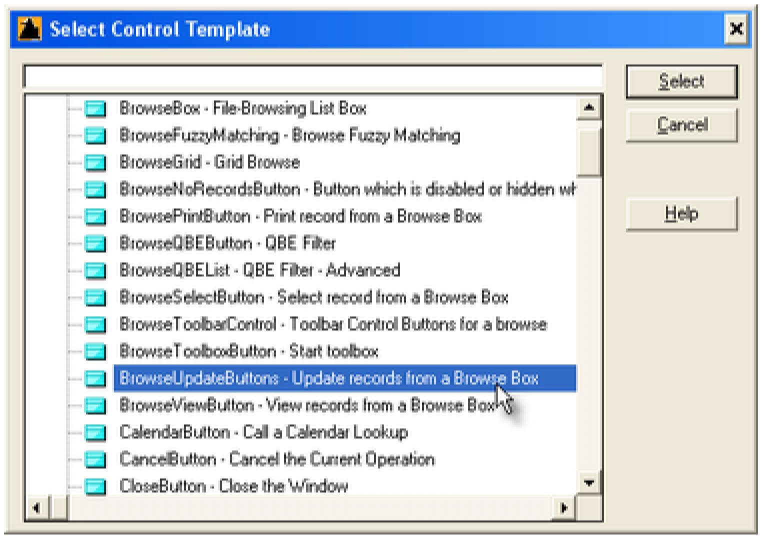
Tables :
- Table Schematic
  - - File-Browsing List Box
    - SampleTable
  - OTHER TABLES
  - GLOBAL DATA
  - MODULE DATA Brows001.clw
  - LOCAL DATA Main

Columns:
- SAM:SampleID
- SAM:SampleText

■ Click "Ok" in the list box formatter

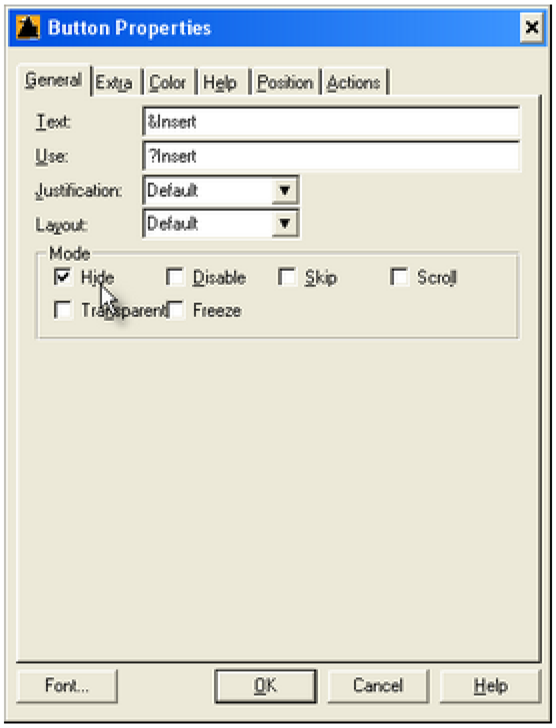
Now you should add browse update buttons

■ Click the "Control template" button ( )
■ Choose the "BrowseUpdateButtons - Update records from a Browse Box"-template



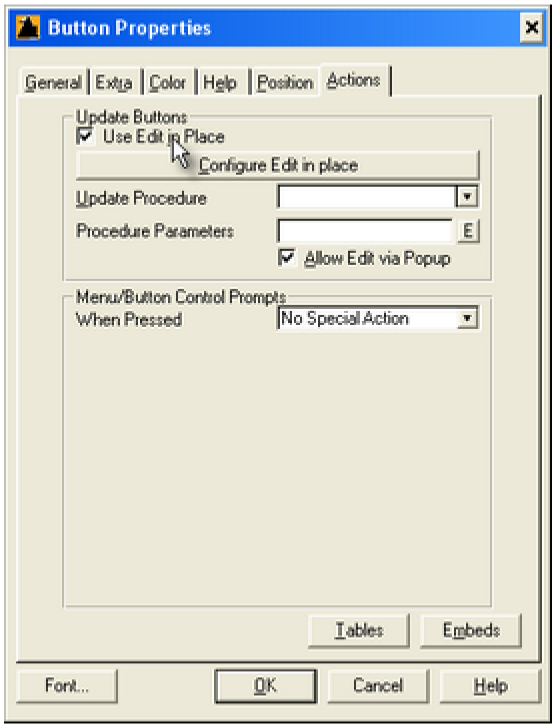
BrowseBox - File-Browsing List Box
BrowseFuzzyMatching - Browse Fuzzy Matching
BrowseGrid - Grid Browse
BrowseNoRecordsButton - Button which is disabled or hidden wh
BrowsePrintButton - Print record from a Browse Box
BrowseQBEButton - QBE Filter
BrowseQBEList - QBE Filter - Advanced
BrowseSelectButton - Select record from a Browse Box
BrowseToolbarControl - Toolbar Control Buttons for a browse
BrowseToolboxButton - Start toolbox
BrowseUpdateButtons - Update records from a Browse Box
BrowseViewButton - View records from a Browse Box
CalendarButton - Call a Calendar Lookup
CancelButton - Cancel the Current Operation
CloseButton - Close the Window

- Position the buttons over the browse box
- Set buttons to "Hide"



Text: &Insert
Use: ?Insert
Justification: Default
Layout: Default
Mode
[✓ Hide] Disable Skip Scroll
Transparent Freeze

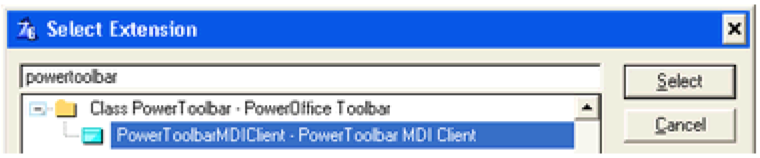
- On one of the button's action-tab, check the "Use Edit in-Place" option



Part 4 - MDI Client template

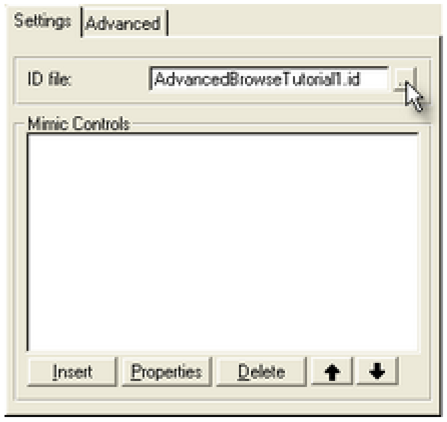
Add the MDI-template

- Click the "Extensions"-button
- Click "Insert"
- Select "PowerToolbarMDIClient - PowerToolbar MDI Client"


Select Extension
powertoolbar
Class PowerToolbar - PowerOffice Toolbar
PowerToolbarMDIClient - PowerToolbar MDI Client
Select
Cancel

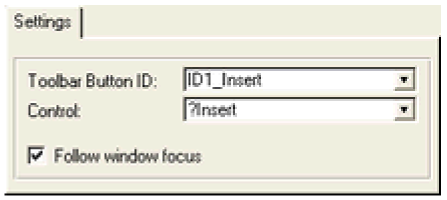
Set up the template

- Click the "..." button next to the ID-file prompt.
- Select the "AdvancedBrowseTutorial1.id"-file. This is the file generated by the toolbar-control i the AppFrame.



ID file: AdvancedBrowseTutorial1.id

Add button mimic for browse "Insert" button
- Click "Insert" to add mimic control
- Choose "ID1_Insert" as "Toolbar Button ID"
- Choose "?Insert" as Control.
- We have now established a link between the toolbar button, and the insert browse update button.

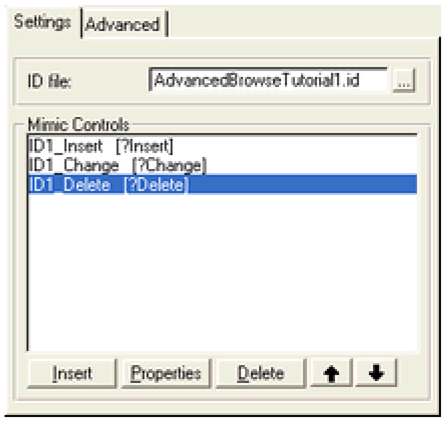


Toolbar Button ID: ID1_Insert
Control: ?Insert
[✓] Follow window focus

- Click "Ok"

Add button mimic for browse "Change" button
- Click "Insert" (below the list box) to add mimic control
- Choose "ID1_Change" as "Toolbar Button ID"
- Choose "?Change" as Control.
- Click "Ok"

Add button mimic for browse "Delete" button
- Click "Insert" (below the list box) to add mimic control
- Choose "ID1_Delete" as "Toolbar Button ID"
- Choose "?Delete" as Control.
- Click "Ok"



ID file: AdvancedBrowseTutorial1.id

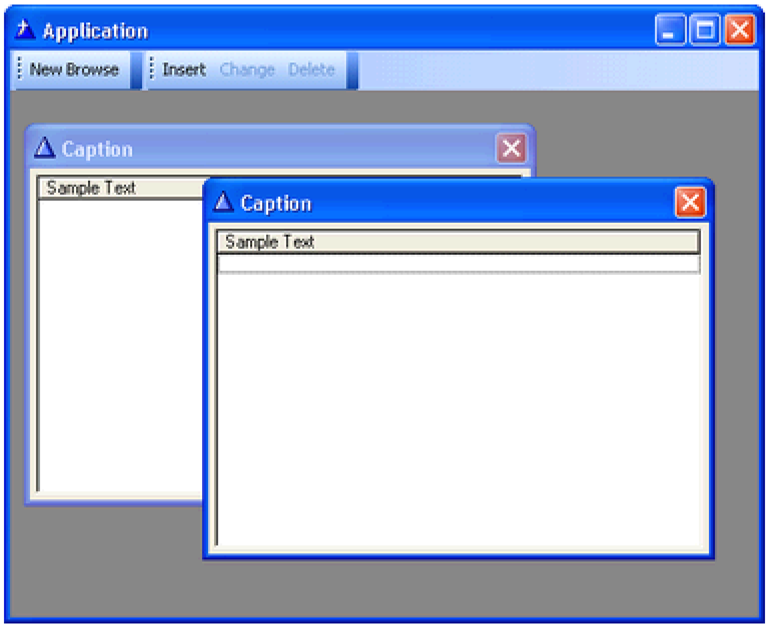
ID1_Insert [?Insert]
ID1_Change [?Change]
ID1_Delete [?Delete]

Part 5 - Try application

Compile and run

- Click "Ok" to get back to the application tree
- Compile and run the application

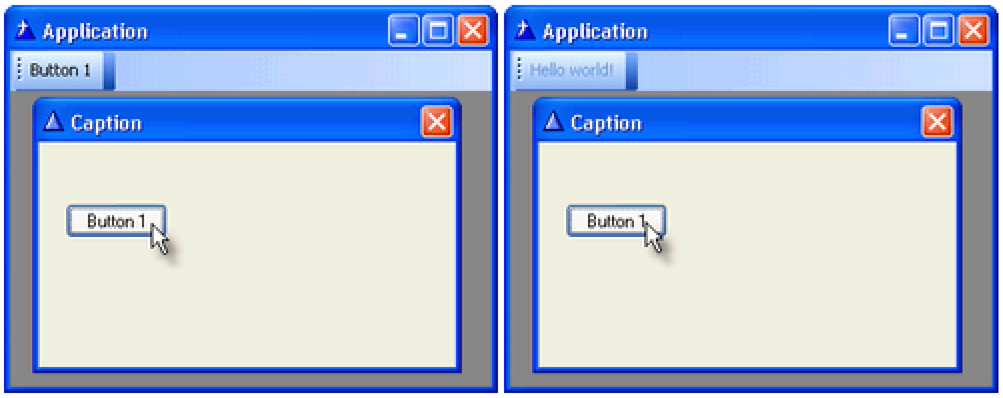
- Try opening and closing a browse
- Try several browses open and controling them from the toolbar



2.4 AppToolbar control from a different thread

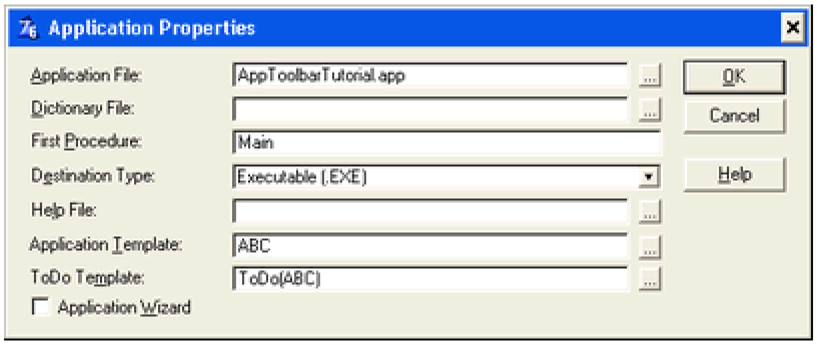
It is possible to control the AppFrame PowerToolbar from any thread in your application. In this tutorial you will be taken through the steps needed.

The finished application for this tutorial can be found at:
Clarion6\3rdParty\Examples\PowerToolbar\AppToolbarTutorial.app



Part 1 - Populate templates

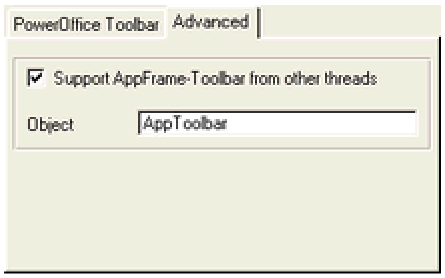
Create a new application


Application Properties

Application File:	AppToolbarTutorial.app
Dictionary File:	
First Procedure:	Main
Destination Type:	Executable (.EXE)
Help File:	
Application Template:	ABC
ToDo Template:	ToDo(ABC)
Application Wizard	☐

Add the global PowerToolbar template
See Adding the templates

Go to the "Advanced" tab
▪ On the advanced tab of the global template, make sure "Support AppFrame-Toolbar from other threads" is checked.

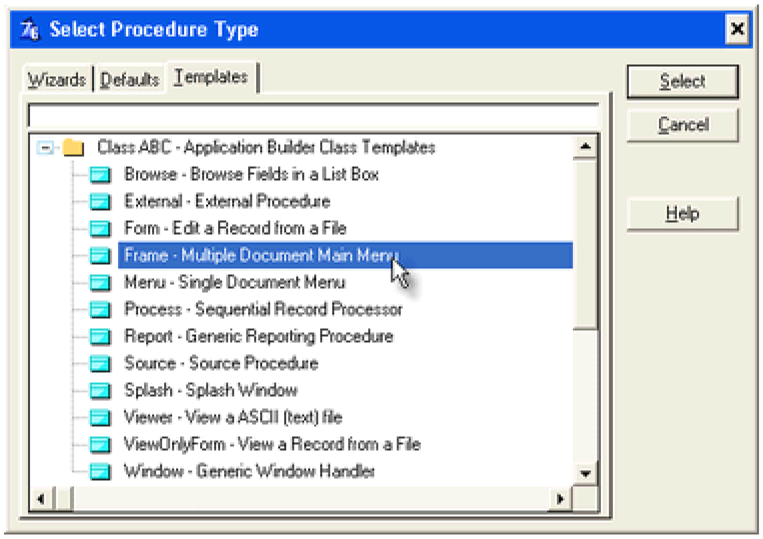


☑ Support AppFrame-Toolbar from other threads
Object: AppToolbar

- Notice the Object name (AppToolbar). This is what we will refer to in our code later on.
- Click "Ok" twice to get back to the Application Tree.

Add AppFrame

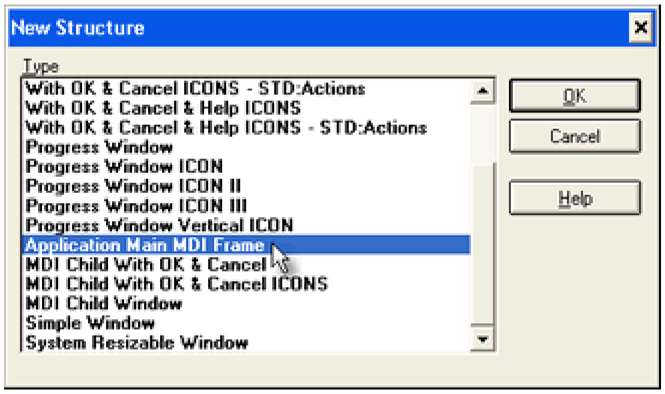
- Double-click the Main(Todo) procedure to bring up the "Select Procedure Type" dialog.
- Select "Frame - Multiple Document Main Menu"



Class ABC - Application Builder Class Templates
Browse - Browse Fields in a List Box
External - External Procedure
Form - Edit a Record from a File
Frame - Multiple Document Main Menu
Menu - Single Document Menu
Process - Sequential Record Processor
Report - Generic Reporting Procedure
Source - Source Procedure
Splash - Splash Window
Viewer - View a ASCII (text) file
ViewOnlyForm - View a Record from a File
Window - Generic Window Handler

Create the window

- Click the Window button
- Select "Application Main MDI Frame" and click OK. This will bring up the window formatter.



With OK & Cancel ICONS - STD:Actions
With OK & Cancel & Help ICONS
With OK & Cancel & Help ICONS - STD:Actions
Progress Window
Progress Window ICON
Progress Window ICON II
Progress Window ICON III
Progress Window Vertical ICON
Application Main MDI Frame
MDI Child With OK & Cancel
MDI Child With OK & Cancel ICONS
MDI Child Window
Simple Window
System Resizable Window

Delete the menubar (We do not want a menu for this tutorial)

- Select the menu-item "Menu -> Delete Menu"

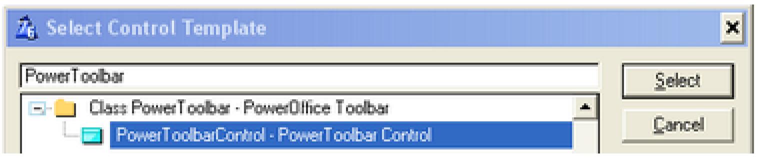
Create the toolbar

- Select the menu-item "Toolbar -> New Toolbar"

Populate PowerToolbar control

- Click the "Control Template" toolbutton

- Choose the template "POToolbar - Toolbar Control"


Select Control Template
PowerToolbar
Class PowerToolbar - PowerOffice Toolbar
PowerToolbarControl - PowerToolbar Control
Select
Cancel

■ Click somewhere inside the Clarion toolbar or Window to place the PowerToolbar-control.

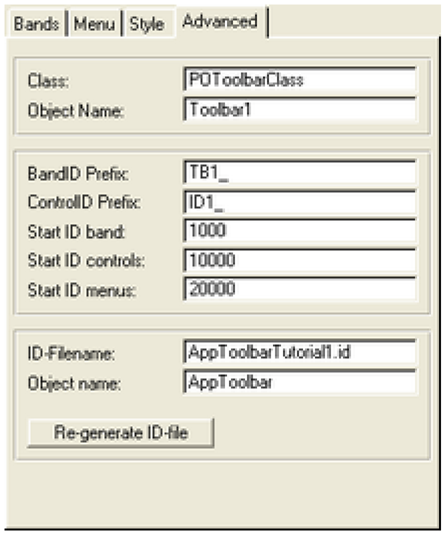
Part 2 - Toolbar settings

Right-click PowerToolbar control, and select "Actions"

Select the "Advanced" tab

■ Notice "ID-Filename". All ID's for this toolbar will be exported to this file. You will need to select this file in the MDI Client template later in this tutorial.

■ "Object name" must be the same as global template setting.



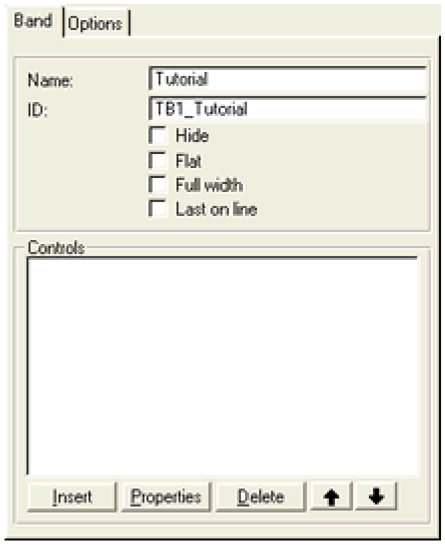
Class:	POToolbarClass
Object Name:	Toolbar1
BandID Prefix:	TB1_
ControlID Prefix:	ID1_
Start ID band:	1000
Start ID controls:	10000
Start ID menus:	20000
ID-Filename:	AppToolbarTutorial1.id
Object name:	AppToolbar

Add a band

■ Select the "Bands"-tab

■ Click the "Insert"-button

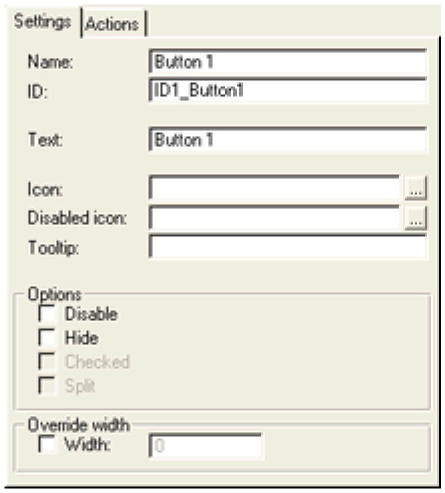
■ As "Name", type "Tutorial"



Name: Tutorial
ID: TB1_Tutorial
☐ Hide
☐ Flat
☐ Full width
☐ Last on line

Add a button

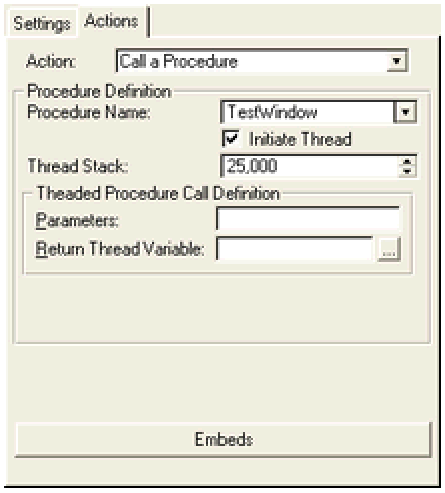
- Click the "Insert"-button
- Type "Button 1" in the Name field.



Name: Button 1
ID: ID1_Button1
Text: Button 1
Icon:
Disabled icon:
Tooltip:
☐ Disable
☐ Hide
☐ Checked
☐ Split
☐ Width: 0

Set button action

- Select the "Actions"-tab
- As "Action" select "Call a Procedure"
- Enter "TestWindow" as procedure name
- Check the "Initiate thread" checkbox



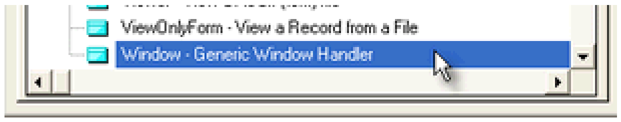
Action: Call a Procedure
Procedure Definition
Procedure Name: TestWindow
☑ Initiate Thread
Thread Stack: 25,000
Theaded Procedure Call Definition
Parameters:
Return Thread Variable:

- Click OK three times to get back to the window formatter
- Save and exit the window

Compile the application
- You must do this to generate the ID-file (AppToolbarTutorial1.id)

Part 3 - Set up Test window

Create the window
- Double-click the "TestWindow (ToDo)"-procedure
- Select "Window - Generic Window Handler"

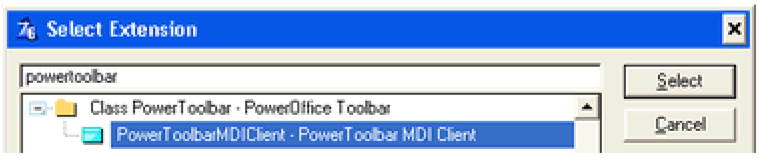


ViewOnlyForm - View a Record from a File
Window - Generic Window Handler

- Click the "Window button"
- Select "MDI Child Window"

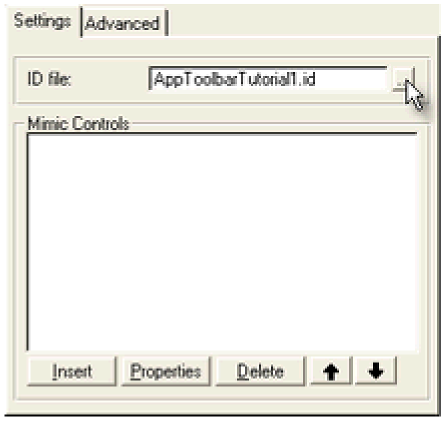
Save and exit the window formatter

Add the MDI-template
- Click the "Extensions"-button
- Click "Insert"
- Select "PowerToolbarMDIClient - PowerToolbar MDI Client"



powertoolbar
Class PowerToolbar - PowerOffice Toolbar
PowerToolbarMDIClient - PowerToolbar MDI Client

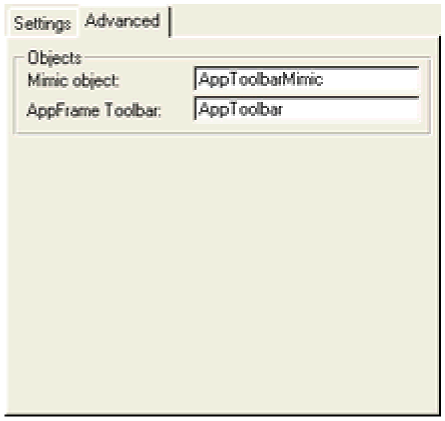
Set up the template
- Click the "..." button next to the ID-file prompt.
- Select the "AppToolbarTutorial1.id"-file. This is the file generated by the toolbar-control i the AppFrame.



ID file: AppToolbarTutorial1.id

▸ **Click the "Advanced"-tab**

▪ Notice "AppFame Toolbar". This is the object name that we saw in both the global template, and the AppFrame toolbar-control.



Mimic object: AppToolbarMimic

AppFrame Toolbar: AppToolbar

▸ **Click OK to get back to the Procedure Properties window.**

▸ **Click the "Window" button to get to the window formatter.**

▸ **Place a button on the window**

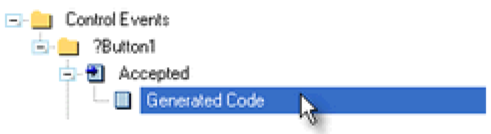
▪ Make sure "Use" is ?Button1

▸ **Code some actions for the toolbar**

▪ Select the "Actions"-tab

▪ Click the embed button

▪ Double click "Control Events -> ?Button1 -> Accepted -> Generated Code"



▪ Select "Source" as Embed type

▸ **Enter code to manipulate the toolbar**

▪ `AppToolbar` is a global object which you can use to control the appframe-toolbar

▪ The various AppToolbar methods is described here

- Enter the following code into the source editor

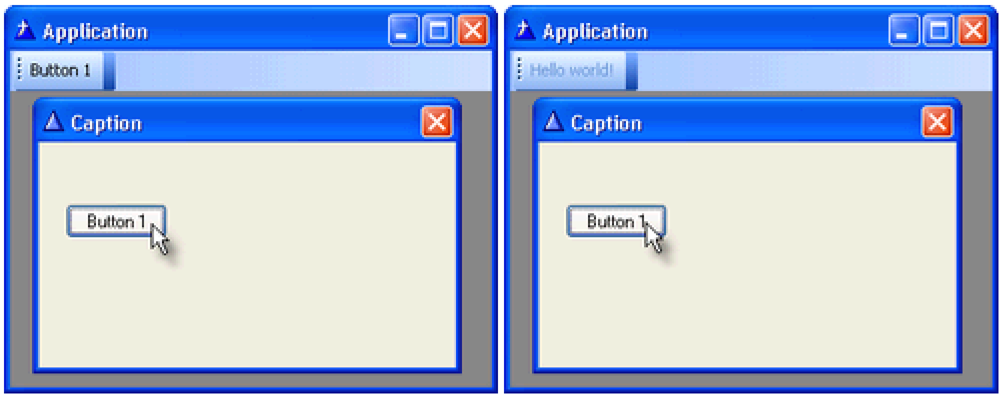
```
!Manipulate AppFrame toolbar button

!Enable/Disable
If AppToolbar.GetEnabled(ID1_Button1)                   !Check if button is
enabled
    AppToolbar.SetEnabled(ID1_Button1, False)           !Disable button
Else
    AppToolbar.SetEnabled(ID1_Button1, True)            !Enable button
End

!Change text
If AppToolbar.GetText(ID1_Button1) = 'Button 1'         !Get button text
    AppToolbar.SetText(ID1_Button1, 'Hello world!')     !Set new text
Else
    AppToolbar.SetText(ID1_Button1, 'Button 1')         !Set back to default
text
End
```

- Save and exit the source editor

Part 4 - Test the application

Compile and run the application.

- Click "Button 1" in the toolbar to open our test-window.
- Click "Button 1" in the test-window and observe what happens to the toolbar button.



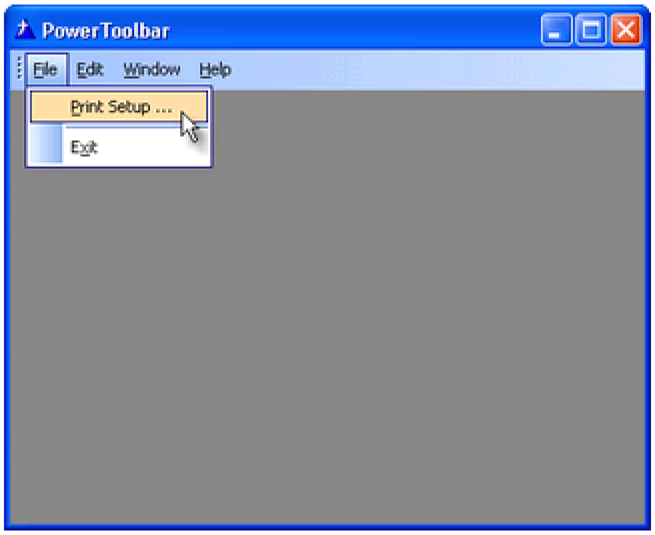
2.5 Working with menus

In this tutorial you learn how to set up the Clarion menu and override with PowerToolbar. Please note that the menu themes do NOT work in Clarion 7 and the PowerToolbar template automatically turns them off. See the Limitations topic for more information. As of Build 2.0.163 menus can again be themed in Clarion 7 and Clarion 8.

When working with the Clarion menu you must obey the following limitations:

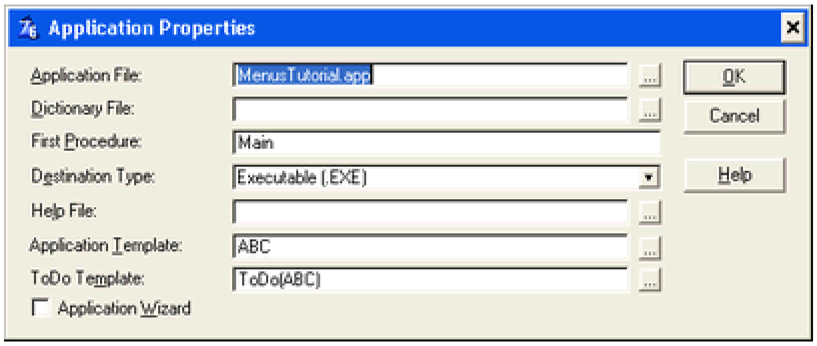
- If you choose to override the main Clarion menu you can NOT have (merging) menus in any child windows.
- All items in menu must have a Use-variable set
- All font colors must be COLOR:NONE
- Width must be "default" on all items
- Height must be "default" on all items
- Top level menu-items is not supported
- Runtime added top level-menus is not supported

The finished application for this tutorial can be found at:
Clarion6\3rdParty\Examples\PowerToolbar\MenusTutorial.app



Part 1 - Populate templates

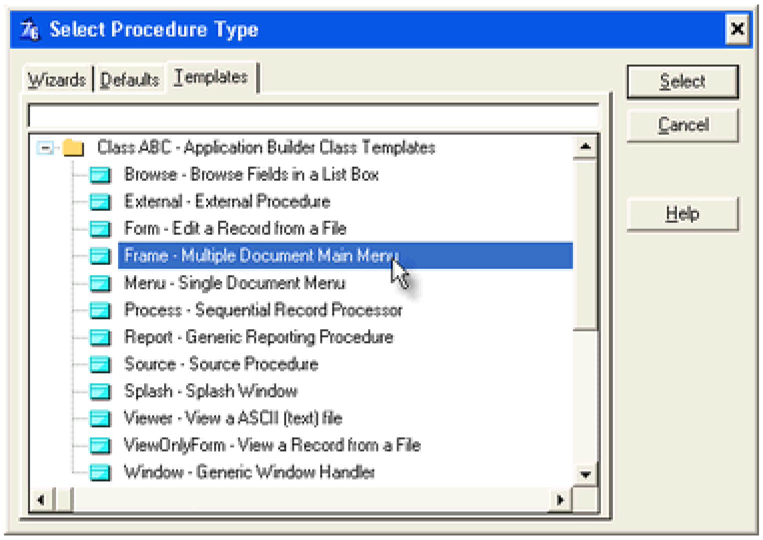
Create a new application



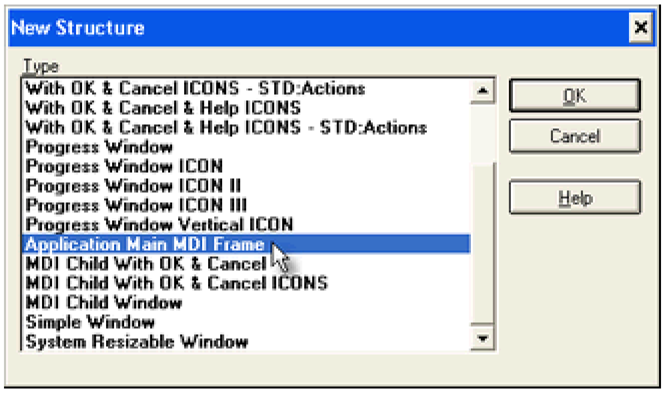
Application File: MenusTutorial.app
Dictionary File:
First Procedure: Main
Destination Type: Executable (.EXE)
Help File:
Application Template: ABC
ToDo Template: ToDo(ABC)
Application Wizard [ ]

Add the global PowerToolbar template
See Adding the templates

Add AppFrame
- Double-click the Main(Todo) procedure to bring up the "Select Procedure Type" dialog.
- Select "Frame - Multiple Document Main Menu"


Select Procedure Type
Wizards
Defaults
Templates
Select
Cancel
Help
Class ABC - Application Builder Class Templates
Browse - Browse Fields in a List Box
External - External Procedure
Form - Edit a Record from a File
Frame - Multiple Document Main Menu
Menu - Single Document Menu
Process - Sequential Record Processor
Report - Generic Reporting Procedure
Source - Source Procedure
Splash - Splash Window
Viewer - View a ASCII (text) file
ViewOnlyForm - View a Record from a File
Window - Generic Window Handler

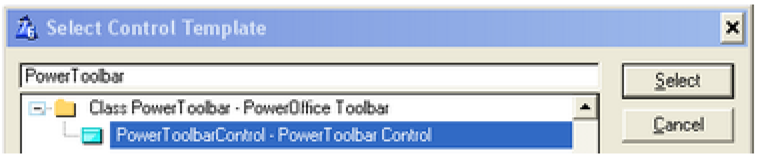
Create the window
- Click the Window button
- Select "Application Main MDI Frame" and click OK. This will bring up the window formatter.


New Structure
Type
With OK & Cancel ICONS - STD:Actions
With OK & Cancel & Help ICONS
With OK & Cancel & Help ICONS - STD:Actions
Progress Window
Progress Window ICON
Progress Window ICON II
Progress Window ICON III
Progress Window Vertical ICON
Application Main MDI Frame
MDI Child With OK & Cancel
MDI Child With OK & Cancel ICONS
MDI Child Window
Simple Window
System Resizable Window
OK
Cancel
Help

Create a toolbar
- Select the menu-item "Toolbar -> New Toolbar"

Populate PowerToolbar control
- Click the "Control Template" toolbutton

- Choose the template "POToolbar - Toolbar Control"


Select Control Template
PowerToolbar
Class PowerToolbar - PowerOffice Toolbar
PowerToolbarControl - PowerToolbar Control
Select
Cancel

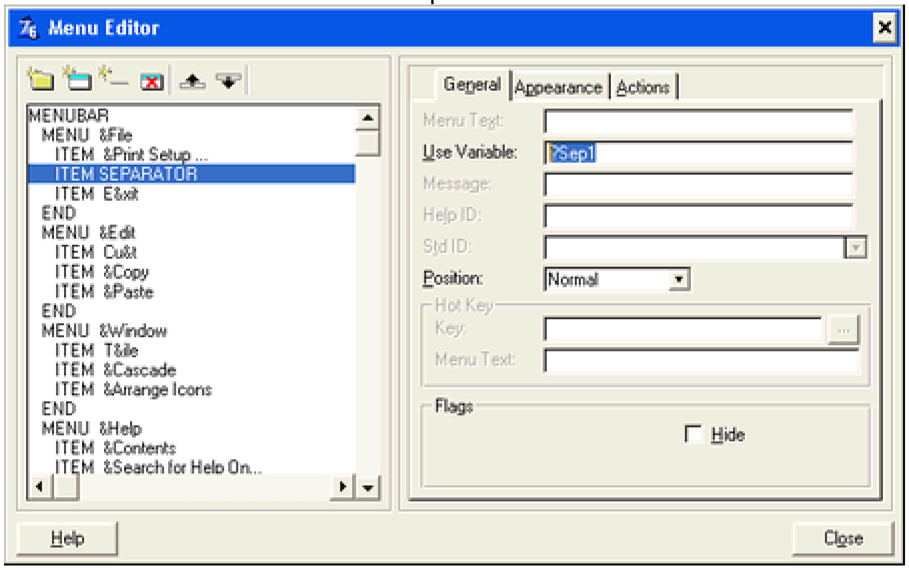
- Click somewhere inside the Clarion toolbar or Window to place the PowerToolbar-control.

Part 2 - Set up the menu

Double-click the menu band to open the menu editor

Place use variable on all items

- You must make sure that all items in the menu has a use-variable.
- In the default menu there is one separator without use-variable. Set it to ?Sep1

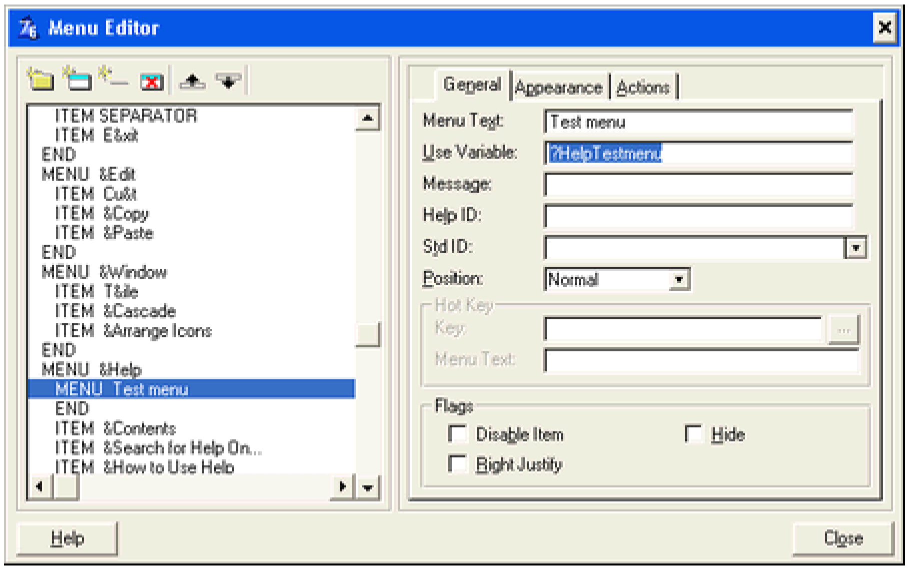


```
MENUBAR
  MENU  &File
    ITEM  &Print Setup ...
    ITEM SEPARATOR
    ITEM  E&xit
  END
  MENU  &Edit
    ITEM  Cu&t
    ITEM  &Copy
    ITEM  &Paste
  END
  MENU  &Window
    ITEM  T&ile
    ITEM  &Cascade
    ITEM  &Arrange Icons
  END
  MENU  &Help
    ITEM  &Contents
    ITEM  &Search for Help On...
```

Menu Text:
Use Variable: ?Sep1
Message:
Help ID:
Std ID:
Position: Normal
Key:
Menu Text:

Add submenu

- Add a submenu under the "Help"-menu by clicking
- Make sure the new item get's a use-variable. By default it doesn't.

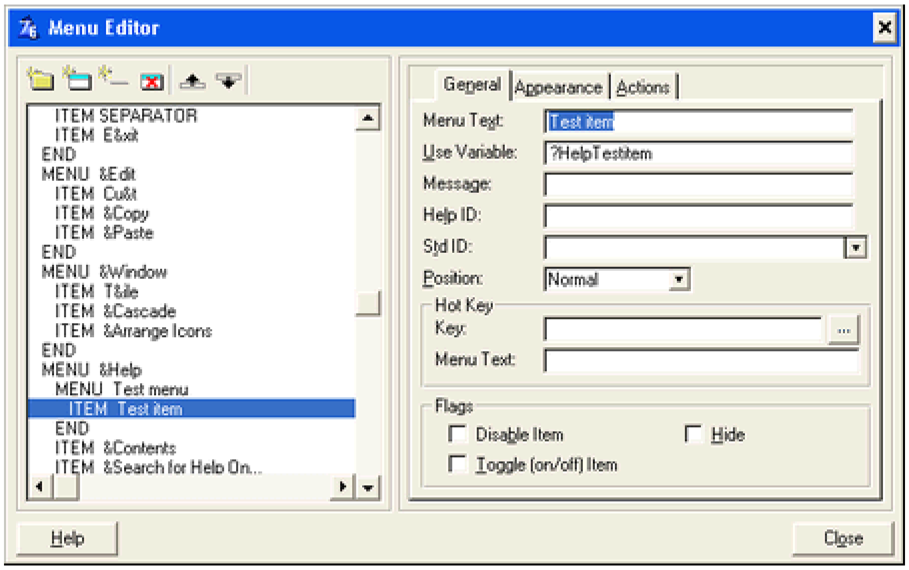


```
    ITEM SEPARATOR
    ITEM  E&xit
  END
  MENU  &Edit
    ITEM  Cu&t
    ITEM  &Copy
    ITEM  &Paste
  END
  MENU  &Window
    ITEM  T&ile
    ITEM  &Cascade
    ITEM  &Arrange Icons
  END
  MENU  &Help
    MENU  Test menu
    END
    ITEM  &Contents
    ITEM  &Search for Help On...
    ITEM  &How to Use Help
```

Menu Text: Test menu
Use Variable: ?HelpTestmenu
Message:
Help ID:
Std ID:
Position: Normal
Hot Key
Key:
Menu Text:
Flags
☐ Disable Item ☐ Hide
☐ Right Justify

Add sub-menu item

- Add a submenu-item under the newly created submenu by clicking

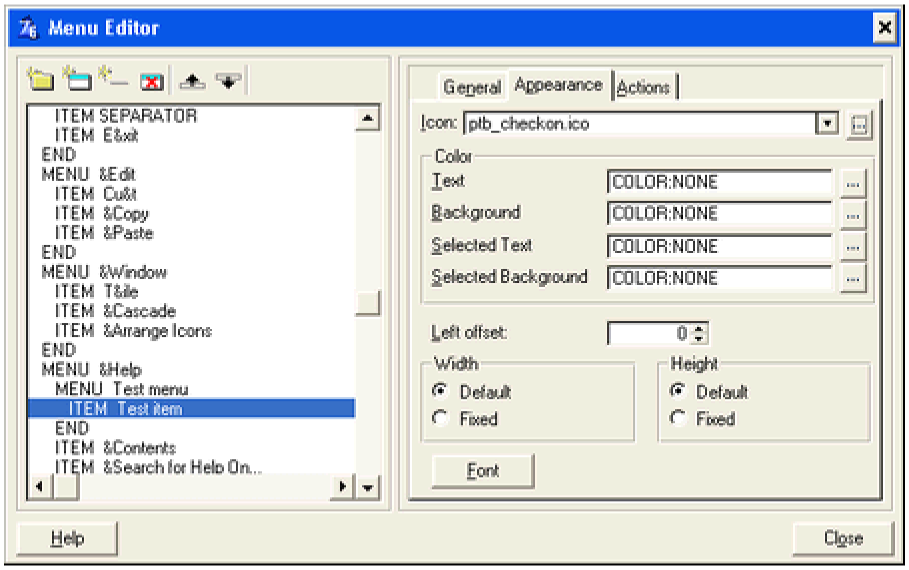


```
    ITEM SEPARATOR
    ITEM  E&xit
  END
  MENU  &Edit
    ITEM  Cu&t
    ITEM  &Copy
    ITEM  &Paste
  END
  MENU  &Window
    ITEM  T&ile
    ITEM  &Cascade
    ITEM  &Arrange Icons
  END
  MENU  &Help
    MENU  Test menu
      ITEM  Test item
    END
    ITEM  &Contents
    ITEM  &Search for Help On...
```

Menu Text: Test item
Use Variable: ?HelpTestitem
Message:
Help ID:
Std ID:
Position: Normal
Hot Key
Key:
Menu Text:
Flags
☐ Disable Item ☐ Hide
☐ Toggle (on/off) Item

Set icon

- Select the "Appearance"-tab
- Choose your favourite icon (16x16) by clicking the "..." button next to the Icon-prompt.
- Click the "Close" button to accept the menu changes and get back to the window formatter.



```
ITEM SEPARATOR
ITEM E&xit
END
MENU &Edit
ITEM Cu&t
ITEM &Copy
ITEM &Paste
END
MENU &Window
ITEM T&ile
ITEM &Cascade
ITEM &Arrange Icons
END
MENU &Help
MENU Test menu
ITEM Test item
END
ITEM &Contents
ITEM &Search for Help On...
```

Icon: ptb_checkon.ico

Color
- Text: COLOR:NONE
- Background: COLOR:NONE
- Selected Text: COLOR:NONE
- Selected Background: COLOR:NONE

Left offset: 0

Width: [✓ Default] Fixed

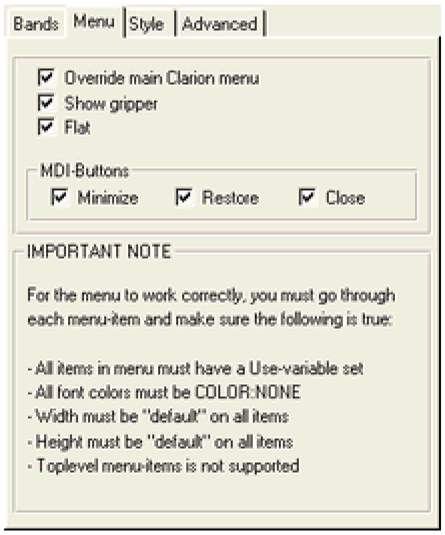
Height: [✓ Default] Fixed

Part 3 - Toolbar settings

Right-click the PowerToolbar control, and select "Actions"

Select the "Menu" tab

- Click the "Override main Clarion menu"-checkbox



- ☑ Override main Clarion menu
- ☑ Show gripper
- ☑ Flat

MDI-Buttons: ☑ Minimize ☑ Restore ☑ Close

IMPORTANT NOTE

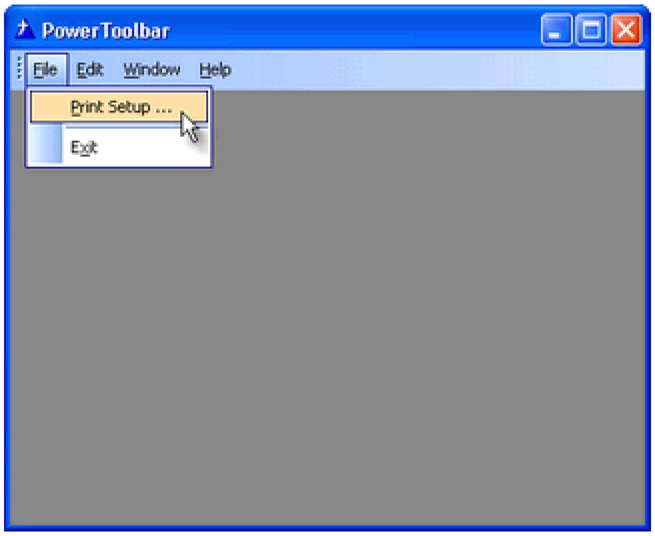
For the menu to work correctly, you must go through each menu-item and make sure the following is true:

- All items in menu must have a Use-variable set
- All font colors must be COLOR:NONE
- Width must be "default" on all items
- Height must be "default" on all items
- Toplevel menu-items is not supported

Part 4 - Try the application

Compile run your application

- Make sure the icon for the item we added shows. If it's missing or placed on the wrong item, you probably got a menu-item without use-variable.


PowerToolbar
File
Edit
Window
Help
Print Setup ...
Exit

Part III

Chapter 3 - Templates

3 Templates

3.1 About the templates

This chapter contains a brief description of the various templates

Have a look at Adding the templates for more info on how to add the templates.
For a detailed description of the various control settings, read Toolbar control template settings

Global Template

The global template is responsible for including the needed files in your application. This template must be present in all applications using the control-template. It should also be added to the data-dll application of multi-dll applications.

Note: The control-template will not be visible in the control template-list until the global template has been added to the application.

Toolbar Control Template

The control template generates the actual PowerToolbar-control. This template is populated into a window (or AppFrame toolbar).
You can have as many toolbars in a window as you like.

MDI Client Template

The MDI Client Template is developed to make it easy to control the AppFrame PowerToolbar from any MDI child window.

With this template you can easily select a control from the AppFrame toolbar, and a button in a window that should be mimiced by the toolbar control. If a mimiced button is disabled or enabled, so will the toolbar control. This is very handy when you want your user to be able to control the browse in the active window, from the toolbar.

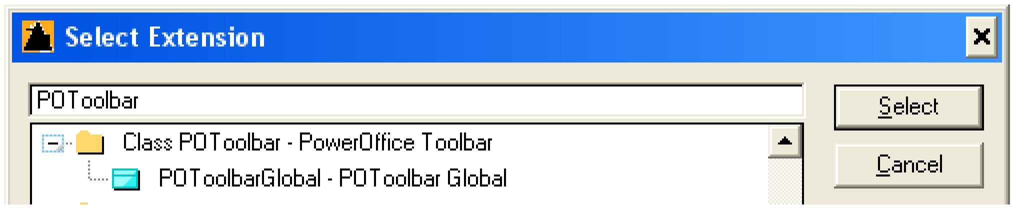
3.2 Adding the templates

This chapter shows you how to add the POToolbar global and control template.

Global template

- Press the "Global"-button

- Click "Extensions"
- Click "Insert"
- Choose "POToolbarGlobal - POToolbar Global"



POToolbar
Class POToolbar - PowerOffice Toolbar
POToolbarGlobal - POToolbar Global

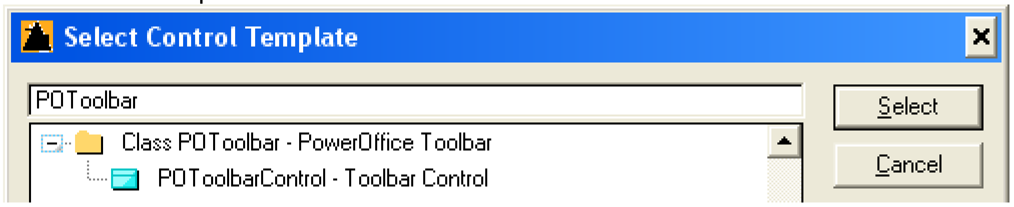
- Click "Ok" to close global settings.

Toolbar control template

The toolbar template can be populated into an existing AppFrame Clarion toolbar, or directly into a Window.

- Click the "Control Template" toolbutton

- Choose the template "POToolbar - Toolbar Control"

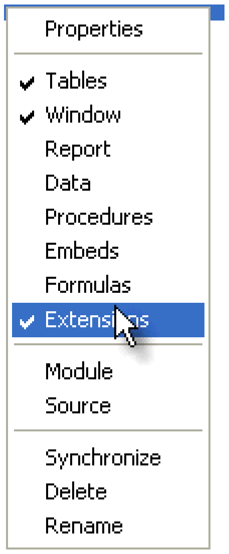


POToolbar
Class POToolbar - PowerOffice Toolbar
POToolbarControl - Toolbar Control

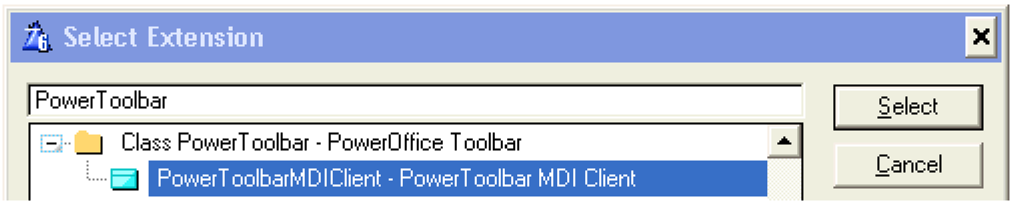
- Click somewhere inside the Clarion toolbar or Window to position the PowerToolbar-control.

MDI Client template

- Right-click the window-procedure you'd like to add the template to, and select "Extensions"



▸ Choose the "PowerToolbarMDIClient - PowerToolbar MDI Client" template



PowerToolbar

Class PowerToolbar - PowerOffice Toolbar

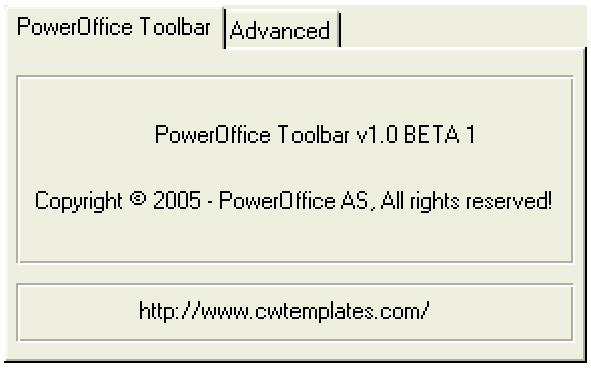
PowerToolbarMDIClient - PowerToolbar MDI Client

▸ Click "Ok" to close the extensions dialog.

3.3 Global Template

3.3.1 Global Settings — Global Template

The global template is responsible for including the needed files in your application. This template must be present in all applications using the control-template. It should also be added to the data-dll application of multi-dll applications.

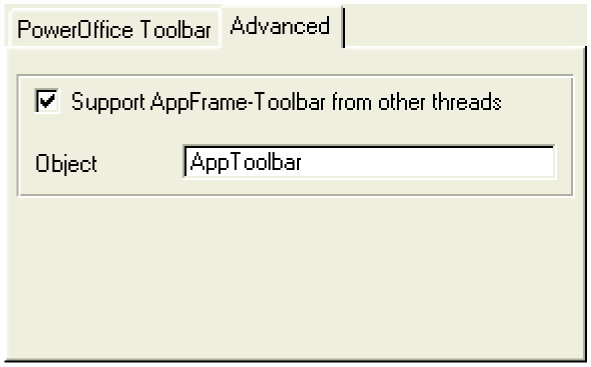


PowerOffice Toolbar v1.0 BETA 1

Copyright © 2005 - PowerOffice AS, All rights reserved!

http://www.cwtemplates.com/

Advanced



☑ Support AppFrame-Toolbar from other threads

Object: AppToolbar

Support AppFrame-Toolbar from other threads
If this option is checked, a global object is generated. With this object you can control the AppFrame PowerToolbar from other threads (eg. MDI child windows).
This is required when you're planning to use the PowerToolbar MDI Client Template.

Object name
Name of the object generated. Usually there's no need to change this name. If you do, you must make sure you have the same name set in the Appframe toolbar, and MDI Client templates.

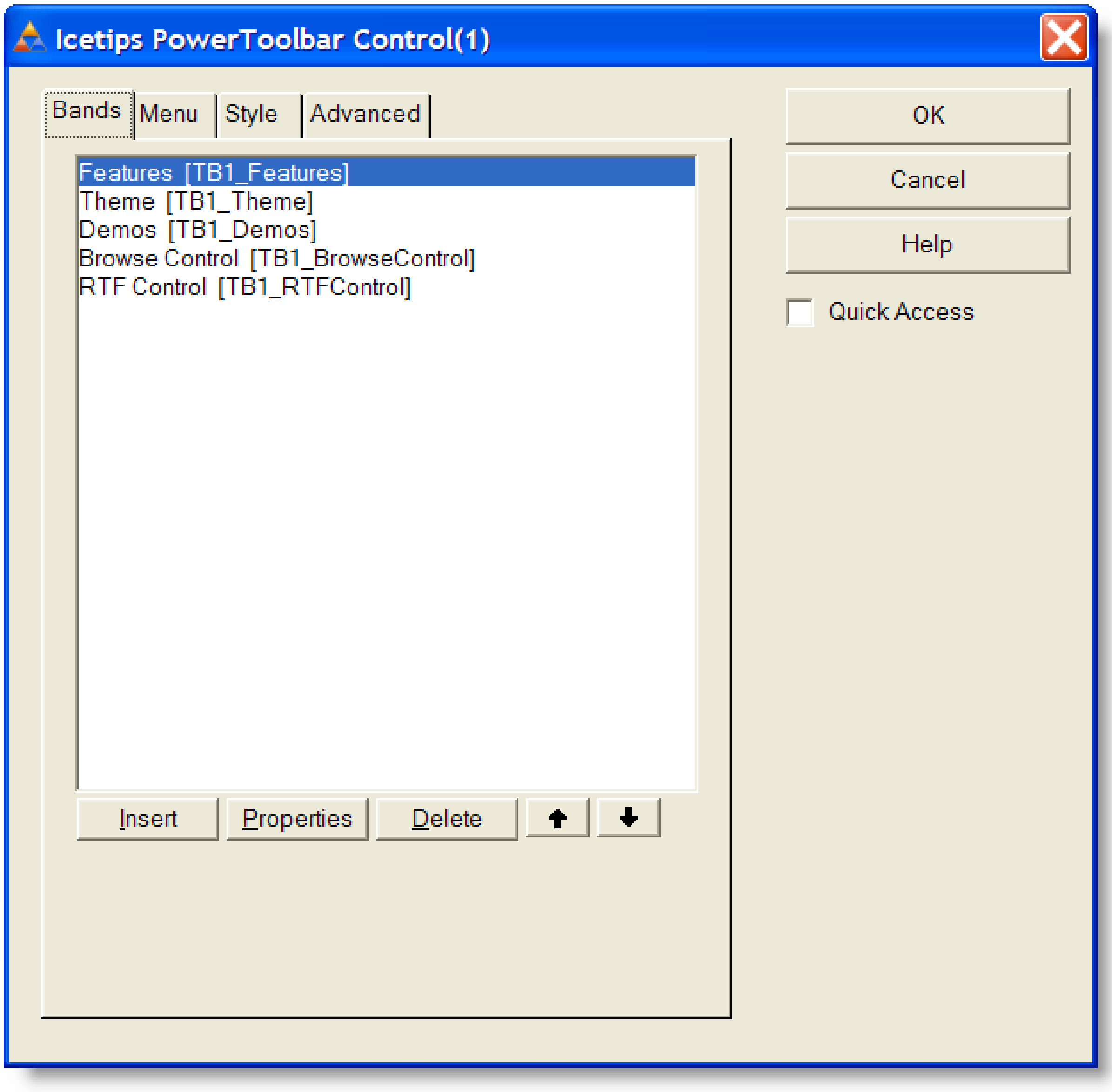
3.4 Toolbar Template

3.4.1 Main settings — Toolbar Template

In the control template settings you can control every aspect the toolbar.
This section describes the main PowerToolbar template settings.

Bands

This is a list of all bands in the toolbar.



Features [TB1_Features]
Theme [TB1_Theme]
Demos [TB1_Demos]
Browse Control [TB1_BrowseControl]
RTF Control [TB1_RTFControl]

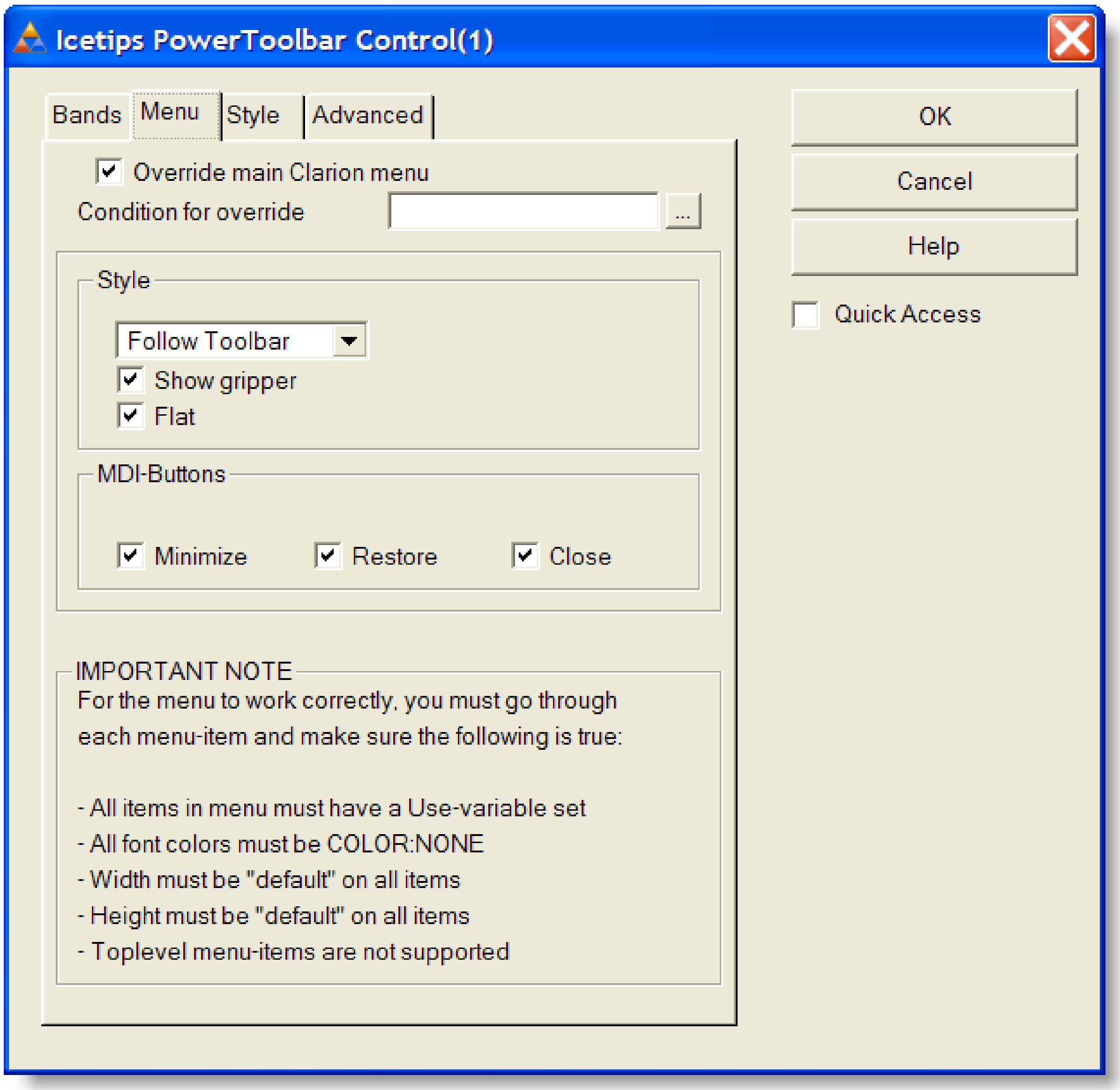
Insert
Inserts a new band, and opens Band properties

Properties
Opens band-properties for the selected band

Delete
Delete selected band

Menu

This is the settings for the AppFrame menu.
If "Override main Clarion menu" is activated, PowerToolbar will take over drawing of the menu and draw it with the selected PowerToolbar style.



[✓] Override main Clarion menu
Condition for override

Style
Follow Toolbar
[✓] Show gripper
[✓] Flat

MDI-Buttons
[✓] Minimize [✓] Restore [✓] Close

[ ] Quick Access

IMPORTANT NOTE
For the menu to work correctly, you must go through each menu-item and make sure the following is true:

- All items in menu must have a Use-variable set
- All font colors must be COLOR:NONE
- Width must be "default" on all items
- Height must be "default" on all items
- Toplevel menu-items are not supported

Override main Clarion menu
If this option is checked, PowerToolbar will draw the menu.

Condition for override
This can be used to set a condition for the override. If the condition evaluates as True, the menu will

be themed.

Style
Sets the style for the menu. It can be "Follow Toolbar" which will theme the menu in the same way as the Toolbar. Additional options are "Office 2000", "Office XP", "Office 2003" and "Native XP"

Show gripper
Controls if the gripper should be shown to the left of the first menu

Flat
Draws the menu-band flat (with no borders)

Mdi-Buttons
By default all MDI-buttons are visible (when the AppFrame has a maximized child window). Uncheck the appropriate check box to hide MDI-buttons of your choice.

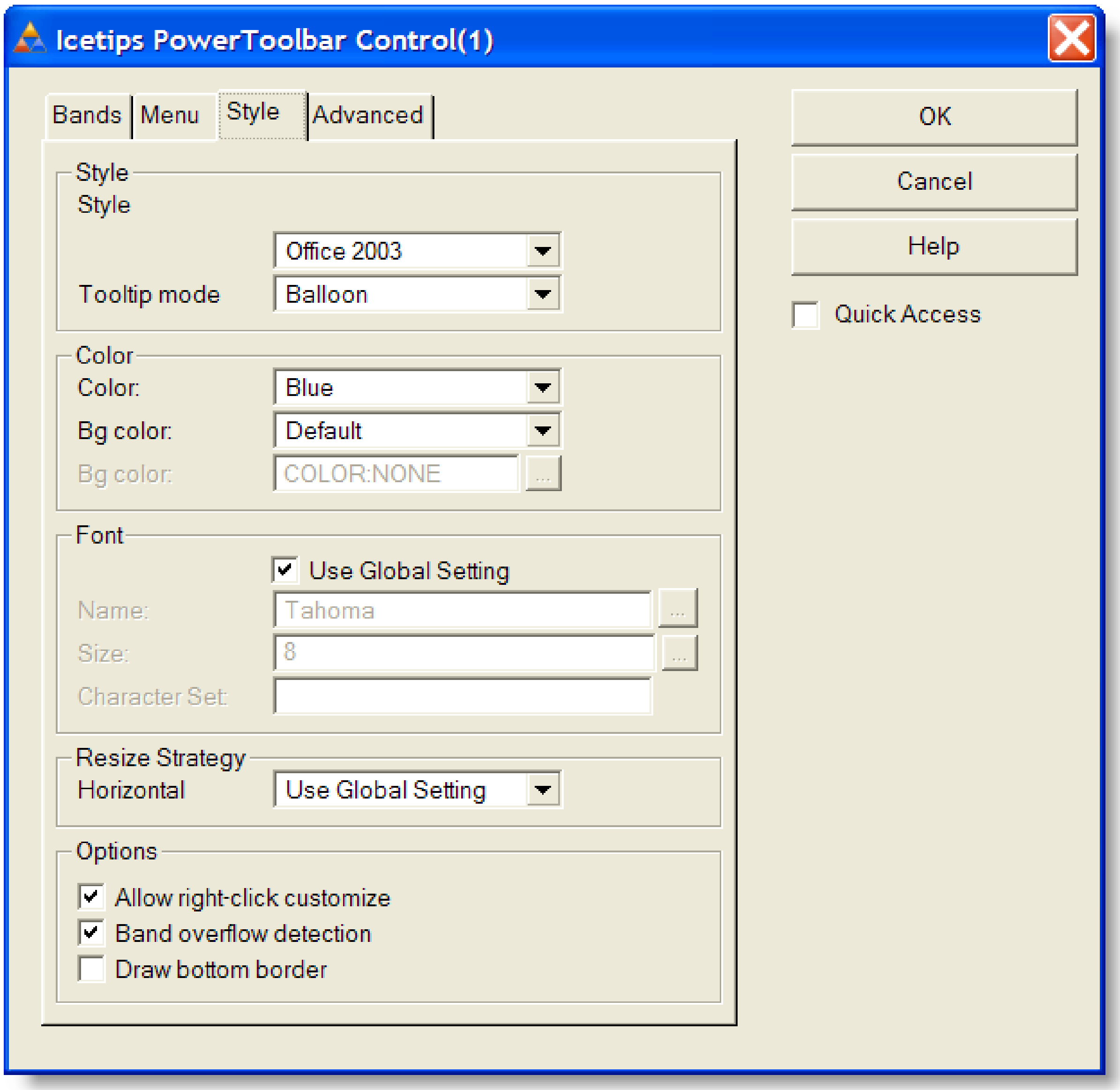
Important note
Several limitations applies when using PowerToolbar to draw the menubar:

- All items in menu must have a Use-variable set
- All font colors must be COLOR:NONE
- Width must be "default" on all items
- Height must be "default" on all items
- Top level menu-items is not supported

Read more about limitations here.

Style

Here you can choose toolbar style and tooltip mode.



☐ Quick Access

Style
Style: Office 2003
Tooltip mode: Balloon

Color
Color: Blue
Bg color: Default
Bg color: COLOR:NONE

Font
☑ Use Global Setting
Name: Tahoma
Size: 8
Character Set:

Resize Strategy
Horizontal: Use Global Setting

Options
☑ Allow right-click customize
☑ Band overflow detection
☐ Draw bottom border

Style
Toolbar draw style.

Available styles:
- Office 2000
- Office XP
- Office 2003
- Native XP

NativeXP is only available if PowerXP-Theme is included in the application, and the application is running under Windows XP or 2003.
On Windows 95/NT4 only Office 2000 is available. Read more about limitations here.

Tooltip mode
Controls how tooltip look

Available modes:

- Disabled
- Normal
- Balloon (only available on Windows XP and later)

Color
Toolbar color style (only affects Office 2003 style)

Available styles:
- Default (Gray)
- Blue
- Silver
- Widbey
- PowerXP-Theme (requires PowerXP-Theme to be included in the application)

Bg color
Determines the background color of the toolbar. It can be default or custom. If Custom is selected, the "Bg Color" entry is enabled and you can select a color.

Font
You can determine what font settings you want to use for the toolbar controls.

Global Settings
If this is checked, the toolbar inherits the settings from the global template. If it is not checked you can select what font, font size and character set to use.

Font name
The name of the font to use for the Toolbar

Font size
Font size to use for the Toolbar

Character set
The character set used for the Toolbar

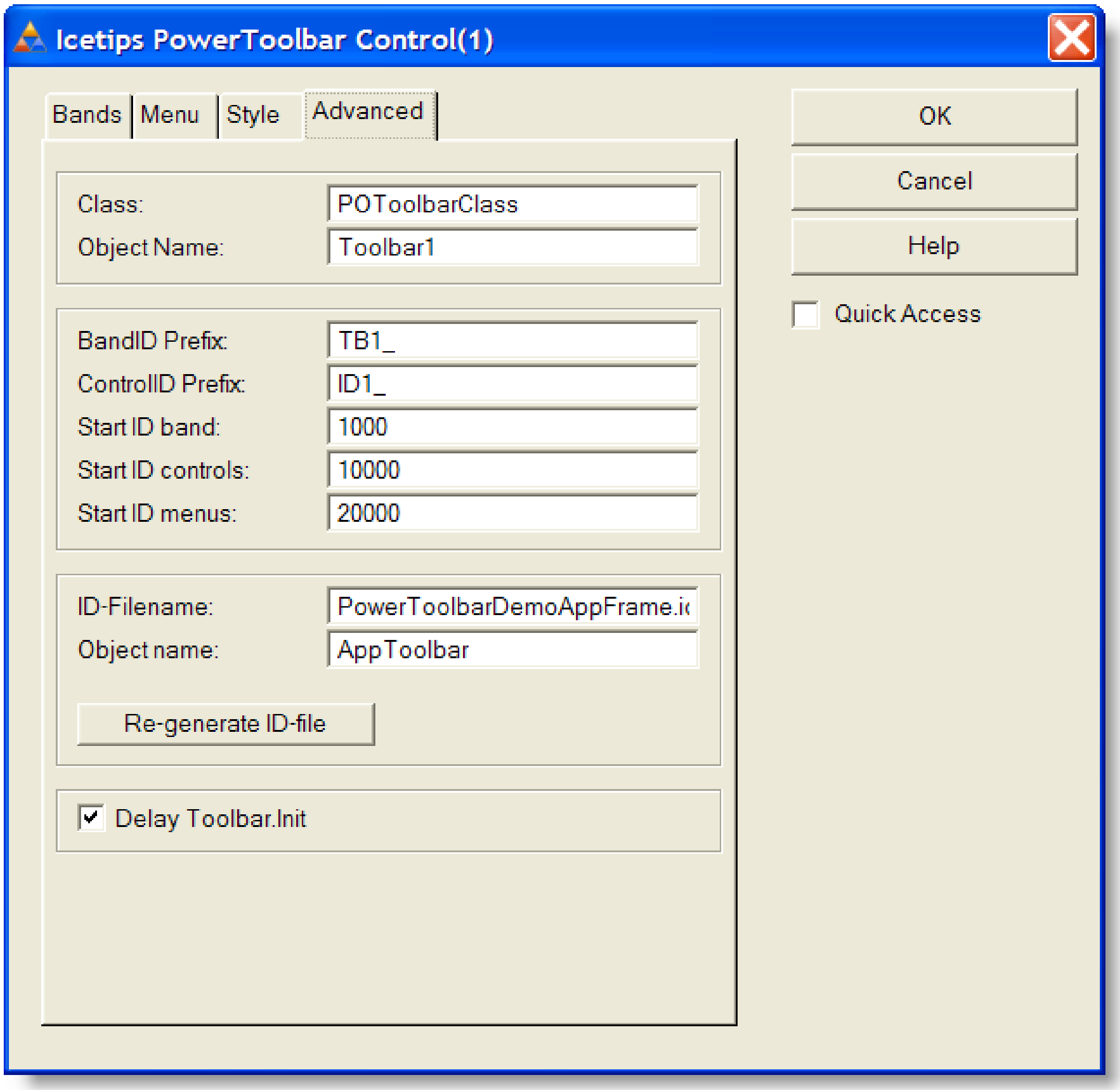
Allow right-click customize
If this option is on, the user can right-click the toolbar and hide bands.

Band overflow detection
This option will, if activated, make bands that would overflow it's row start on the next row instead.

Draw bottom border
If this option is on, a line with the appropriate border color is drawn at the bottom of the Toolbar.

Advanced

Advanced settings.
You usually won't need to do any changes here.



Class: POToolbarClass
Object Name: Toolbar1

BandID Prefix: TB1_
ControlID Prefix: ID1_
Start ID band: 1000
Start ID controls: 10000
Start ID menus: 20000

ID-Filename: PowerToolbarDemoAppFrame.i
Object name: AppToolbar

[✓] Delay Toolbar.Init

[ ] Quick Access

Class
Name of the class used for PowerToolbar

Object name
The object name used when generating the PowerToolbar object.
This is the object-label you will use when writing hand code to control the toolbar.

BandID prefix
Prefix of auto generated Band-ID's

ControlID prefix
Prefix of auto generated Control-ID's

Start ID band
Start value for Band-ID's

Start ID controls

Start value for Control-ID's

Start ID menus
Start value for top level Menu-ID's

ID-Filename
File name to write all ID's into. The file is used by the PowerToolbar MDI Child template.
This prompt is only available if PowerToolbar is populated into the AppFrame

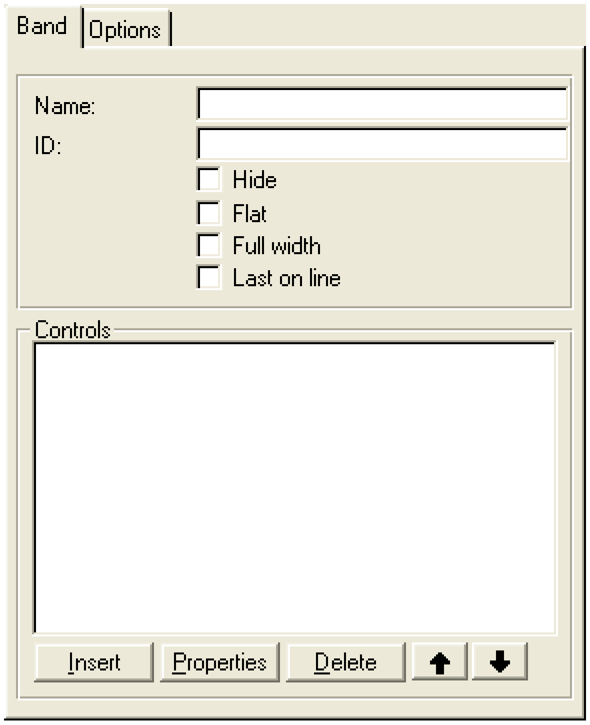
Object name
The name of the global AppFrame toolbar object. This must be the same as the global setting.
This prompt is only available if PowerToolbar is populated into the AppFrame.

Re-generate ID-file
If you click this button, the ID-file is generated. The file is generated with each compile, but if you want to refresh ID's without compiling, use this button.

Delay Toolbar.Init
Added in build 2.0.163. This delays the call to the Toolbar.Init method to prevent duplicate drawing of the menu, observed sometimes particularly in testing under Clarion 8. Note that this can cause GPFs if you have code that calls certain methods before the INIT is called. This can happen after installing 2.0.163 since it will default to TRUE for the delay and can cause problems in your code. In build released later in June 2012, we have added a message() that shows what method is being called in case it is called before the Init method is called.

3.4.2 Band settings — Toolbar Template

Here you control the settings for a band



Name
A descriptive name for the band.

This name is shown in the "Customize" list when right-clicking a band. If this name is empty, the band will not be shown in the customize-list.

ID
The equate label for this control

Hide
Band is initially hidden

Flat
Band is drawn without borders and vertical gradient.

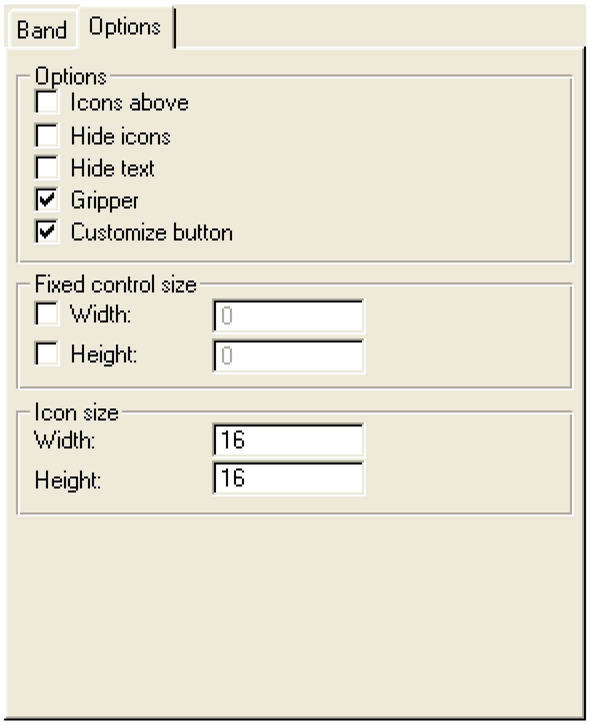
Full width
The band extends all the way to the right side of the toolbar.

Last on line
Next band will start on a new row

Controls
List of controls in this band.
Press *Insert* to add a new control

Band options

Here you control advanced settings for a band



Options
- [ ] Icons above
- [ ] Hide icons
- [ ] Hide text
- [x] Gripper
- [x] Customize button

Fixed control size
- [ ] Width: 0
- [ ] Height: 0

Icon size
Width: 16
Height: 16

Icons above
Button icons will be drawn above button text.

Hide icons
Do not show icons

Hide text
Do not show text on buttons

Gripper
Draws a gripper at the left hand side of the band

Customize button (Office 2003 only)
Draws a dark blue area at the end of the band

Include in right-click customize
This band is show in the customize list accessed with a right-click

Fixed control width
All controls in this band, will get this width

Fixed control height
The band and all it's controls will get this height

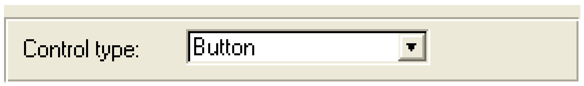
Icon width
Width of button icons

Icon height
Height of button icons

3.4.3 Control settings Toolbar Template

Each control type has it's own prompts. This chapter describes each control type.

Choose a control type from the droplist in the template.

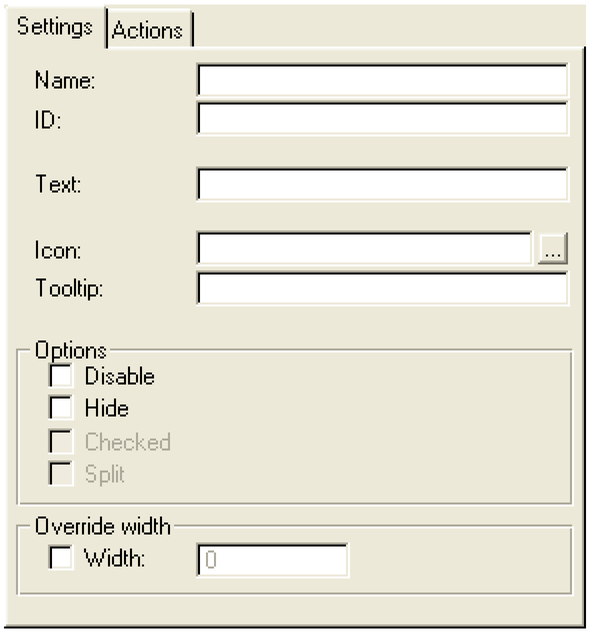


Available control types:
- Button
- CheckButton
- RadioButton
- MimicButton
- DropButton
- DropCombo
- Entry
- Text
- Separator
- Spacer

Most controls has an Actions-tab.

Button

The button control is a simple toolbar button. When the button is clicked, the selected action is executed.



Name
This is a descriptive name for the control.

ID
The ID equate label for the control.

Text
Control visible text. You can specify a variable by prefixing it with '!'. Eg. !MyVariable
If you put an & in front of a character, the character will be the hotkey for this control. Pressing Alt+<hotkey> will execute the control action.
If you want a & to appear in the text, use double ampersands (&&).

Icon
An icon for the control

DisabledIcon
Icon to use when control is disabled.
If you leave this prompt empty, the toolbar will create a temporary gray scaled version of the control icon, which will be used when the control is disabled.

Tooltip
Tooltip for the control

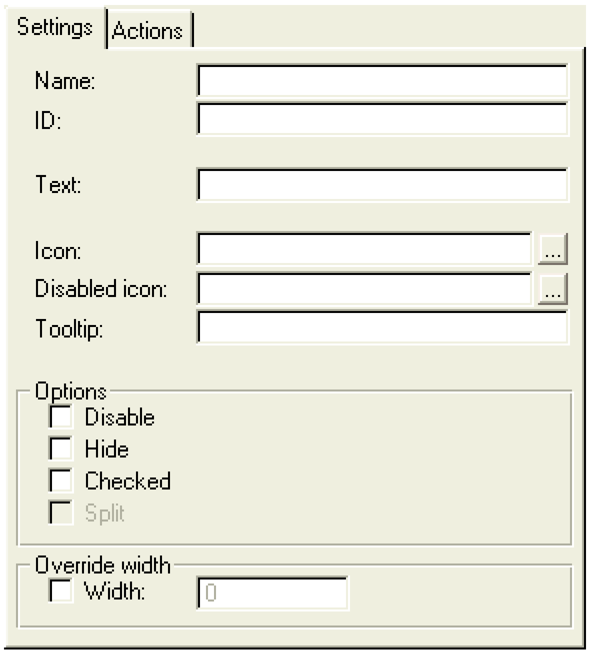
Disable
If this option is checked, the control is initially disabled

Hide
If this option is checked, the control is initially hidden

Width
You can use this option to set a width for the control (in pixels).

CheckButton

A checkbutton is a button that can be set to checked. If a button is checked, it will be drawn highlighted.



Name
This is a descriptive name for the control.

ID
The ID equate label for the control.

Text
Control visible text. You can specify a variable by prefixing it with '!'. Eg. !MyVariable
If you put an & in front of a character, the character will be the hotkey for this control. Pressing Alt+<hotkey> will execute the control action.
If you want a & to appear in the text, use double ampersands (&&).

Icon
An icon for the control

DisabledIcon
Icon to use when control is disabled.
If you leave this prompt empty, the toolbar will create a temporary gray scaled version of the control icon, which will be used when the control is disabled.

Tooltip
Tooltip for the control

Disable
If this option is checked, the control is initially disabled

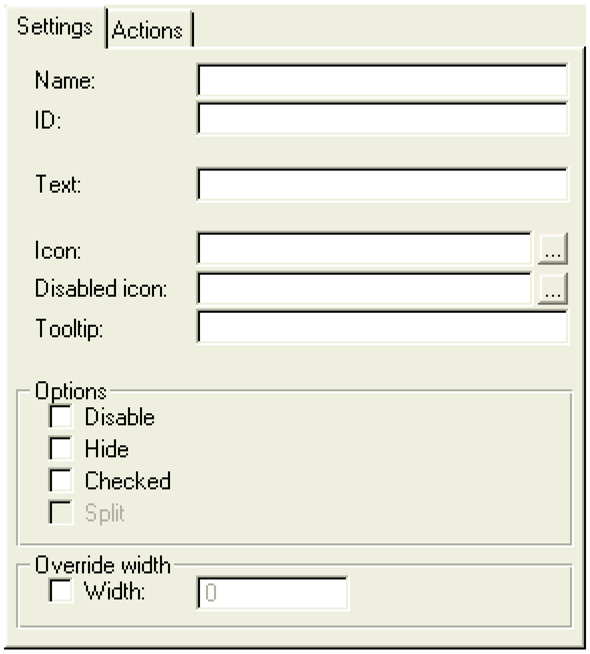
Hide
If this option is checked, the control is initially hidden

Checked
If this option is checked, the control will be initially checked.

Width
You can use this option to set a width for the control (in pixels).

RadioButton

A radiobutton is a button that can be set to checked. If a button is checked, it will be drawn highlighted. When a radio becomes checked, it will uncheck all radios next to it, on both sides.



Name
This is a descriptive name for the control.

ID
The ID equate label for the control.

Text
Control visible text. You can specify a variable by prefixing it with '!'. Eg. !MyVariable
If you put an & in front of a character, the character will be the hotkey for this control. Pressing Alt+<hotkey> will execute the control action.
If you want a & to appear in the text, use double ampersands (&&).

Icon
An icon for the control

DisabledIcon
Icon to use when control is disabled.
If you leave this prompt empty, the toolbar will create a temporary gray scaled version of the control icon, which will be used when the control is disabled.

Tooltip
Tooltip for the control

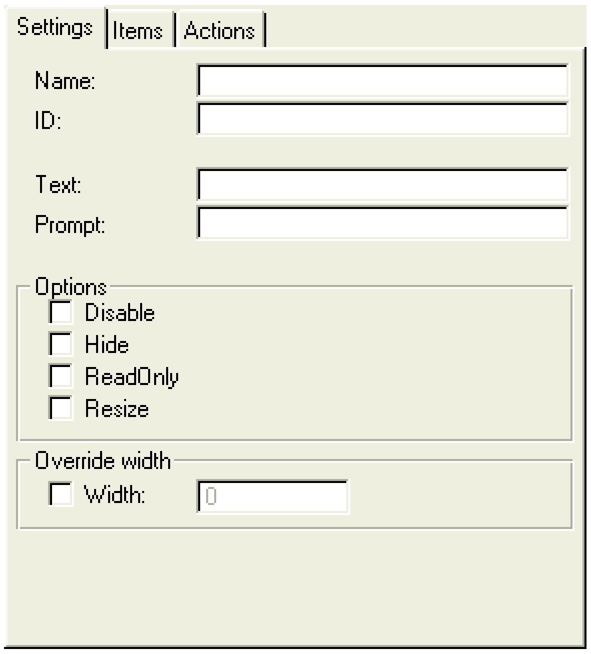
Disable
If this option is checked, the control is initially disabled

Hide
If this option is checked, the control is initially hidden

Checked
If this option is checked, the control will be initially checked.

Width
You can use this option to set a width for the control (in pixels).

DropCombo



Name
This is a descriptive name for the control.

ID
The ID equate label for the control.

Text
Control visible text. You can specify a variable by prefixing it with '!'. Eg. !MyVariable

Prompt
Optional text to show in front of the DropCombo.

Disable
If this option is checked, the control is initially disabled

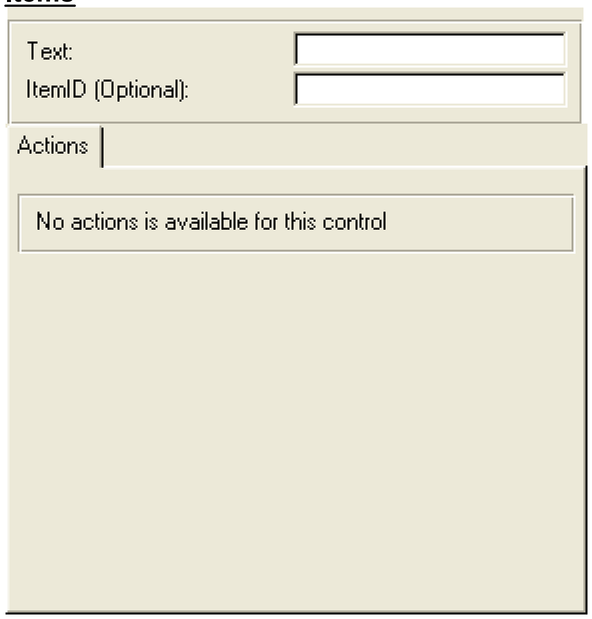
Hide
If this option is checked, the control is initially hidden

Readonly
Control is read-only, and can not be edited by the user.

Resize
Resize the control to maximum available width

Width
You can use this option to set a width for the control (in pixels).

Items



No actions is available for this control

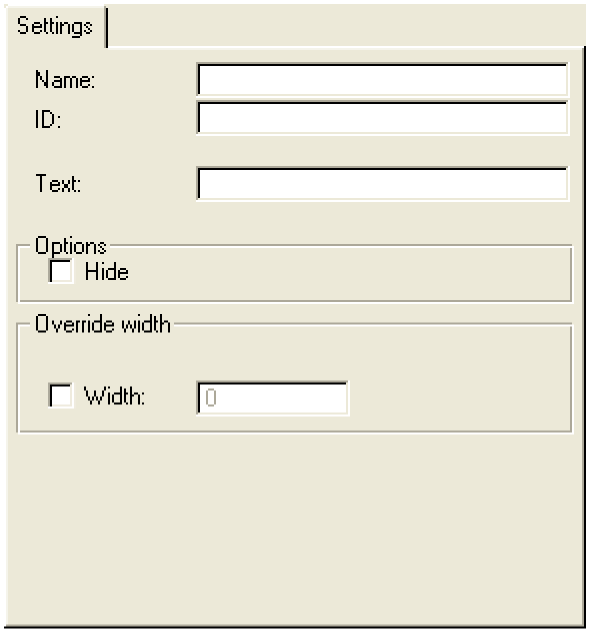
Text
Display text for this item

ItemID
A numeric ID to use for the item. If you leave this field empty, items will be enumerated, starting with ID=1.
You can specify a variable by prefixing it with '!' (eg. !MyVariable)

Actions
DropCombo controls has no item actions

Text

The Text control is a static control with no actions. It's only intended to add descriptive text to the toolbar.



Name
This is a descriptive name for the control.

ID
The ID equate label for the control.

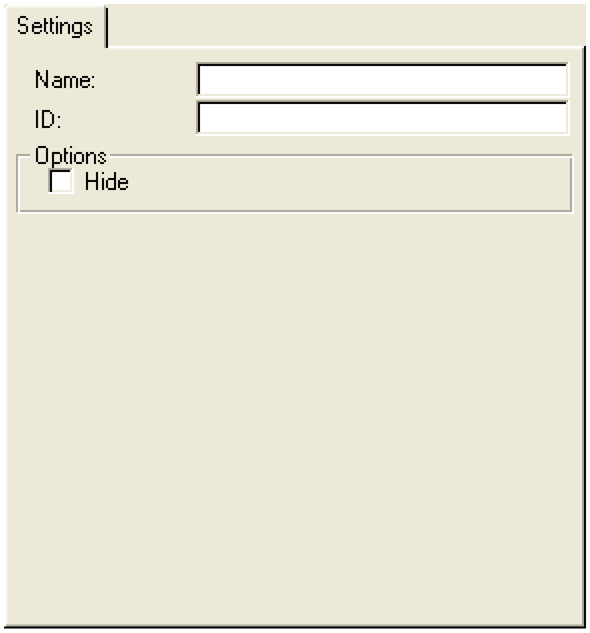
Text
Control visible text. You can specify a variable by prefixing it with '!'. Eg. !MyVariable

Hide
If this option is checked, the control is initially hidden

Width
You can use this option to set a width for the control (in pixels).

Separator

The separator control draws a vertical bar.



Name:

ID:

Options

☐ Hide

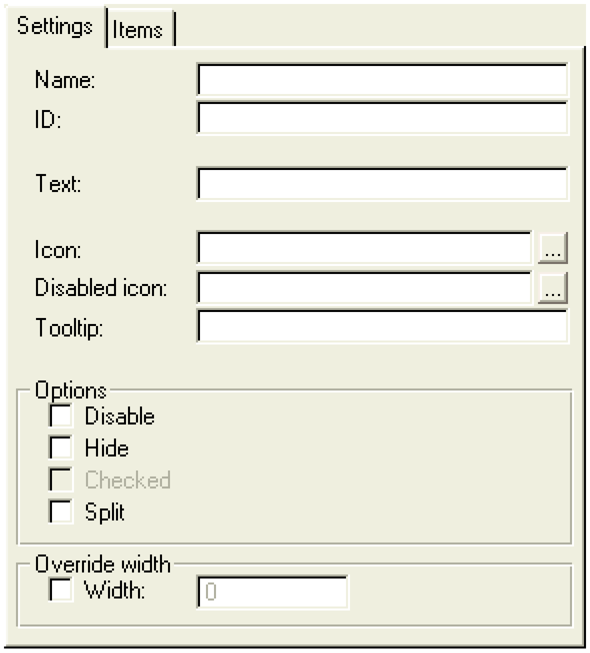
Name
This is a descriptive name for the control.

ID
The ID equate label for the control.

Hide
If this option is checked, the control is initially hidden

DropButton

A DropButton is a button that opens a drop-list with multiple items to choose from. A drop-button can also be split into a button-area, and a drop-area.



Name:

ID:

Text:

Icon: ...

Disabled icon: ...

Tooltip:

Options

☐ Disable

☐ Hide

☐ Checked

☐ Split

Override width

☐ Width: 0

Name
This is a descriptive name for the control.

ID
The ID equate label for the control.

Text
Control visible text. You can specify a variable by prefixing it with '!'. Eg. !MyVariable
If you put an & in front of a character, the character will be the hotkey for this control. Pressing Alt+<hotkey> will execute the control action.
If you want a & to appear in the text, use double ampersands (&&).

Icon
An icon for the drop button.

DisabledIcon
Icon to use when control is disabled.
If you leave this prompt empty, the toolbar will create a temporary gray scaled version of the control icon, which will be used when the control is disabled.

Tooltip
Tooltip for the control

Disable
If this option is checked, the control is initially disabled

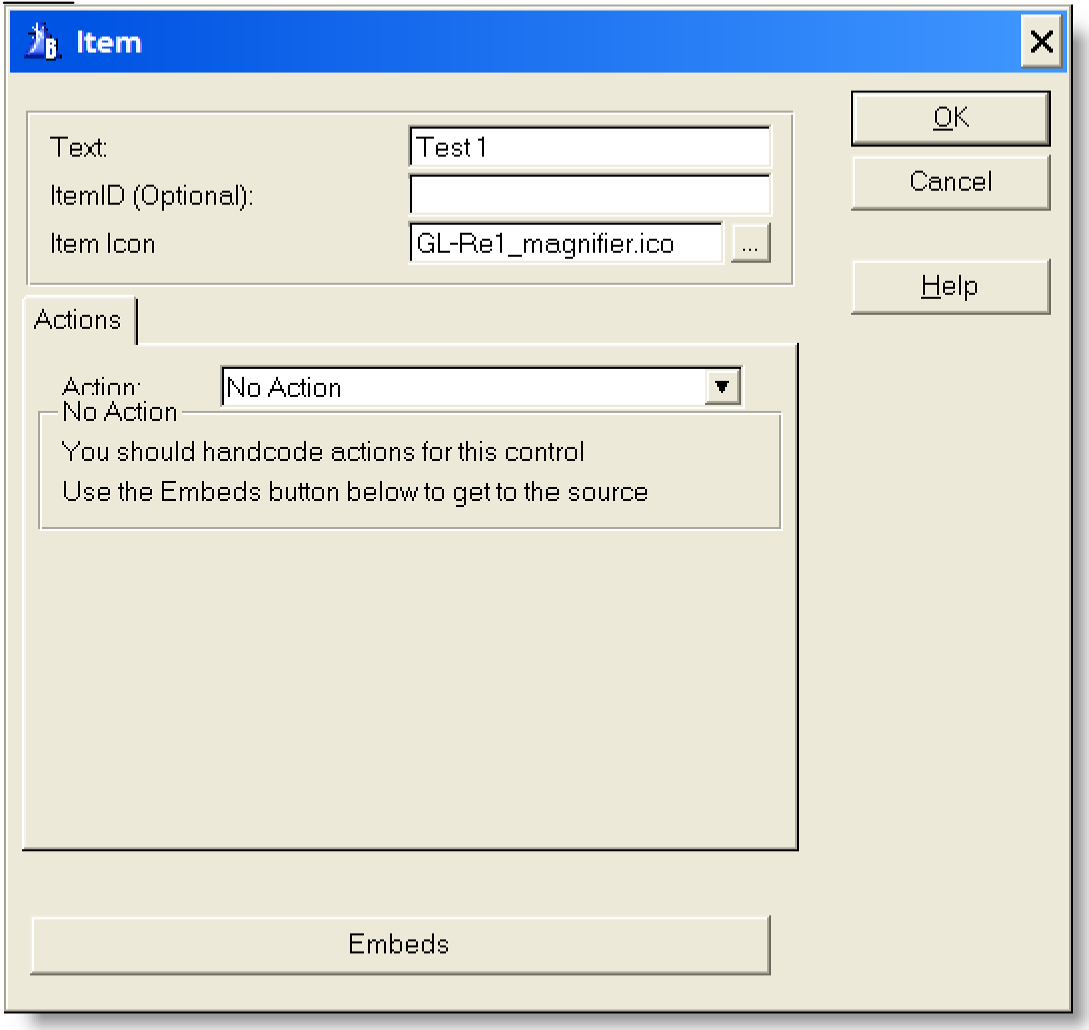
Hide
If this option is checked, the control is initially hidden

Split
Indicates that this is a split-button. Split buttons can have actions for both the button and any drop-items.

Width

You can use this option to set a width for the control (in pixels).

Items



Text: Test 1
ItemID (Optional):
Item Icon: GL-Re1_magnifier.ico

Action: No Action

No Action
You should handcode actions for this control
Use the Embeds button below to get to the source

Text
Display text for this item. To add a separator just use a single dash/hyphen: "-"

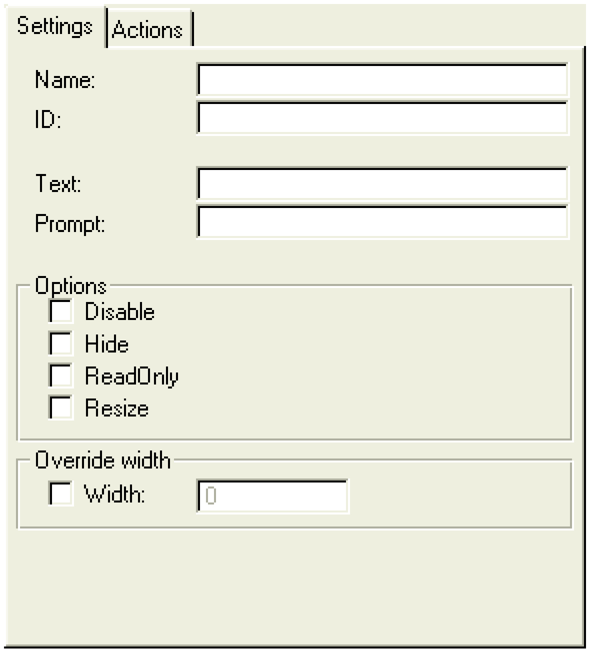
ItemID
A numeric ID to use for the item. If you leave this field empty, items will be enumerated, starting with ID=1.
You can specify a variable by prefixing it with '!' (eg. !MyVariable)

Icon
You can specify an icon for the menu items in the drop menu.

Actions
The settings for DropCombo item actions is the same as for control actions.

Entry

The entry control provides the user a possibility to write text in the toolbar.



Name
This is a descriptive name for the control.

ID
The ID equate label for the control.

Text
Control visible text. You can specify a variable by prefixing it with '!'. Eg. !MyVariable

Prompt
Optional text to show in front of the DropCombo.

Disable
If this option is checked, the control is initially disabled

Hide
If this option is checked, the control is initially hidden

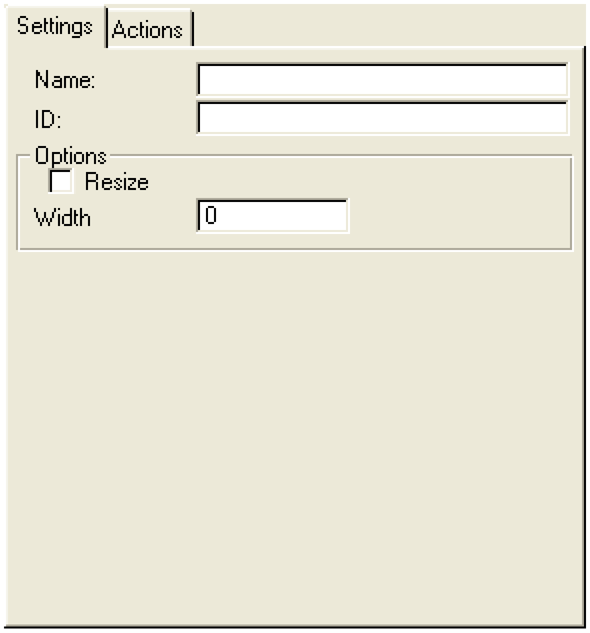
Readonly
Control is read-only, and can not be edited by the user.

Resize
Resize the control to maximum available width

Width
You can use this option to set a width for the control (in pixels).

Spacer

Spacers are invisible controls used to add space between other controls.



Name:

ID:

Options

☐ Resize

Width 0

Name
This is a descriptive name for the control.

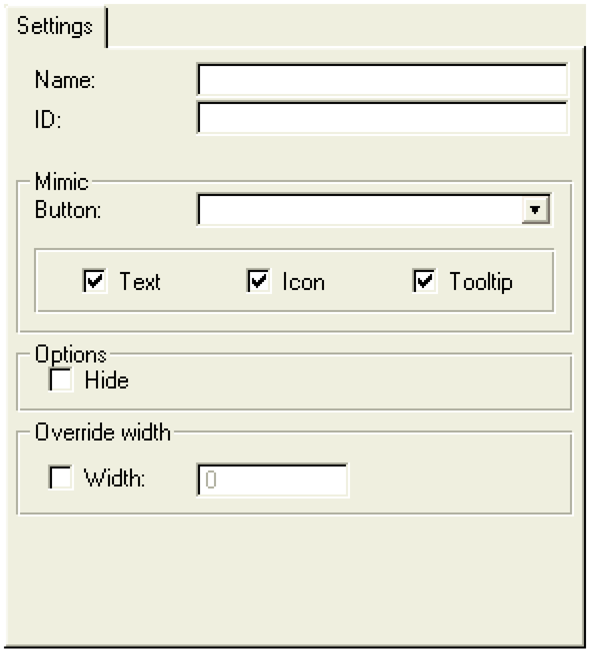
ID
The ID equate label for the control.

Resize
Resize the control to maximum available width

Width
You can use this option to set a width for the control (in pixels).

MimicButton

A mimicbutton is a button mimicing a Clarion button. The mimicbutton can get it's text, icon and tooltip from the mimiced button. A mimicbutton also follows the enabled/disabled state of the mimiced button. When the mimicbutton is clicked, an EVENT:Accepted is posted to the mimiced button.



Name
This is a descriptive name for the control.

ID
The ID equate label for the control.

Button
The label of the button you want to mimic.

Text
If this option is checked, the MimicButton will get it's text from the button

Icon
If this option is checked, the MimicButton will get it's icon from the button

Tooltip
If this option is checked, the MimicButton will get it's tooltip from the button

Hide
If this option is checked, the control is initially hidden

Width
You can use this option to set a width for the control (in pixels).

3.4.4 Actions Toolbar Template

Most controls and drop items can have an action associated with them.
Bellow is a description of each action type.

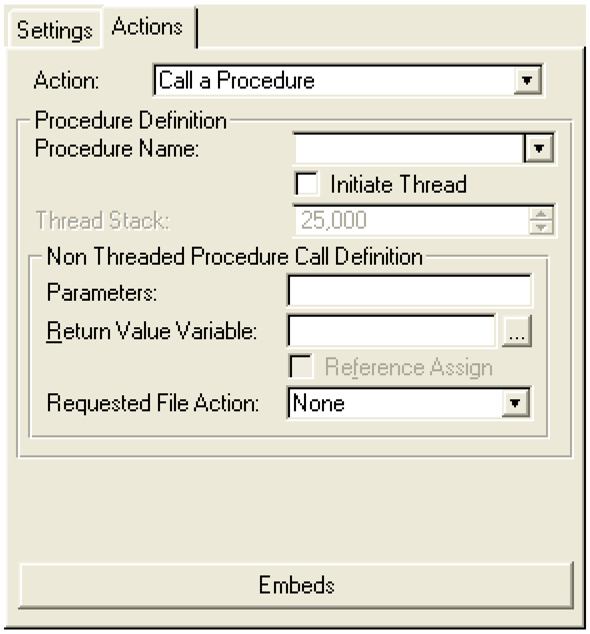
No action

This action has no prompts, and generates no code.

You should use the "Embed" button to get to the hand code-embeds.

Call a procedure

This action type mimics Clarion's "Call a procedure" action, and should be well know to Clarion programmers.



Action: Call a Procedure

Procedure Definition

Procedure Name:

Initiate Thread

Thread Stack: 25,000

Non Threaded Procedure Call Definition

Parameters:

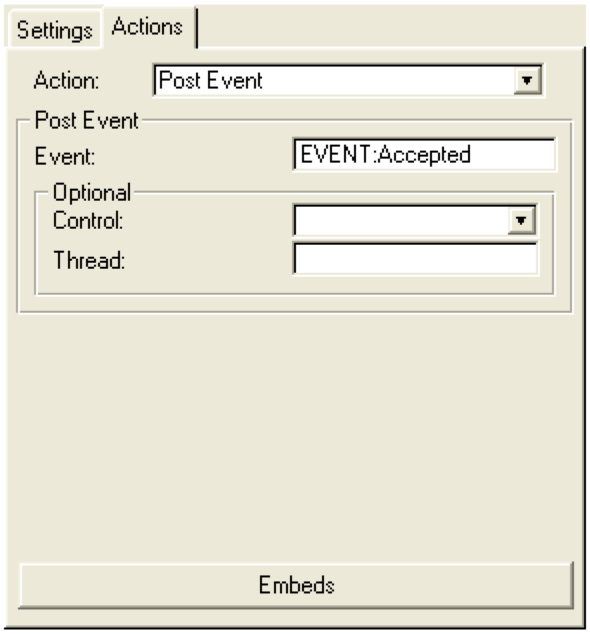
Return Value Variable:

Reference Assign

Requested File Action: None

Post Event

The Post Event action can be used to post an event to a control or thread.



Action: Post Event

Post Event

Event: EVENT:Accepted

Optional

Control:

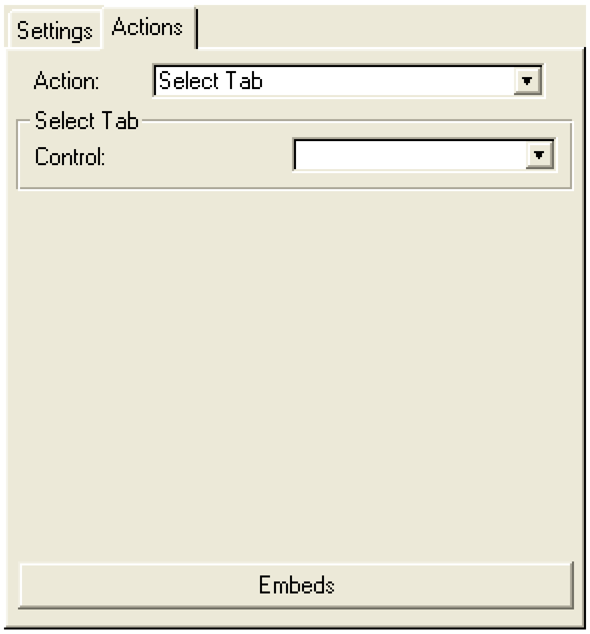
Thread:

Event
The event to be posted

Control
Choose control to post even to

Thread
Receiving thread

Select Tab
Use this action type to easily change tabs from a toolbar.

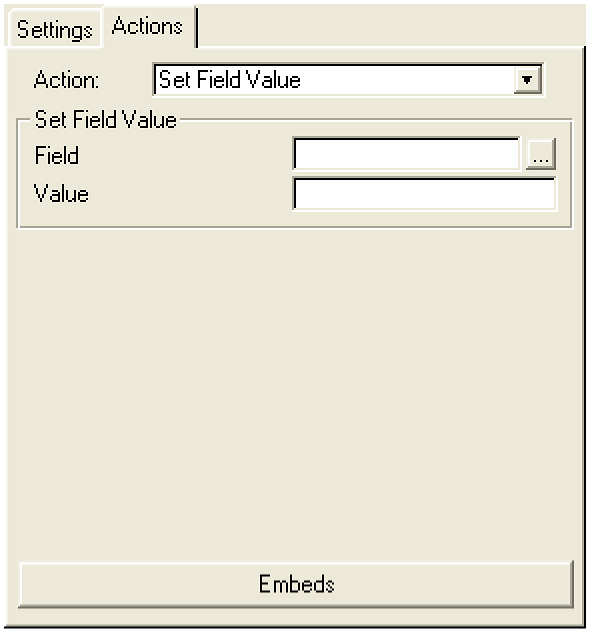


Action: Select Tab

Select Tab
Control:

Control
Choose tab control to change to.

Set field value
The set field value action assign a value to a field/variable.



Action: Set Field Value

Set Field Value

Field

Value

Field
Field/Variable to receive the value.

Value
Value to write to field. If this is a string, it must be enclosed into ''. (Eg. 'My string')
If this value is a variable or numeric, no prefixing is needed.

3.5 MDI Client Template

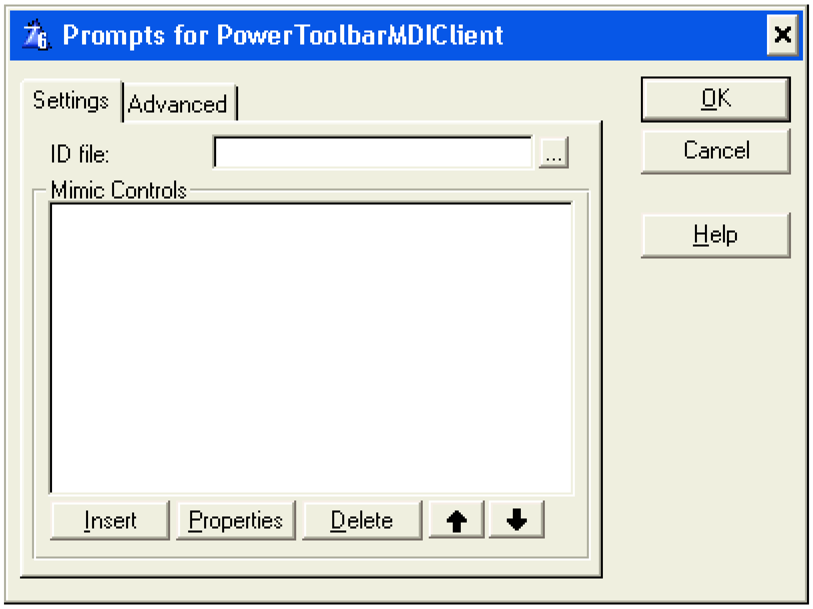
3.5.1 MDI Client Settings

MDI Client Template

The MDI Client Template is developed to make it easy to control the AppFrame PowerToolbar from any MDI child window.

With this template you can easily select a control from the AppFrame toolbar, and a button in a window that should be mimiced by the toolbar control. If a mimiced button is disabled or enabled, so will the toolbar control. This is very handy when you want your user to be able to control the browse in the active window, from the toolbar.

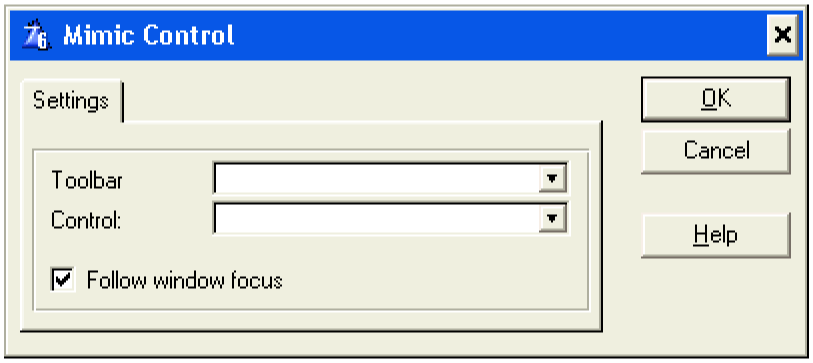
Settings



ID file
This is the ID file generated by the PowerToolbar template (populated into the AppFrame toolbar). This file contains all Control ID's used in the toolbar.

Mimic controls
Add/Change/Delete control mimics

Mimic Control



Toolbar
Control:
☑ Follow window focus

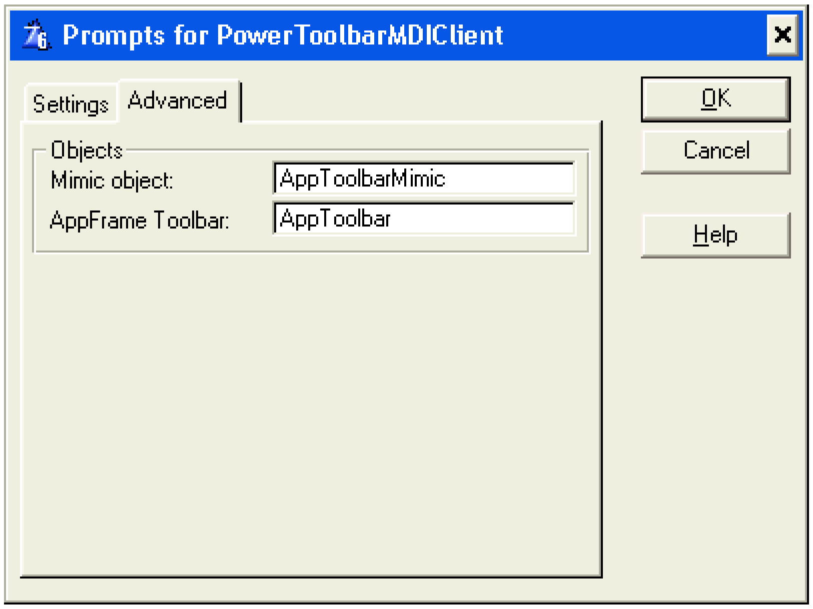
Toolbar Control ID
This is the Control ID for the AppFrame PowerToolbar control. These ID's is loaded from the "ID file" .

Control
A button in the current window that should be mimiced

Follow window focus
If this option is checked, the toolbar control will be disabled whenever this window loses focus. The button will be enabled when the window gains focus.
If this option is combined with mimicing BrowseUpdateButtons, the user will only be able to update the browse that's got focus.

Advanced

Under normal circumstances, you will not need to change these settings.



Objects
Mimic object: AppToolbarMimic
AppFrame Toolbar: AppToolbar

Mimic object
This is an object generated in the current thread, to keep track of the mimiced button state.

AppFrame Toolbar
The name of the global AppToolbar object. This name must match the setting of the global template.

Part IV

Chapter 4 - Misc

4 Misc

4.1 License agreement

Icetips "PowerToolbar" End-User License Agreement

Important - read carefully!

By installing this software you have agreed to be bound by the following End-User Licence Agreement.

ICETIPS ALTA LLC ("ICETIPS") IS WILLING TO LICENSE THE SOFTWARE ONLY UPON THE CONDITION THAT YOU ACCEPT ALL OF THE TERMS CONTAINED IN THIS SOFTWARE LICENSE AGREEMENT. PLEASE READ THE TERMS CAREFULLY. BY CLICKING ON "YES, ACCEPT" OR BY INSTALLING THE SOFTWARE, YOU WILL INDICATE YOUR AGREEMENT WITH THEM. IF YOU ARE ENTERING INTO THIS AGREEMENT ON BEHALF OF A COMPANY OR OTHER LEGAL ENTITY, YOUR ACCEPTANCE REPRESENTS THAT YOU HAVE THE AUTHORITY TO BIND SUCH ENTITY TO THESE TERMS, IN WHICH CASE "YOU" OR "YOUR" SHALL REFER TO YOUR ENTITY. IF YOU DO NOT AGREE WITH THESE TERMS, OR IF YOU DO NOT HAVE THE AUTHORITY TO BIND YOUR ENTITY, THEN ICETIPS IS UNWILLING TO LICENSE THE SOFTWARE, AND YOU SHOULD SELECT THE "NO, DECLINE" BUTTON AND THE DOWNLOAD OR INSTALL WILL NOT CONTINUE.

SOFTWARE LICENSE AGREEMENT

1. Parties. The parties to this Agreement are you, the licensee ("You") and Icetips. If You are not acting on behalf of Yourself as an individual, then "You" means Your company or organization.

2. The Software. The Software licensed under this Agreement consists of computer programs only in compiled, object code form, data compilation(s), and documentation referred to as Icetips subscription product (the "Software").

3. Subscription Term For Registered User Version. The term of the license granted herein for the registered version of the Software shall be on a subscription basis with an initial term of one (1) year, and optional recurring renewal terms of one (1) year each, unless prior to renewal this license is terminated by written notice by You for convenience or terminated by either party for material breach. Renewal procedures are described in the accompanying documentation, and unless such procedures are strictly satisfied, including the payment of any required license fee, Your updates to the Software is not authorized, but use of Your existing Software is authorized. No updates or upgrades to the Software can be authorized unless the license fee is paid.

4. Registered Version License Grant for Single Copies (Non-Network Use). If You are a registered user of the Software, You are granted non-exclusive rights to install and use the Software by a single person who uses the Software only on one or more computers or workstations. You may copy the Software for archival purposes, provided that any copy must contain the original Software's proprietary notices in unaltered form.

5. Registered Version License Grant For Network Use. If You are a registered user of the Software, You are granted non-exclusive rights to install and use the Software and/or transmit the Software over an internal computer network, provided You acquire and dedicate a licensed copy of the Software for each user who may access the Software concurrently with any other user. You may copy the Software for archival purposes, provided that any copy must contain the original Software's proprietary notices in unaltered form.

6. Restrictions. You may not: (i) permit others to use the Software, except as expressly provided above for authorized network use; (ii) modify or translate the Software; (iii) reverse engineer, decompile, or disassemble the Software, except to the extent this restriction is expressly prohibited by applicable law; (iv) create derivative works based on the Software; (v) merge the Software with another product; (vi) copy the Software, except as expressly provided above; or (vii) remove or obscure any proprietary rights notices or labels on the Software.

7. Purchase of Additional Licenses. Registered users of the Software may purchase license rights for additional authorized use of the Software in accordance with Icetips's then-current volume pricing schedule. Such additional licenses shall be governed by the terms and conditions hereof. You agree that, absent Icetips's express written acceptance thereof, the terms and conditions contained in any purchase order or other document issued by You to Icetips for the purchase of additional licenses, shall not be binding on Icetips to the extent that such terms and conditions are additional to or inconsistent with those contained in this Agreement.

8. Transfers. You may make a one-time permanent transfer of all of your license rights to the Software to another party, provided that all of the following conditions are satisfied: (a) you notify us in writing of your intent to transfer your license rights and identify the party receiving the Software with complete contact information; (b) the transfer must include all of the Software, including all its component parts, original media, printed materials and this License Agreement; (c) you do not retain any copies of any version of the Software, full or partial, including copies stored on a computer or other storage device; and (d) the party receiving the Software reads and agrees to accept the terms and conditions of this License Agreement. Notwithstanding the foregoing, we reserve the right to require the transfer of possession of all physical copies of the Software to us for purposes of re-issue of replacement copies to the party receiving the Software.

9. Ownership. Icetips and its suppliers own the Software, all physical copies thereof, and all intellectual property rights embodied therein, including copyrights and valuable trade secrets embodied in the Software's design and coding methodology. The Software is protected by United States copyright laws and international treaty provisions. This Agreement provides You only a limited use license, and no ownership of any intellectual property. We reserve the right to require you to transfer possession of all physical copies of the Software to us for purposes of re-issue of replacement copies.

10. Warranty Disclaimer; Limitation of Liability. ICETIPS PROVIDES THE SOFTWARE "AS-IS" AND PROVIDED WITH ALL FAULTS. NEITHER ICETIPS NOR ANY OF ITS SUPPLIERS OR RESELLERS MAKES ANY WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. ICETIPS AND ITS SUPPLIERS SPECIFICALLY DISCLAIM THE IMPLIED WARRANTIES OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, SYSTEM INTEGRATION, AND DATA ACCURACY. THERE IS NO WARRANTY OR GUARANTEE THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED, ERROR-FREE, OR VIRUS-FREE, OR THAT THE SOFTWARE WILL MEET ANY PARTICULAR CRITERIA OF PERFORMANCE, QUALITY, ACCURACY, PURPOSE, OR NEED. YOU ASSUME THE ENTIRE RISK OF SELECTION, INSTALLATION, AND USE OF THE SOFTWARE. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS AGREEMENT. NO USE OF THE SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

11. Local Law. If implied warranties may not be disclaimed under applicable law, then ANY IMPLIED WARRANTIES ARE LIMITED IN DURATION TO THE PERIOD REQUIRED BY APPLICABLE LAW. Some jurisdictions do not allow limitations on how long an implied warranty may last, so the above limitations may not apply to You. This warranty gives you specific rights, and You may have other rights which vary from jurisdiction to jurisdiction.

12. Limitation of Liability. INDEPENDENT OF THE FORGOING PROVISIONS, IN NO EVENT AND UNDER NO LEGAL THEORY, INCLUDING WITHOUT LIMITATION, TORT, CONTRACT, OR STRICT PRODUCTS LIABILITY, SHALL ICETIPS OR ANY OF ITS SUPPLIERS BE LIABLE TO YOU OR ANY OTHER PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY KIND, INCLUDING WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER MALFUNCTION, OR ANY OTHER KIND OF COMMERCIAL DAMAGE, EVEN IF ICETIPS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY TO THE EXTENT PROHIBITED BY APPLICABLE LAW. IN NO EVENT SHALL ICETIPS'S LIABILITY FOR DAMAGES FOR ANY CAUSE WHATSOEVER, AND REGARDLESS OF THE FORM OF ACTION, EXCEED IN THE AGGREGATE THE AMOUNT OF THE PURCHASE PRICE PAID FOR THE SOFTWARE LICENSE.

13. Export Controls. You agree to comply with all export laws and restrictions and regulations of the United States or foreign agencies or authorities, and not to export or re-export the Software or any direct product thereof in violation of any such restrictions, laws or regulations, or without all necessary approvals. As applicable, each party shall obtain and bear all expenses relating to any necessary licenses and/or exemptions with respect to its own export of the Software from the U.S. Neither the Software nor the underlying information or technology may be electronically transmitted or otherwise exported or re-exported (i) into Cuba, Iran, Iraq, Libya, North Korea, Sudan, Syria or any other country subject to U.S. trade sanctions covering the Software, to individuals or entities controlled by such countries, or to nationals or residents of such countries other than nationals who are lawfully admitted permanent residents of countries not subject to such sanctions; or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals and Blocked Persons or the U.S. Commerce Department's Table of Denial Orders. By downloading or using the Software, Licensee agrees to the foregoing and represents and warrants that it complies with these conditions.

14. U.S. Government End-Users. The Software is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire the Software with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished rights are reserved under the copyright laws of the United States.

15. Licensee Outside The U.S. If You are located outside the U.S., then the following provisions shall apply: (i) Les parties aux presentes confirment leur volonte que cette convention de meme que tous les documents y compris tout avis qui siy rattache, soient rediges en langue anglaise (translation: "The parties confirm that this Agreement and all related documentation is and will be in the English language."); and (ii) You are responsible for complying with any local laws in your jurisdiction which might impact your right to import, export or use the Software, and You represent that You have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

16. Severability. If any provision of this Agreement is declared invalid or unenforceable, such provision shall be deemed modified to the extent necessary and possible to render it valid and enforceable. In any event, the unenforceability or invalidity of any provision shall not affect any other provision of this Agreement, and this Agreement shall continue in full force and effect, and be construed and enforced, as if such provision had not been included, or had been modified as above provided, as the case may be.

17. Arbitration. Except for actions to protect intellectual property rights and to enforce an arbitrator ’s decision hereunder, all disputes, controversies, or claims arising out of or relating to this Agreement or a breach thereof shall be submitted to and finally resolved by arbitration under the rules of the American Arbitration Association ("AAA") then in effect. There shall be one arbitrator, and such arbitrator shall be chosen by mutual agreement of the parties in accordance with AAA rules. The arbitration shall take place in Port Angeles, Washington, USA, and may be conducted by telephone or online. The arbitrator shall apply the laws of the State of Washington, USA to all issues in dispute. The controversy or claim shall be arbitrated on an individual basis, and shall not be consolidated in any arbitration with any claim or controversy of any other party. The findings of the arbitrator shall be final and binding on the parties, and may be entered in any court of competent jurisdiction for enforcement. Enforcements of any award or judgment shall be governed by the United Nations Convention on the Recognition and Enforcement of Foreign Arbitral Awards. Should either party file an action contrary to this provision, the other party may recover attorney's fees and costs up to $1000.00.

18. Jurisdiction And Venue. The courts of Clallam County in the State of Washington, USA and the nearest U.S. District Court shall be the exclusive jurisdiction and venue for all legal proceedings that are not arbitrated under this Agreement.

19. Force Majeure. Neither party shall be liable for damages for any delay or failure of delivery arising out of causes beyond their reasonable control and without their fault or negligence, including, but not limited to, Acts of God, acts of civil or military authority, fires, riots, wars, embargoes, Internet disruptions, hacker attacks, or communications failures. Notwithstanding anything to the contrary contained herein, if either party is unable to perform hereunder for a period of thirty (30) consecutive days, then the other party may terminate this Agreement immediately without liability by ten (10) days written notice to the other.

20. Miscellaneous. This Agreement constitutes the entire understanding of the parties with respect to the subject matter of this Agreement and merges all prior communications, representations, and agreements. This Agreement may be modified only by a written agreement signed by the parties. If any provision of this Agreement is held to be unenforceable for any reason, such provision shall be reformed only to the extent necessary to make it enforceable. This Agreement shall be construed under the laws of the State of Washington, USA, excluding rules regarding conflicts of law. The application the United Nations Convention of Contracts for the International Sale of Goods is expressly excluded. The parties agree that the Uniform Computer Transactions Act or any version thereof, adopted by any state, in any form ("UCITA"), shall not apply to this Agreement, and to the extent that UCITA may be applicable, the parties agree to opt out of the applicability of UCITA pursuant to the opt-out provision(s) contained therein.

Icetips Alta LLC
3430 East Highway 101, Ste. #28
Port Angeles WA 98362
EMail: support@icetips.com
http://www.icetips.com

4.2 Multi-DLL applications

In your data-dll you must add the global template "**PowerToolbarGlobal - Icetips PowerToolbar Global**".

Note about filenames
DLL's with a PowerToolbar-control must not be renamed after compilation. If the dll is not found by its compile-time name, in the current dir or path, PowerToolbar will be unable to load icons from the DLL. Instead it will try to load them from the .exe file (which in most cases will lead to "invisible" icons)...

If you MUST rename your DLL: The programmer is responsible for setting the correct filename, at runtime, with the method SetAppName.

4.3 Limitations

For some of the many features of PowerToolbar limitations apply. In this section you'll get a overview of these limitations.

Clarion versions supported
▸PowerToolbar supports Clarion 6.0 and later with reservation (see below) about the MDI client only being supported in Clarion 6.1 and later.

Mimic controls and menu items
▸It is possible to mimic menu items and create items on toolbars that directly correspond to menu items. However, mimicking menu items does not work with all the mimic attributes correctly. It does, however, work correctly for icons, tooltips, text, hide/unhide, enable/disable for controls on the toolbar or the window. Only icons and hide/unhide seem to work correctly on menu items. This limitation is in review and we hope to fix it in early 2010.

PowerToolbar MDI Client Extension Template
▸This template is not compatible with Clarion 6.0 (tested with build 0.915). The Clarion RTL does not always post EVENT:GainFocus when a window gains focus, so the MDI client buttons are not enabled. Note that this was fixed probably in Clarion 6.1. It does not happen in Clarion 6.3 build 9053 or 9058.

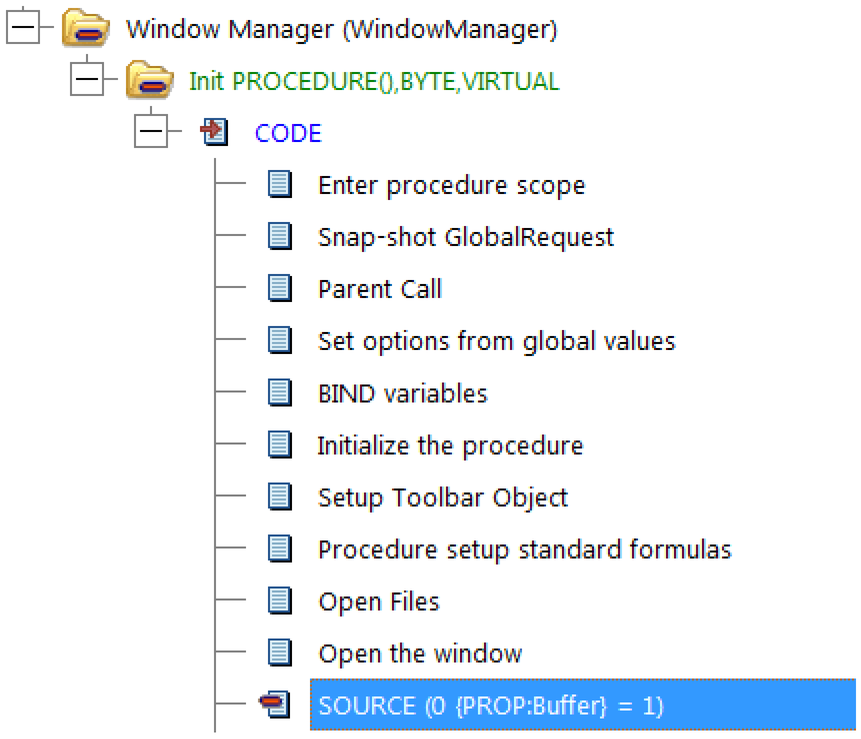
Clarion 7
▸Clarion 7 cannot support owner drawn menus. There for it is currently impossible for PowerToolbar to support menus under Clarion 7. The template disables the option and turns it off automatically so it will not affect the functionality of the application once converted to Clarion 7. However we want to point out that Clarion 7 has various theme options for menus that can at least partially replace the menu themes in the PowerToolbar.

Clarion 8
▸With May 2012 release of Powertoolbar it does support menu theming in Clarion 7 and Clarion 8. **While this has been in testing for a while we appreciate any feedback on any issues that you may run into!**

Clarion 10
▸Somewhere along the way in Clarion 10 (could be in previous versions as well), a flicker would appear in the app-frame client area if a centered wallpaper image was used. It would not occur if the image was larger than the client area. The solution is to add:

```
0 {PROP:Buffer} = 1
```

right after the window opens, see image of the embed tree below.


Window Manager (WindowManager)
Init PROCEDURE(),BYTE,VIRTUAL
CODE
Enter procedure scope
Snap-shot GlobalRequest
Parent Call
Set options from global values
BIND variables
Initialize the procedure
Setup Toolbar Object
Procedure setup standard formulas
Open Files
Open the window
SOURCE (0 {PROP:Buffer} = 1)

Clarion controls in Toolbars
- Clarion spin and entry controls on toolbars are themed (flat) even when Classic Windows theme is used in Windows. We are looking into a fix for this.
- Clarion controls added to toolbars cannot be on tabs on the window. This can cause bleeding through of other controls on the tabs. To work around this, simply place the controls on the window and hide/unhide them as needed when tabs are selected.
- See section below on toolbars on dockable windows:

Toolbars on Dockable windows with Clarion controls in toolbar
- If you are using PowerToolbar control on a dockable window and you are adding Clarion controls to it, these contro

```
 Toolbar4.Init()

 ! [Priority 8107]
```

```
Window{Prop:Docked}  = DOCK:Bottom
Window{Prop:Dock}    = DOCK:Bottom
```

AppFrame Toolbar
- Clarion toolbars and merging Clarion toolbars is NOT supported. If you got any merging toolbars in your application these will most likely corrupt the display of PowerToolbar.
- PowerToolbar will take up the entire Clarion toolbar. Other controls will be overdrawn. (This does not apply when the toolbar is populated into a Window)
- Clarion's Toolbar Browse Control template is not compatible with PowerToolbar (use the PowerToolbar MDI Client template instead).

Clarion Menus
- If you choose to override the main Clarion menu you can NOT have menus in any child windows, i.e. MDI windows.
- All items in menu must have a Use-variable set
- All font colors must be COLOR:NONE

- Width must be "default" on all items
- Height must be "default" on all items
- Top level menu-items is not supported
- Runtime added top level-menus are not supported
- The text from the Msg() attribute is not drawn in the status bar.
- HIDE attribute on MENU is **not** supported. Use Hide(?MenuLabel) or ?MenuLabel {PROP:Hide}=TRUE in code instead to hide menus. HIDE **is** supported on menu ITEMs. However, it can cause problems and we strongly suggest NOT using HIDE on menu items and hide them at runtime!

Native XP Style

- Native XP style requires PowerXP-Theme. If PowerXP-Theme isn't included in the application, the Toolbar-style will fall back to OFFICE2000.

XP-Theme colors

- To use colors of the currently selected XP-Theme in Office2003-style, you have to include PowerXP-Theme in your application. If not, colors will fall back to "blue theme".

Windows versions and Toolbar styles

- When running under Windows 95/NT4 only OFFICE2000 style is available (all other styles will fall back to OFFICE2000)
- OFFICE2000, OFFICEXP and OFFICE2003 is available under Windows 98, 2000, XP and 2003.
- NATIVEXP is only available when running under Windows XP (or 2003 with Windows Visual Themes switched on).

Tooltip

- Balloon type tooltip is only available on WindowsXP and later.

4.4 Contact us

Upgrades

The latest upgrades can be downloaded from www.icetips.com Note that you must have a valid Gold Subscription login to download.

Bug reports and suggestions

If you want to contact us, please send us emails to support@icetips.com

If your app is a multi-dll app, please try to recompile all your apps before reporting possible bugs.

When reporting bugs, please let us know your version of Clarion and PowerToolbar, and if possible, provide a screen shot. An example app or steps to reproduce in the demo app is highly appreciated.

Please report bugs to our bug tracking system at http://icetips.fogbugz.com Make sure that you select the proper project (PowerToolbar) and the correct area (Class/Documentation/Install/Misc/Templates) and give us good, detailed description of what the problem is. You can attach files to your bug reports.

Other Clarion Products

You'll find more of our Clarion products at www.icetips.com

4.5 Installed files

The following files are installed as part of PowerToolbar:

```
C:\Clarion6\3rdParty\LibSrc\potoolbar.inc
C:\Clarion6\3rdParty\LibSrc\potoolbar.clw
C:\Clarion6\3rdParty\LibSrc\potoolctrl.inc
C:\Clarion6\3rdParty\LibSrc\potoolctrl.clw
C:\Clarion6\3rdParty\Template\potoolbar.tpl
C:\Clarion6\3rdParty\Images\ptb_checkon.ico
C:\Clarion6\3rdParty\Images\ptb_checkoff.ico
C:\Clarion6\3rdParty\Docs\PowerToolbar\PowerToolbar.chm
C:\Clarion6\3rdParty\Examples\PowerToolbar\Quickstart.app
C:\Clarion6\3rdParty\Examples\PowerToolbar\BrowseTutorial.app
C:\Clarion6\3rdParty\Examples\PowerToolbar\AdvancedBrowseTutorial.app
C:\Clarion6\3rdParty\Examples\PowerToolbar\AppToolbarTutorial.app
C:\Clarion6\3rdParty\Examples\PowerToolbar\MenusTutorial.app
C:\Clarion6\3rdParty\Examples\PowerToolbar\ptbexample.dct
C:\Clarion6\3rdParty\Uninstall\Uninst_PowerToolbar.exe
C:\Clarion6\3rdParty\Uninstall\Uninst_PowerToolbar.log
```

For an upto date list of the files, please see:

http://www.icetips.com/productbuilds/PowerToolbar_install.htm
http://www.icetips.com/productbuilds/PowerToolbar_installC7.htm

4.6 Version History

Version 2018.10.210.196 [October 14, 2018]

July 8, 2016:

▪ Classes	Sometimes the control would not be refreshed correctly after a window resize. This is hopefully fixed now.

May 10, 2016:

▪ Template/Classes	In some cases an "Unknown Identifier" error was thrown on the PTB:IpMsImgLib global variable. Some modifications to try to catch why this is happening.
▪ Classes	Regression: XPTheme themes no longer worked. Fixed.
▪ Classes	Regression: Drop Buttons would cause a GPF. Fixed.

February 22, 2016:

▪ Classes	There was no method to set the currently selected Drop Combo item. Added SelectComboItem.

November 17, 2015:

▪ Classes	Item IDs from DropButtons were not working correctly. Fixed.
▪ Classes	SetItemEnabled in DropButtonClass did not work correctly. Fixed.
▪ Installer	Installer is now dual code signed.

March 17, 2015:

▪ Classes	In some circumstances the menu text was hidden. Fixed
▪ Classes	MDI buttons on maximized MDI windows do not work properly. This has been found to be a bug in the Clarion runtime library in both Clarion 9.1 and 10-Beta.

Version 2.1.186 [February 24, 2015]

July 14, 2014:

▪ Classes	PTB:ShowScrollBar caused "Calling function as a procedure" warning.

	Fixed.

July 2, 2014:

▪ Classes	Horizontal scrollbar could show up at the bottom of Powertoolbar controls in Clarion 9.1. build 11014 or newer. Fixed.

Version 2.1.181 [January 28, 2014]

January 28, 2014:

▪ Installer	Installer is now Clarion 9.1 compatible.

November 16, 2013:

▪ Classes	Drop down arrows were not drawn or usable on drop down combo controls. Fixed.

August 6, 2013:

▪ Template	Code generation for Drop Down Buttons could be wrong if there were multiple Drop Down Buttons on the same window. Fixed.

Version 2.1.175 [May 19, 2013]

May 19, 2013:

▪ Product	Files from XPTheme are now included with PowerToolbar and installed along with it. This includes: templates\xptheme.tpl libsrc\xptheme.clw libsrc\xptheme.inc libsrc\xpthemeBoxes.clw libsrc\xpthemedc.clw libsrc\xpthemedc.inc libsrc\xpthemeDrops.clw libsrc\xpthemeEntries.clw libsrc\xpthemeeq.inc libsrc\xpthemeListboxTakePaint.clw libsrc\xpthemelistcolumns.clw libsrc\xpthemeLists.clw libsrc\xpthemeRadios.clw libsrc\xpthemeSheets.clw libsrc\xpthemeSpins.clw libsrc\xptheme_group_ending.clw In Clarion 6 and older those files will be in %ROOT%\3rdParty but in Clarion 7 and newer they will be in accessory\template\win and accessory\libsrc\win.

June 29, 2012:

■ Documentation	Updated documentation for the Toolbar template.

June 29, 2012:

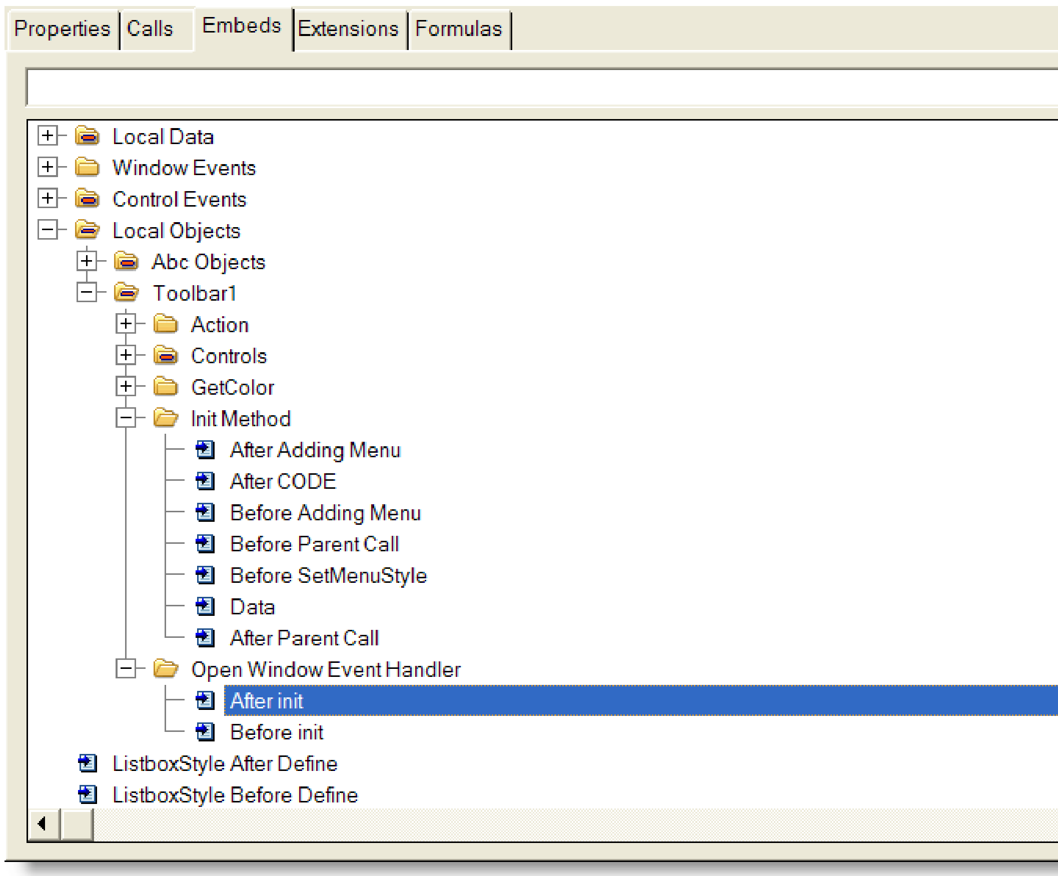
■ Classes

The "Delay .INIT" could cause problems if embedded code had been added to call methods in the PowerToolbar class and that code was now executed before the .INIT. A check has been added that will throw an error message if that happens and halt the program. It will indicate the program name and the method being called.

In hand coded projects, you must now set the SELF.InitCalled property to TRUE right after the call to AppToolbar.Init() in the derived POToolbarClass.Init method. This is currently done by the templates to generate code like this:

```
Toolbar1.Init     PROCEDURE()
! Start of "Init Method Data section"
! End of "Init Method Data section"
    Code
! Start of "Init Method After Code statement"
! End of "Init Method After Code statement"
    AppToolbar.Init(Toolbar1)
    SELF.InitCalled = True
```

To avoid problems, please use one of the embeds that are executed after the INIT method is called. See:



The "Open Window Event Handler" embeds are the ones created around the call to the PowerToolbar.Init method. The name of this embed changes depending on if the "Delay Toolbar.Init" option is checked on the "Advanced" tab on the PowerToolbar control template, but the code will always be correctly placed before or after the call to the PowerToolbar.Init method. You can also use the "Init Method | After Parent Call" embed. It is generated inside the derived INIT method.

Version 2.0.165 [June 20, 2011]

June 26, 2012:

■ Classes	A Message() had been accidentally left inside the DropDownMenu method of the DropButtonClass. Fixed.

Version 2.0.163 [June 19, 2011]

March 27, 2012:

■ Classes	GetItemID method did not return correct results for DropComboButton if no ItemID was specified. Fixed. Returns 1 based index of the selected item in the list.

December 14, 2011:

▪ Template	Delay Initialization was set to true for windows other than the application frame. Fixed.
▪ Classes	SetText method did not select the item in a Drop Combo control, it just set the text. Fixed.

November 1, 2011:

▪ Classes	Under some circumstances windows were not removed from the window list. Seems to be tied mostly to windows with toolbar caption. Fixed.
▪ Classes	10th item in Windows list menu should show "More windows..." which opens a window with a list of all open windows. Instead it was listing the 10th open window. Fixed.
▪ Classes	Some debug code was still left in some methods. Fixed.

October 28, 2011:

▪ Classes	Window list was not being created correctly. Text was missing from the window list items and the active window was not indicated. Fixed. Note that menus on MDI windows are NOT supported, neither merged or non-merged. Sent to beta testers.

July 26, 2011:

▪ Template	Added option to delay initialization of the PowerToolbar classes to prevent duplicate menus.

June 2, 2011:

▪ Classes	SetBandHidden and GetBandHidden were not implemented in the Global class. Fixed.
▪ Template	Added SetBandHidden and GetBandHidden to the export section of the template.
▪ Documents	Added documentation about SetBandHidden, GetBandHidden, SaveBandVisible and RestoreBandVisible.

Version 2.0.151 BETA [May 4, 2011]

May 4, 2011:

■ Installer	Installer built for Clarion 8.0

March 4, 2011:

■ Classes	Further refinements for the Visible vs. Hidden options. bUserHidden has been taken out of use as it does not serve any purpose any more.

March 3, 2011:

■ Classes

Added SetBandHidden and GetBandHidden. Those methods are designed to remove a band and hide it. The band will not show up in the band customize menu and cannot be made visible with SetBandVisible. SetBandVisible only hides the band and does not remove it from the cusomize menu. Use SetBandHidden to remove and bands from the user for example based on user access privileges.

■ Classes

If a Clarion dropdown list or dropdown combo was used as a control in the toolbar it would not receive EVENT:NewSelection. Fixed.

■ Classes

Added SaveBandVisible(String pINIFileName, String pProcedureName) and RestoreBandVisible(String pINIFileName, String pProcedureName). Those methods take a INI filename and a Procedure name and save the status of each band to the specified INI file, like this:

```
[Main]
PowerToolbar20000_Hidden=0
PowerToolbar20000_Visible=1
PowerToolbar1000_Hidden=0
PowerToolbar1000_Visible=1
PowerToolbar1001_Hidden=0
PowerToolbar1001_Visible=1
PowerToolbar1002_Hidden=0
PowerToolbar1002_Visible=1
PowerToolbar1003_Hidden=0
PowerToolbar1003_Visible=1
PowerToolbar1004_Hidden=0
PowerToolbar1004_Visible=1
PowerToolbar1005_Hidden=0
PowerToolbar1005_Visible=1
```

Call the RestoreBandVisible in ThisWindow.Init, after the Powertoolbar is initialized, around priority around 8300)

Make sure that you call it before any calls to SetBandHidden!

Call the SaveBandVisible in ThisWindow.Kill, around priority 7500.

The RestoreBandVisible refreshes the toolbar so it is completely redrawn. Example

```
Toolbar1.RestoreBandVisible(INIMgr.Filename, 'Main')

Toolbar1.SaveBandVisible(INIMgr.Filename, 'Main')
```

February 14, 2011:

▪ Templates	Modification to global and procedure templates to allow font name and size to be specified by variables. Use ! and variablename to use a variable, for example !Glo:FontName or !Glo:FontSize.
▪ Classes	Added omit-able parameter to SetFont that specifies the character set to use in all texts created by PowerToolbar.
▪ Templates	Added option to global and procedure templates to specify character set. It can be either a numeric value, a CHARSET: constant or a variable preceded with an exclamation mark (for example !Loc:CharSet)
▪ Build	Our build process now includes building a test multi-DLL application to make sure that classes are exported correctly. This will prevent problems with "unresolved external" methods if the template is not updated if new methods are added. This involves no changes to the deployed product.
▪ Install	The installer was set to accept Clarion 5.5 as a supported Clarion version. PowerToolbar is NOT compatible with Clarion 5.5, only Clarion 6.0 and later.

February 5, 2011:

▪ Classes	Menu theming was unreliable and could cause various corruptions in the menus. This was particularly noticeable on Vista and Windows 7. Fixed.

January 10, 2011:

▪ Template	Template would throw "Unknown Variable '%EnableRunTimeTranslator'" in Legacy apps. Fixed.

Version 2.0.140 [December 21, 2010]

August 3, 2010:

▪ Classes	If a band was hidden and the option to customize bands was turned on, the band would still show up in the drop down menu and the customize window. Fixed.

June 21, 2010:

▪ Templates	Added a condition field to the Menu tab so the menu override can be conditional. For PowerToolbar to theme the menu, the condition must be true. If no condition is specified, no IF/END statement is generated.
▪ Templates	Added several embeds into the Init method to make customizing the initialization easier.

Version 2.0.134 [March 15, 2010]

March 3, 2010:

■ Classes	End shadow width (Office 2003 theme) could not be changed. Added SetEndShadowWidth and GetEndShadowWidth methods and EndShadowWidth property which now can be used to control the width of the end shadow. Default is set to 8 in the constructor, same as it was originally.

December 7, 2009:

■ Classes	Regression: Separators in Drop Button items still did not work correctly. In testing.

December 1, 2009:

■ Classes	Separators in Drop Button items did not work correctly. Fixed.

November 30, 2009:

■ Classes	Items in Drop Button menu could not be enable/disabled. Fixed
■ Classes	Added SetItemEnabled method to enable/disable items in Drop Button menu.
■ Classes	Mimic buttons did not mimic icons when the mimiced button changed. Fixed.
■ Classes	Mimic buttons did not mimic disable/enable state when the mimiced button changed. Fixed.
■ Classes	Mimic buttons did not mimic hide/unhide state when the mimiced button changed. Fixed. Note that for this to work the mimic control must have the "Show/Hide" checked in the "Mimic" section of the template.
■ Classes	Mimic buttons did not mimic tooltips when the mimiced button changed. Fixed.

November 7, 2009:

■ Classes	Added new CalculateMenuHotkeyW (protected) method to calculate the width of the hotkey text.
■ Classes	Still an issue with the hotkeys not being drawn correctly. Fixed.

November 6, 2009:

■ Classes	Hotkey text was not drawn correctly on hovering on menus with submenus. Fixed.

October 26, 2009:

■ Templates	If multiple procedure that used PowerToolbar MDI extension were in the same source module, the ID file would only be included once since the Include() statement was generated with ",ONCE" attribute. Fixed.

October 22, 2009:

■ Classes	GetItemID on DropButtons would always return the last entry. Fixed.

October 12, 2009:

■ Classes/Templ.	DropButton items now support icons in both classes and templates.

September 18, 2009:

■ Classes	DropButton menus with separators would not work correctly and would return the wrong ItemID based on the selected menu item. Fixed.

July 15, 2009:

■ Classes	Focus was moved from the control with focus if a button in the toolbar was clicked. Fixed. Note: Concern for issues this might cause elsewhere but tests have not shown any side effects so far.

June 1, 2009:

■ Classes	The GetItemID would not return the correct ItemID on Drop Buttons. Fixed.

Version 2.0.112 [April 15, 2009]

March 9, 2009:

■ Template	Template did not use runtime translations. Fixed - thanks to Bo Schmitz.
■ Classes	When running using Windows Classic style, the toolbar would attempt to theme Clarion controls with mixed results. Fixed - Clarion controls are no longer themed when using Windows Classic style.

January 5, 2009:

■ Classes	Clarion controls in toolbars were not themed. Fixed.
■ Classes	Clarion controls in toolbars on dockable windows were not moved into the toolbar band. This is a limitation that has to be fixed in code. Remove the DOCK and DOCKED attributes from the window (uncheck everything in the Dock group on the Extra tab in the window properties) and then set the docking options using PROP:DOCK and PROP:DOCKED in ThisWindow.Init, priority 8110. This fixes the problem

January 1, 2009:

- The application would GPF when it was closing if the "Add functionality to the Frame" extension was added to the frame and the "Enable Tray Icon" was checked. Generated code for the .INIT method moved down from priority 8000 to 1 in %AfterWindowOpening embed. Fixed.

Version 2.0.105 [December 29, 2008]

- All code and documentation modified from PowerOffice AS to Icetips Creative, Inc.
- Bug tracking set up at http://icetips.fogbugz.com
- Added link to bug tracking to Global template
- Updated link to website to point to www.icetips.com
- Added ITRun32.dll to install
- Version information added to global template
- Added documentation for new methods, such as AddClarionControl, DelayRefresh, GetActiveTab, GetMenuStyle, InitControl, IsClosing, SetActiveTab, SetFocus and SetMenuStyle
- Previous beta changes are all implemented in public release - see below:

1.0 BETA 7 - not released publicly before

- New: Support for merging MDI menus.
- New: After-action embeds for dropbutton and dropitem actions
- Changed: Template prompts for menu overrides now only shows on appframe and windows without MDI attribute
- Changed: Since band customize button has no function, it is now default off
- Fixed: Bug with MDI buttons and hidden windows
- Fixed: Button didn't hover if mouse was over it on when unhidden
- Fixed: Alt key didn't trigger menu keyboard mode
- Fixed: RadioButtons separated by another control type would still act as one group of radios
- Fixed: Submenus was overdrawn by parent menu border
- Fixed: Hidden/Disabled menus didn't become hidden/disabled on window open
- Fixed: Default color setting would generate gray toolbars even when XP-Theme was present
- Fixed: Icons wasn't reloaded after calls to SetIconHeight/Width
- Fixed: Menu opened at wrong position when appframe had layout set to right-to-left
- Fixed: Disabled menu items with hotkey is drawn incorrectly in Win 2000 style.

1.0 BETA 6

- New: MimicButton can now mimic visibillity
- New: Up/Down-buttons in Customize window
- New: 'After Action'-embed
- New: 'After Accepted'-embed for controls
- New: Global setting for toolbar style, color, font and tooltip
- New: Option to let menustyle follow toolbar
- New: Option for adding separators in front and/or after buttons and clarion controls
- New: Option for 'Prompt' on ClaironControls
- New: Option to not auto-resize toolbar height
- New methods: IsClosing, SetActiveTab, GetActiveTab
- Changed: Prototype for FixColor has changed to support Color:None
- Changed: Added 5 pixels to the width of menu buttons
- Changed: Menu separators is now purely handled by API
- Changed: Text-controls draw text one pixel lower
- Changed: Header band does not draw border when Windows Theme colors isn't active
- Changed: Flags for AddMimicButton() has changed
- Changed: DropButton and SplitButton now handles items with '-' in the text field as separators
- Changed: Clarion-contol now use the design-time width of the clarion control
- Changed: 'Allow Right-click customize' is now default off
- Fixed: Hide/Disable bug for Clarion-control type
- Fixed: If last item before Window-list was toggle, each window-item would become toggled to
- Fixed: GPF on Windows 2000 when hovering tabs
- Fixed: DropCombo and Entry control would grab focus on window open
- Fixed: Width calculation of Tabs
- Fixed: Wrong symbol was tested in the template and could produce compile errors with DropItem-actions

- Fixed: FixedWidth on Clarion-control didn't work
- Fixed: Id-file could contain wrong IDs
- Fixed: If FixedWidth was set on a TextControl, the height would be set too
- Fixed: Clarion-controls didn't get EVENT:Newselection
- Fixed: Clarion-controls got reversed tab-order
- Fixed: DropCombo and Entry didn't properly refresh on SetVisible
- Fixed: Global class wasn't threadsafe and some code was executed in the wrong thread
- Fixed: Menus didn't support enabled/disabled
- Fixed: Menus didn't support hide/unhide

1.0 BETA 5

- Change: Text-control now support icon and font settings
- Change: Prototype for AddBand got a new optional BandType argument
- Change: Menu now has separate style setting
- New methods: SetMenuStyle, GetMenuStyle
- New method: AddClarionControl
- New method: DelayRefresh
- New method: SetFocus
- New method: DeleteItem for deleting Drop or Combo items
- New: Iconnames in the form of FileName.exe[1] and FileName.dll[n] is now supported
- New: ProcessButton and Clarion-control
- New: Header band
- New: Tabs band
- Fixed: GPF when restoring MDI-child or resizing AppFrame
- Fixed: AppFrame toolbar draws background on init before adding controls to avoid bleed through
- Fixed: Balloon tooltip mode didn't work
- Fixed: Ampersand was removed from tooltip text
- Fixed: Menu hotkey would always open first menu if a MDI-child was open
- Fixed: Entry-control now accepts Tab and Enter keys
- Fixed: Mimiced buttons now check state on window open
- Fixed: Band width was wrong when a mimiced button changed to wider text
- Fixed: SetTooltipMode ddin't work after toolbar init
- Fixed: Icons after STD:WindowList could appear on wrong item
- Fixed: Icons on menu items with subitems didn't appear
- Fixed: Toggle items wasn't handled properly
- Fixed: GPF on menu close

1.0 BETA 4

- Added: Call to Update before mimic is executed
- Change: AppFrame toolbar is now subclassed instead of the control. Resizing should be more stable.
- Change: Init was renamed to InitControl, and a new Init method was added
- Change: Parent Init-call is now called after template generated init code
- Fixed: Calculate method was called about 10 times every resize.
- Fixed: AppFrame toolbar 'jumping' on window open and resize

1.0 BETA 3

- Fixed: Resize refresh issues (hopefully)
- Fixed: Customize-list could become empty

1.0 BETA 2

- Fixed: GPF when using PowerXP-Colors or Office2003-style on non-XP Windows
- Added: Option to ignore band in right-click customize menu
- Added: Option to limit size of DropCombo droplist and automatic scrollbar
- Added methods: GetControlBandID, UncheckAll, Set/GetBandShowText, Set/GetBandIconsAbove, IsControlInBand, SetMaxComboItems
- Added: 'Refresh IDs' button in MDI-template
- Fixed: balloon tooltip mode is now working
- Added: embed-button for items in template

1.0 BETA 1

- First beta version

Part V

Chapter 5 - Reference

5 Reference

5.1 Embed points

In this section you'll get an overview of the various embed-points generated by PowerToolbar.

All embeds are generated under "**Local Objects -> Toolbar1**". Toolbar1 will be the object name set in the template.

Toolbar embeds

Toolbar1 -> Init -> Before Init
This embed is executed just before the toolbar init code.

Toolbar1 -> Init -> After Init
This embed is the best place to add your own controls by hand code (if you need to do that).

Toolbar1 -> Action -> Before Action
Code in this embed is executed when controls action is about to execute

Toolbar1 -> Action -> After Action
Code in this embed is executed after controls action has been executed

Toolbar Control embeds

For each control an embed will be generated based on the ID of the control. For instance, if the ID of the control is ID_BUTTON1 the following embed will be generated:

Toolbar1 -> Controls -> ID_BUTTON1 -> Accepted
This embed is executed whenever a control has been clicked or when an item has been selected from a DropEntry or DropButton. This embed will also be called when text in a Entry control has changed (and been accepted by focus-change or enter/tab key).

Toolbar1 -> Controls -> ID_BUTTON1 -> NewSelection
This embed is only generated for DropEntries and Entry controls. Executed whenever the user inputs characters into a DropEntry or Entry control.

Drop Item embeds

For each item of a DropCombo or DropButton, an embed is executed just before the item action:

Toolbar1 -> Controls -> ID_DROPCOMBO1 -> Item1 -> Accepted

Toolbar color embed

When color style is set to "Gray" or "Widbey" an embed for colors is generated. In this embed you can override each color.

Toolbar1 -> GetColor -> Before generated code

5.2 Methods by category

Toolbar
- GetDrawBottomBorder
- GetTooltipMode
- Init
- Refresh
- SetAppName
- SetAllowCustomize
- SetDrawBottomBorder
- SetFont
- SetOverflowDetection
- SetPXP
- SetStyle
- SetText
- SetTooltipMaxWidth
- SetTooltipMode

Band methods
- AddBand
- DeleteBand
- GetBandIconsAbove
- GetBandShowIcons
- GetBandShowText
- GetBandVisible
- GetFixedHeight
- GetFixedWidth
- IsControlInBand
- SetBandControlsEnabled
- SetBandIconsAbove
- SetBandShowIcons
- SetBandShowText
- SetBandVisible
- SetIconHeight
- SetIconWidth
- UncheckAll

Colors
- GetColor
- GetStyle
- SetDefaultColors

Control methods
- AddButton
- AddCombo
- AddComboItem
- AddDropButton
- AddDropItem
- AddEntry
- AddMenu
- AddMimicButton
- AddSeparator
- AddSpacer
- AddText
- DeleteControl
- DeleteItem
- DeleteItems
- ExecuteControl
- GetCheckbox
- GetChecked
- GetEnabled
- GetIcon
- GetItemID
- GetItemText
- GetDisabledIcon
- GetRadio
- GetReadOnly
- GetText
- GetVisible
- IsControlInBand
- SetCheckbox
- SetChecked
- SetEnabled
- SetFixedHeight
- SetFixedWidth
- SetIcon
- SetDisabledIcon
- SetMaxComboItems
- SetRadio
- SetReadOnly
- SetText
- SetTooltip
- SetVisible
- SetWidth
- GetPrompt
- SetPrompt
- SetPromptWidth
- GetPromptWidth
- GetControlBandID

AppToolbar (Global object)
- SetBandControlsEnabled
- SetBandVisible
- SetChecked
- SetEnabled
- SetText
- SetVisible
- GetBandVisible
- GetChecked
- GetEnabled
- GetText
- GetVisible

5.3 Methods by alphabet

- AddBand
- AddButton
- AddCombo
- AddComboItem
- AddDropButton
- AddDropItem
- AddEntry
- AddMenu
- AddMimicButton
- AddSeparator
- AddSpacer
- AddText
- DeleteBand
- DeleteControl
- DeleteItem
- DeleteItems
- ExecuteControl
- GetBandIconsAbove
- GetBandShowIcons
- GetBandShowText
- GetBandVisible
- GetCheckbox
- GetChecked
- GetColor
- GetControlBandID
- GetDisabledIcon
- GetDrawBottomBorder
- GetEnabled
- GetFixedHeight
- GetFixedWidth
- GetIcon
- GetItemID
- GetItemText
- GetPrompt
- GetPromptWidth
- GetRadio
- GetReadOnly
- GetStyle
- GetText
- GetTooltipMode
- GetVisible
- Init
- IsControlInBand
- Refresh
- SetAppName
- SetAllowCustomize
- SetBandControlsEnabled
- SetBandIconsAbove
- SetBandShowIcons
- SetBandShowText
- SetBandVisible
- SetCheckbox
- SetChecked
- SetDefaultColors
- SetDrawBottomBorder

- SetEnabled
- SetFixedHeight
- SetFixedWidth
- SetFont
- SetIcon
- SetDisabledIcon
- SetIconHeight
- SetIconWidth
- SetMaxComboItems
- SetOverflowDetection
- SetPrompt
- SetPromptWidth
- SetPXP
- SetRadio
- SetReadOnly
- SetStyle
- SetText
- SetTooltip
- SetTooltipMaxWidth
- SetTooltipMode
- SetVisible
- SetWidth
- UncheckAll

5.4 Equates

This section contains an overview of the various equates available to the developer.

Styles	
PTB:STYLE_OFFICE2000	Office 2000 style
PTB:STYLE_OFFICEXP	Office XP style
PTB:STYLE_OFFICE2003	Office 2003 style
PTB:STYLE_NATIVEXP	Native XP style
Band types	
PTB:BT_NORMAL	Normal band
PTB:BT_MENU	Menu band
PTB:BT_TABS	Tabs band
PTB:BT_HEADER	Header band
Band flags	
PTB:BAND_LASTONLINE	Band is last on line, next band will start on next row
PTB:BAND_ICONSABOVE	Draw icons above text on buttons
PTB:BAND_HIDETEXT	Do not show text
PTB:BAND_HIDEICONS	Do not show icons
PTB:BAND_FULL	Band extends to the right border of the toolbar
PTB:BAND_HIDE	Band is initially hidden
PTB:BAND_FLAT	Band is flat (no borders)
PTB:BAND_NOGRIP	Do not draw gripper
PTB:BAND_NOCUSTOMIZE	Do not draw customize area (only Office2003)
PTB:BAND_NORIGHTCUSTOMIZE	Band is not included in right-click customize
Control flags	
PTB:BTN_CHECK	Button is checkbox
PTB:BTN_RADIO	Button is radiobutton
PTB:BTN_DISABLED	Control is initially disabled
PTB:BTN_CHECKED	Control is initially checked
PTB:BTN_HIDE	Control is initially hidden
PTB:BTN_READONLY	Control is read-only (only applies to DropEntry and Entry)
PTB:BTN_SPLIT	DropButton is split-button
PTB:BTN_NOTEXTMIMIC	Do not mimic text
PTB:BTN_NOICONMIMIC	Do not mimic icon
PTB:BTN_NOTOOLTIPMIMIC	Do no mimic tootlip
PTB:BTN_RESIZE	Resize control (only applies to DropEntry, Entry and Spacer)
Mdi buttons ID	
PTB:ID_MINIMIZE	ControlID of the MDI Minimize button
PTB:ID_RESTORE	ControlID of the MDI Restore button
PTB:ID_CLOSE	ControlID of the MDI Close button
Color ID's	
PTBCOL:BtnHighLight	
PTBCOL:BtnShadow	
PTBCOL:BtnPressed	
PTBCOL:Text	
PTBCOL:DisabledText	

PTBCOL:DisabledTextShadow PTBCOL:Background PTBCOL:MenuBackground PTBCOL:MenuBorder_Highlight PTBCOL:MenuBorder_Shadow PTBCOL:MenuSelection PTBCOL:MenuSelectionBorder PTBCOL:MenuSeparator PTBCOL:MenuText PTBCOL:OFFXP_BtnHighLight PTBCOL:OFFXP_BtnShadow PTBCOL:OFFXP_BtnFace PTBCOL:OFFXP_BtnHover PTBCOL:OFFXP_Border PTBCOL:OFFXP_DarkBorder PTBCOL:OFFXP_Text PTBCOL:OFFXP_DisabledText PTBCOL:OFFXP_Background_From PTBCOL:OFFXP_Background_To PTBCOL:OFFXP_MenuBackground PTBCOL:OFFXP_MenuBorder PTBCOL:OFFXP_MenuSelection PTBCOL:OFFXP_MenuSelectionBorder PTBCOL:OFFXP_MenuSeparator PTBCOL:OFFXP_MenuText PTBCOL:OFF23_White PTBCOL:OFF23_DarkBlue PTBCOL:OFF23_Gradient_From PTBCOL:OFF23_Gradient_To PTBCOL:OFF23_DarkGradient_From PTBCOL:OFF23_DarkGradient_To PTBCOL:OFF23_Line PTBCOL:OFF23_DarkLine PTBCOL:OFF23_Border PTBCOL:OFF23_BtnGradient_From PTBCOL:OFF23_BtnGradient_To PTBCOL:OFF23_BtnHvrGradient_From PTBCOL:OFF23_BtnHvrGradient_To PTBCOL:OFF23_Text PTBCOL:OFF23_DisabledText PTBCOL:OFF23_Background_From PTBCOL:OFF23_Background_To PTBCOL:OFF23_MenuBackground PTBCOL:OFF23_MenuBorder PTBCOL:OFF23_MenuSelection PTBCOL:OFF23_MenuSelectionBorder PTBCOL:OFF23_MenuSeparator PTBCOL:OFF23_MenuText	

5.5 Toolbar Methods

5.5.1 AddBand Toolbar Methods

Add a band to the toolbar. You must add a band to be able to add controls. Each band is a container for other toolbar controls.

Prototype:

```
AddBand(LONG ID, ULONG Flags=0, <STRING BandName>, LONG nType=0)
```

Arguments:

ID	The ID for this band. Must be unique
Flags	Flags that modifies look and behaviour of the band. This can be one or more of the following flags:
	PTB:BAND_LASTONLINE Forces the next band to start on the next row
	PTB:BAND_ICONSABOVE Place icons above text on buttons
	PTB:BAND_HIDETEXT Do not show button text
	PTB:BAND_HIDEICONS Do not show button icons
	PTB:BAND_FULL Band extends to full width of the toolbar (this forces LASTONLINE)
	PTB:BAND_HIDE Band starts hidden
	PTB:BAND_FLAT Band is drawn flat (no borders)
	PTB:BAND_NOGRIP Do not draw band gripper
	PTB:BAND_NOCUSTOMIZE Do not draw band customize area (ignored when style isn't OFFICE2003)
BandName	Optional, A string containing a name to use in the customize list. If this name is omitted, the band will not be shown in the customize list.
nType	Band type. Supported values: PTB:BT_NORMAL, PTB:BT_TABS and PTB:BT_HEADER. Defaults to PTB:BT_NORMAL.

See also:

DeleteBand

Example:

```
Toolbar1.AddBand(ID_MYBAND, PTB:BAND_LASTONLINE + PTB:BAND_FLAT)
```

5.5.2 AddButton Toolbar Methods

Adds a toolbar button to a band

Prototype:
AddButton(LONG BandID, LONG ID, STRING Caption, <STRING IconName>, ULONG Flags=0, <LONG InsertAfterID>)

Arguments:

BandID	ID of the band the control should be added to
ID	The unique ID of this control
Caption	The text for this control
IconName	Optional. Icon name
Flags	Optional. button style flags. Can be one or more of the following flags: *PTB:BTN_CHECK* This is a checkbox button *PTB:BTN_RADIO* Radiobutton *PTB:BTN_DISABLED* Control is initially disabled *PTB:BTN_CHECKED* Control is initially checked *PTB:BTN_HIDE* Control is hidden *PTB:BTN_READONLY* Control is read only
InsertAfterID	Optional. Control should be inserted after the control with InsertAfterID. If you specify 0 as InsertAfterID, the control will be inserted first in band.

See also:
AddBand, AddCombo, AddDropButton, AddEntry, AddMenu, AddSpacer, AddText

Example:
```
  Toolbar1.AddButton(TB_SomeBand, ID_MyButton, 'Hello World!')
  Toolbar1.AddButton(TB_SomeBand, 10001, 'Hello World!', 'MyIcon.ico',
PTB:BTN_CHECK+PTB:BTN_CHECKED)
```

5.5.3 AddClarionControl Toolbar Methods

Adds a Clarion control to a band

Prototype:
AddClarionControl(LONG BandID, LONG ID, LONG Feq, <STRING szPrompt>, LONG Flags=0, <LONG InsertAfterID>)

Arguments:

BandID	ID of the band the control should be added to
ID	The unique ID of this control

Feq	Clarion Field Equate Label of the control
szPrompt	Not used
Flags	Optional. button style flags. Can be one or more of the following flags: *PTB:BTN_DISABLED* Control is initially disabled *PTB:BTN_HIDE* Control is hidden *PTB:BTN_READONLY* Control is read only
InsertAfterID	Optional. Control should be inserted after the control with InsertAfterID. If you specify 0 as InsertAfterID, the control will be inserted first in band.

5.5.4 AddCombo Toolbar Methods

Adds a DropCombo to a band. A DropCombo typically has several combo items the user can choose from. However, text can freely be entered into the entry part of the control. To avoid this, specify the PTB:BTN_READONLY flag.

Prototype:
AddCombo(LONG BandID, LONG ID, STRING Txt, LONG Flags=0, <LONG InsertAfterID>)

Arguments:

BandID	ID of the band the control should be added to
ID	The unique ID of this control
Txt	The default text for the entry part of the control
Flags	Optional. button style flags. Can be one or more of the following flags: *PTB:BTN_DISABLED* Control is initially disabled *PTB:BTN_HIDE* Control is hidden *PTB:BTN_READONLY* Control is read only
InsertAfterID	Optional. Control should be inserted after the control with InsertAfter-ID. If you specify 0 as InsertAfterID, the control will be inserted first in band.

See also:
AddComboItem, AddBand

Example:
```
Toolbar1.AddCombo(TB_SomeBand, ID_DropCombo1, 'Hello world!', PTB:BTN_READONLY)
```

5.5.5 AddComboItem Toolbar Methods

Adds a drop item to a DropCombo control.

Prototype:
AddComboItem(LONG ComboID, STRING Txt, <LONG ItemID>)

Arguments:

ComboID	ID of an already existing DropCombo
Txt	Display text for this item
ItemID	Optional. Item ID

See also:
AddCombo, AddBand

5.5.6 AddDropButton Toolbar Methods

Adds a DropButton or SplitButton to a band.

Prototype:
AddDropButton(LONG BandID, LONG ID, STRING Caption, <STRING IconName>, ULONG Flags=0, <LONG InsertAfterID>)

Arguments:

BandID	ID of the band the control should be added to
ID	The unique ID of this control
Caption	Text for this control
IconName	Optional. Icon name
Flags	Optional. Flags to modify behaviour. Can be one or more of the following: *PTB:BTN_DISABLED* Control is initially disabled *PTB:BTN_HIDE* Control is hidden *PTB:BTN_SPLIT* DropButton is split-button
InsertAfterID	Optional. Control should be inserted after the control with InsertAfter-ID. If you specify 0 as InsertAfterID, the control will be inserted first in band.

See also:
AddDropItem, AddBand

5.5.7 AddDropItem — Toolbar Methods

Add a menu-item to a DropButton or SplitButton.

Prototype:
AddDropItem(LONG DropButtonID, STRING Txt, <LONG ItemID>)

Arguments:

DropButtonID	ID of an already existing DropButton
Txt	Display text for the item
ItemID	Optional, Item ID (eg. SysID)

See also:
AddDropButton

5.5.8 AddEntry — Toolbar Methods

Add an entry control to a band.

Prototype:
AddEntry(LONG BandID, LONG ID, STRING Txt, LONG Flags=0, <LONG InsertAfterID>)

Arguments:

BandID	ID of the band the control should be added to
ID	The unique ID of this control
Txt	The text for this control
Flags	Optional. button style flags. Can be one or more of the following flags: *PTB:BTN_DISABLED* Control is initially disabled *PTB:BTN_HIDE* Control is hidden
InsertAfterID	Optional. Control should be inserted after the control with InsertAfter-ID. If you specify 0 as InsertAfterID, the control will be inserted first in band.

See also:
GetText, SetText, AddBand

5.5.9 AddMenu — Toolbar Methods

Prototype:
AddMenu(LONG StartID=20000, LONG Flags=0)

Arguments:

StartID	Optional. Start value for automatic menu ID generator
Flags	Optional. Flags that modify looks or behaviour. Can be one or more of the following flags: *PTB:BAND_NOGRIP* Do not draw gripper *PTB:BAND_FLAT* Menu band is flat (no borders)

Example:

```
Toolbar1.AddMenu()
```

5.5.10 AddMimicButton Toolbar Methods

Adds a mimic button to a band

Prototype:
AddMimicButton(LONG BandID, LONG ID, LONG Feq, ULONG Flags=0, <LONG InsertAfterID>)

Arguments:

BandID	ID of the band the control should be added to
ID	The unique ID of this control
Feq	The field equate label for the button to mimic
Flags	Optional. button style flags. Can be one or more of the following flags: *PTB:BTN_HIDE* Control is hidden *PTB:BTN_NOTEXTMIMIC* Do not mimic text *PTB:BTN_NOICONMIMIC* Do not mimic icon *PTB:BTN_NOTOOLTIPMIMIC* Do not mimic tooltip
InsertAfterID	Optional. Control should be inserted after the control with InsertAfterID. If you specify 0 as InsertAfterID, the control will be inserted first in band.

See also:
AddBand, AddCombo, AddDropButton, AddEntry, AddMenu, AddSpacer, AddText

5.5.11 AddSeparator Toolbar Methods

Add a separator control. This control will draw a vertical line that separates groups of controls

Prototype:
AddSeparator(LONG BandID, LONG ID=0, LONG Flags=0, <LONG InsertAfterID>)

Arguments:

BandID	ID of the band the control should be added to
ID	The unique ID of this control
Flags	Not in use! Always omit, or set to 0
InsertAfterID	Optional. Control should be inserted after the control with InsertAfter-ID. If you specify 0 as InsertAfterID, the control will be inserted first in band.

5.5.12 AddSpacer Toolbar Methods

Adds space between controls.

Prototype:
AddSpacer(LONG BandID, LONG ID, LONG nWidth=-1, <LONG InsertAfterID>)

Arguments:

BandID	ID of the band the control should be added to
ID	The unique ID of this control
nWidth	Optional. Width of this control in pixels. If omitted, the size defaults to *PTB:BTN_RESIZE*.
InsertAfterID	Optional. Control should be inserted after the control with InsertAfter-ID. If you specify 0 as InsertAfterID, the control will be inserted first in band.

5.5.13 AddText Toolbar Methods

Adds a static control with text. This control has no actions.

Prototype:
AddText(LONG BandID, LONG ID, STRING Txt, LONG Flags=0, <LONG InsertAfterID>)

Arguments:

BandID	ID of the band the control should be added to
ID	The unique ID of this control
Txt	The text for this control
Flags	Optional. Style flags. Can be one or more of the following flags: *PTB:BTN_DISABLED* Control is initially disabled *PTB:BTN_HIDE* Control is hidden

InsertAfterID	Optional. Control should be inserted after the control with InsertAfter-ID. If you specify 0 as InsertAfterID, the control will be inserted first in band.

5.5.14 DelayRefresh — Toolbar Methods

Refreshes the toolbar with an option to delay the refresh.

Prototype:
DelayRefresh(BYTE bDelay)

Arguments:

bDelay	If true, the refresh is delayed, if true, the toolbar is refreshed immediately.

Return value:
Returns True if text is visible, False otherwise.

See also:
Refresh

5.5.15 DeleteBand — Toolbar Methods

Delete a band and all it's controls.

Prototype:
DeleteBand(LONG BandID)

Arguments:

BandID	ID of a band

5.5.16 DeleteControl — Toolbar Methods

Delete a control

Prototype:
DeleteControl(LONG ID)

Arguments:

ID	ID of the control to be deleted

5.5.17 DeleteItem Toolbar Methods

Removes a drop item from a DropButton, SplitButton or DropEntry-control.

Prototype:
DeleteItem(LONG ControlID, LONG ItemID)

Arguments:

ControlID	ID of a DropButton, SplitButton or DropEntry
ItemID	ID of the item you want to remove

5.5.18 DeleteItems Toolbar Methods

Removes all drop items or combo items of a DropButton, SplitButton or DropEntry.

Prototype:
DeleteItems(LONG ControlID)

Arguments:

ControlID	ID of a DropButton, SplitButton or DropEntry

5.5.19 ExecuteControl Toolbar Methods

Triggers the action specified for this control

Prototype:
ExecuteControl(LONG ControlID)

Arguments:

ControlID	ID of the control to be executed

Example:
```
 Toolbar1.ExecuteControl(ID_MyButton1)
```

5.5.20 GetActiveTab Toolbar Methods

Returns the pointer to the active tab in the Self.BandQ.ControlQ

Prototype:
GetActiveTab(LONG BandID),LONG

Arguments:

BandID	ID of the control to be found

Return value:
Returns Pointer() to the Self.BandQ.ControlQ if an active tab is found, otherwise it returns zero.

See also
SetActiveTab

5.5.21 GetBandHidden Toolbar Methods

Returns if a band is visible or not and if it's visible in the Customize menu. See also GetBandVisible

Prototype:
GetBandHidden(LONG BandID),BYTE

Arguments:

BandID	ID of the band

Return value:
Returns True if band is hidden.

5.5.22 GetBandIconsAbove Toolbar Methods

Returns if icons is drawn above button text

Prototype:
GetBandIconsAbove(LONG BandID),BYTE

Arguments:

BandID	ID of the band

Return value:
Returns True icons is drawn above button text, False otherwise.

5.5.23 GetBandShowIcons Toolbar Methods

Returns if icons is visible or not in a band

Prototype:
GetBandShowIcons(LONG BandID),BYTE

Arguments:

BandID	ID of the band

Return value:

Returns True if icons is visible, False otherwise.

5.5.24 GetBandShowText Toolbar Methods

Returns if controls text is visible or not in a band

Prototype:

```
GetBandShowText(LONG BandID),BYTE
```

Arguments:

BandID	ID of the band

Return value:

Returns True if text is visible, False otherwise.

5.5.25 GetBandVisible Toolbar Methods

Returns if a band is visible or not.

Prototype:

```
GetBandVisible(LONG BandID),BYTE
```

Arguments:

BandID	ID of the band

Return value:

Returns True if band is visible

5.5.26 GetCheckbox Toolbar Methods

Test if a button is a checkbox

Prototype:

```
GetCheckbox(LONG ControlID),BYTE
```

Arguments:

ControlID	ID of the control

Return value:
Returns True if this control is a checkbox (or radio), false otherwise.

5.5.27 GetChecked Toolbar Methods

Test if a control is checked

Prototype:
GetChecked(LONG ControlID),BYTE

Arguments:

ControlID	ID of the control

Return value:
Returns True if the control is checked, False otherwise.

5.5.28 GetColor Toolbar Methods

Returns the color value for a ColorID

Prototype:
GetColor(LONG ColorId),ULONG,VIRTUAL

Arguments:

ColorID	A colorID, this can be one of the color-equates described in the Equates section

Return value:
None

5.5.29 GetControlBandID Toolbar Methods

Get the parent band ID fora control.

Prototype:
GetControlBandID(LONG ControlID),LONG

Arguments:

ControlID	ID of the control

Return value:

Returns the band ID or 0 if the control wasn't found.

5.5.30 GetDisabledIcon Toolbar Methods

Get the name of the icon for a control

Prototype:

GetDisabledIcon(LONG ControlID),STRING

Arguments:

ControlID	ID of the control

Return value:

Returns a STRING containing the icon name. If no icon has been specified, an empty string is returned.

5.5.31 GetDrawBottomBorder Toolbar Methods

Test if this toolbar has bottom border

Prototype:

GetDrawBottomBorder(),BYTE

Return value:

Returns True if bottom border is switched on.

5.5.32 GetEnabled Toolbar Methods

Check if a control is enabled

Prototype:

GetEnabled(LONG ControlID),BYTE

Arguments:

ControlID	ID of the control

Return value:
Returns True if the control is enabled, and False if it's disabled.

5.5.33 GetEndShadowWidth — Toolbar Methods

Gets the width of the shadow that is drawn at the right edge of bands when the Office 2003 theme is active.

Prototype:
GetEndShadowWidth(),BYTE

Return value:
Returns the width of the shadow. Default value is 8 pixels.

See also:
SetEndShadowWidth

5.5.34 GetFixedHeight — Toolbar Methods

Get the fixed height set for a band

Prototype:
GetFixedHeight(LONG BandID),LONG

Arguments:

BandID	ID of the band

Return value:
Returns the fixed height in pixels. If height isn't fixed, the return value is 0

5.5.35 GetFixedWidth — Toolbar Methods

Get the fixed width set for a band

Prototype:
GetFixedWidth(LONG BandID),LONG

Arguments:

BandID	ID of the band

Return value:
Returns the fixed width in pixels. If width isn't fixed, the return value is 0

5.5.36 GetIcon Toolbar Methods

Get the name of the icon for a control

Prototype:
GetIcon(LONG ControlID),STRING

Arguments:

ControlID	ID of the control

Return value:
Returns a STRING containing the icon name. If no icon has been specified, an empty string is returned.

5.5.37 GetItemID Toolbar Methods

Returns the ID of a DropCombo-item or DropButton-item.

Prototype:
GetItemID(LONG ControlID, <LONG Index>),LONG

Arguments:

ControlID	ID of the control
Index	Optional. Specifies which item you want to get the ID from. If this argument is omitted the currently selected item will be used for DropCombo's. If it's a DropButton, the first item will be returned.

Return value:
Returns a LONG containing the ID. Returns -1 string if Index is out of bounds. If not ID has been set, 1 based index value is returned.

5.5.38 GetItemText Toolbar Methods

Returns the text of a DropCombo-item or DropButton-item.

Prototype:
GetItemText(LONG ControlID, <LONG Index>),STRING

Arguments:

ControlID	ID of the control
Index	Optional. Specifies which item you want to get the text from. If this argument is omitted the currently selected item will be used for DropCombo's. If it's a DropButton, the first item will be returned.

Return value:
Returns a STRING containing the text. Returns empty string if Index is out of bounds.

5.5.39 GetMenuStyle Toolbar Methods

Retrieves the style of the menu in use.

Prototype:
GetMenuStyle(),LONG

Return value:
Returns one of the PTB:STYLE_ style values - see top of potoolbar.inc for definitions of those styles.

See also:
SetMenuStyle

5.5.40 GetPrompt Toolbar Methods

Returns the prompt text of an entry or dropcombo.

Prototype:
GetPrompt(LONG ControlID),STRING

Arguments:

ControlID	ID of the control

Return value:
Returns a STRING containing the prompt for this control.

5.5.41 GetPromptWidth — Toolbar Methods

Returns the prompt width of an entry or dropcombo.

Prototype:
GetPromptWidth(LONG ControlID),LONG

Arguments:

ControlID	ID of the control

Return value:
Returns a LONG containing the prompt width for this control.

5.5.42 GetRadio — Toolbar Methods

Returns if this is a radio control

Prototype:
GetRadio(LONG ControlID),BYTE

Arguments:

ControlID	ID of the control

Return value:
Returns True if this control is a radio-button. False otherwise.

5.5.43 GetReadOnly — Toolbar Methods

Check if a DropEntry or Entry is read only.

Prototype:
GetReadOnly(LONG ControlID),BYTE

Arguments:

ControlID	ID of the control

Return value:
Return True if the DropEntry or Entry is read-only.

5.5.44 GetStyle Toolbar Methods

Returns the current toolbar style.

Prototype:
GetStyle(),LONG

Arguments:

Return value:
One of the following equates:
PTB:STYLE_OFFICE2000
PTB:STYLE_OFFICEXP
PTB:STYLE_OFFICE2003
PTB:STYLE_NATIVEXP

5.5.45 GetText Toolbar Methods

Returns the display text of a control

Prototype:
GetText(LONG ControlID),STRING

Arguments:

ControlID	ID of the control

Return value:
Returns a STRING containing the text of this control.

5.5.46 GetTooltipMode Toolbar Methods

Returns which tooltip mode has been selected

Prototype:
GetTooltipMode(),BYTE

Return value:
Returns 0 for disabled tooltip, 1 for normal, and 2 for balloon style tooltip.

5.5.47 GetVisible — Toolbar Methods

Returns is a control is visible

Prototype:
GetVisible(LONG ControlID),BYTE

Arguments:

ControlID	ID of the control

Return value:
Returns True if control is visible, False otherwise.

5.5.48 InitControl — Toolbar Methods

Initializes the toolbar control with an option to autoresize

Prototype:
InitControl(LONG nControl, BYTE bAutoResize=0)

Arguments:

nControl	Clarion FEQ of the toolbar control to use for the toolbar.
bAutoResize	Specifies if the toolbar should resize automatically.

See also:
Init

5.5.49 InitControl — Toolbar Methods

Initializes the toolbar control. Must be called before adding bands and controls

Prototype:
InitControl(LONG cGradient)

Arguments:

cGradient	FEQ of the Toolbar control.

5.5.50 IsClosing Toolbar Methods

Uncertain

Prototype:
IsClosing(),BYTE

Return value:
Returns the value of SELF.bIsClosing, which is set to true in the subclassing procedure (POToolbarClass:WndProc) when the window is notified about it's destruction.

5.5.51 IsControlInBand Toolbar Methods

Test if a controls is in a particular band.

Prototype:
IsControlInBand(LONG BandID, LONG ControlID),BYTE

Arguments:

BandID	ID of the band
ControlID	ID of the control

Return value:
Returns True if control is in the band, False otherwise.

5.5.52 Refresh Toolbar Methods

Forces redraw of the control.

Prototype:
Refresh(BYTE Recalculate=0)

Arguments:

Recalculate	Optional. If set to True the toolbar will recalculate position and sizes of it self and child bands and controls.

5.5.53 RestoreBandVisible — Toolbar Methods

Restores the visible status of a band from an ini file

Prototype:
RestoreBandVisible(String pINIFileName, String pProcedureName)

Arguments:

pINIFileName	Name of INI file name to restore from
pProcedureName	Name of the procedure where the toolbar is.

See also:
SaveBandVisible

5.5.54 SaveBandVisible — Toolbar Methods

Saves the visible status of a band to an ini file

Prototype:
SaveBandVisible(String pINIFileName, String pProcedureName)

Arguments:

pINIFileName	Name of INI file name to save to
pProcedureName	Name of the procedure where the toolbar is.

See also:
RestoreBandVisible

5.5.55 SelectComboItem — Toolbar Methods

Selects a given Combo Item based on the item text.

Prototype:
```
SelectComboItem        PROCEDURE(LONG ComboID, STRING Txt)
```

Arguments:

ComboID	The ID for the drop down combo control
Txt	The text of the item to select.

See also:
GetItemText

5.5.56 SetActiveTab Toolbar Methods

Triggers the action specified for this control

Prototype:
SetActiveTab(LONG BandID, LONG nActiveTabPos)

Arguments:

BandID	ID of the control to be executed
nActiveTabPos	Active Position to set

See also:
GetActiveTab

5.5.57 SetAllowCustomize Toolbar Methods

Use this method to disable the right-click customize dialog.

Prototype:
SetAllowCustomize(BYTE CustomizeEnabled)

Arguments:

CustomizeEnabled	Set to true if you want to enable customize. Set to false to disable.

5.5.58 SetAppName Toolbar Methods

If you change the name of a dll containing a PowerToolbar control, you must set the new name with this method. See Multi-DLL applications.

Prototype:
SetAppName(STRING szAppName)

Arguments:

szAppName	String containing the name of the dll

See also:
Multi-DLL applications

5.5.59 SetBandControlsEnabled Toolbar Methods

Enable all controls in a band.

Prototype:
SetBandControlsEnabled(LONG BandID, BYTE bEnabled=1)

Arguments:

BandID	ID of a band
bEnabled	Optional. Specify True to enable or False to disable. Defaults to True.

5.5.60 SetBandHidden Toolbar Methods

Hides or unhides a specified band and removes it from the customize menu. See also SetBandVisible

Prototype:
SetBandHidden(LONG BandID, BYTE IsHidden)

Arguments:

BandID	ID of a band
IsHidden	True or false. If True, the band is hidden and removed from the customize menu. If false it is unhidden and added to the customize menu.

.

5.5.61 SetBandIconsAbove Toolbar Methods

Controls whether icons is drawn above button text or not in a band.

Prototype:
SetBandIconsAbove(LONG BandID, BYTE bIconsAbove)

Arguments:

BandID	ID of a band
bIconsAbove	Set to True to display icons above button text in the band.

5.5.62 SetBandShowIcons Toolbar Methods

Controls whether icons is visible or not in a band.

Prototype:
SetBandShowIcons(LONG BandID, BYTE bShowIcons)

Arguments:

BandID	ID of a band
bShowIcons	Set to True to display icons in the band, or False to hide all icons.

5.5.63 SetBandShowText Toolbar Methods

Controls whether text is visible or not in a band.

Prototype:
SetBandShowText(LONG BandID, BYTE bShowText)

Arguments:

BandID	ID of a band
bShowText	Set to True to if you want controls text to display in the band.

5.5.64 SetBandVisible Toolbar Methods

Controls the visibility of a band (and it's child controls) See also SetBandHidden

Prototype:
SetBandVisible(LONG BandID, BYTE IsVisible)

Arguments:

BandID	ID of a band
IsVisible	Set to False to hide band, True to show band.

5.5.65 SetCheckbox — Toolbar Methods

Controls if a control is checkbox or not.

Prototype:
SetCheckbox(LONG ControlID, BYTE IsCheckbox)

Arguments:

ControlID	ID of the control
IsCheckbox	If this argument is True, the button control will be turned into a checkbox

5.5.66 SetChecked — Toolbar Methods

Check or uncheck a checkbox control

Prototype:
SetChecked(LONG ControlID, BYTE bChecked)

Arguments:

ControlID	ID of the control
bChecked	Specify True to set the control checked, False to uncheck.

5.5.67 SetDefaultColors — Toolbar Methods

This method is called to set default colors. Usually overridden by the control template if PowerXP-Theme is present.

Prototype:
SetDefaultColors(),VIRTUAL

5.5.68 SetDisabledIcon — Toolbar Methods

Set disabled icon for a control

Prototype:
SetDisabledIcon(LONG ControlID, STRING IconName)

Arguments:

ControlID	ID of the control
IconName	Name of the icon

5.5.69 SetDrawBottomBorder Toolbar Methods

Prototype:
SetDrawBottomBorder(BYTE bDrawBottomBorder)

Arguments:

bDrawBottomBorder	Set to True to draw a bottom-border for the toolbar.

5.5.70 SetEnabled Toolbar Methods

Disable or enable a control

Prototype:
SetEnabled(LONG ControlID, BYTE bEnabled)

Arguments:

ControlID	ID of the control
bEnabled	True enables the control, False disables.

5.5.71 SetEndShadowWidth Toolbar Methods

Sets the width of the shadow that is drawn at the right edge of bands when the Office 2003 theme is active. To set the width, call this method before the Init method.

Prototype:
SetEndShadowWidth(Byte pEndShadowWidth)

Arguments:

pEndShadowWidth	Width of end shadow of toolbar bands in Office 2003

	theme. Default 8 pixels.

See also:
GetEndShadowWidth

5.5.72 SetFixedHeight Toolbar Methods

Set fixed height for controls in a band

Prototype:
SetFixedHeight(LONG BandID, LONG BandHeight)

Arguments:

BandID	ID of a band
BandHeight	The fixed height for the band (in pixels)

5.5.73 SetFixedWidth Toolbar Methods

Set fixed width of the controls in a band

Prototype:
SetFixedWidth(LONG BandID, LONG BtnWidth)

Arguments:

BandID	ID of a band
BtnWidth	Fixed width for controls in the specified band

5.5.74 SetFocus Toolbar Methods

Sets focus to the specified control

Prototype:
SetFocus(LONG ItemID)

Arguments:

ItemID	Control handle (hwnd) to set focus to.

5.5.75 SetFont Toolbar Methods

Set toolbar font

Prototype:
SetFont(STRING FontName, LONG nSize)

Arguments:

FontName	The name of the font
nSize	Size in pixels (the size is converted to DLU's at runtime)

5.5.76 SetIcon Toolbar Methods

Set icon for a control

Prototype:
SetIcon(LONG ControlID, STRING IconName)

Arguments:

ControlID	ID of the control
IconName	Name of the icon

5.5.77 SetIconHeight Toolbar Methods

Set icon height used for all controls in a band

Prototype:
SetIconHeight(LONG BandID, LONG IconHeight)

Arguments:

BandID	ID of a band
IconHeight	Height in pixels

5.5.78 SetIconWidth Toolbar Methods

Set icon width used for all controls in a band

Prototype:
SetIconWidth(LONG BandID, LONG IconWidth)

Arguments:

BandID	ID of a band
IconWidth	Width in pixels

5.5.79 SetItemEnabled Toolbar Methods

Set item in a Drop Button enabled or disabled.

Prototype:
SetItemEnabled (LONG ControlID, LONG ItemID, BYTE bEnabled)

Arguments:

ControlID	ID of the control
ItemID	ID of the item.
bEnabled	True to enable an item, False to disable an item.

Example:

```
 Toolbar1.SetItemEnabled(ID1_DropButtonTest, 2, False)
```

Disables item number 2.

NOTE: This does currently NOT work correctly IF you set the drop button item numbers manually.

5.5.80 SetMaxComboItems Toolbar Methods

Set maximum number of visible items in a DropCombo droplist.

Prototype:
SetMaxComboItems(LONG ComboID, LONG MaxComboItems)

Arguments:

ControlID	ID of the DropCombo control
MaxComboItems	Maximimum number of visible items in the droplist. Set this value to 0 to set unlimited number of items.

5.5.81 SetMenuStyle Toolbar Methods

Sets the style of the menu.

Prototype:
SetMenuStyle(LONG Style)

Arguments:

Style	One of the PTB:STYLE_ style values - see top of potoolbar.inc for definitions of those styles.

See also:
GetMenuStyle

5.5.82 SetOverflowDetection Toolbar Methods

Controls if bands that would overflow it's row should be moved to the next row.

Prototype:
SetOverflowDetection(BYTE bOverflowDetection)

Arguments:

bOverflowDetection	Set tot True if overflow detection should be active.

5.5.83 SetPrompt Toolbar Methods

Change the prompt of an entry or dropcombo control

Prototype:
SetPrompt(LONG ControlID, STRING Txt)

Arguments:

ControlID	ID of the control
Txt	New prompt for the control

5.5.84 SetPromptWidth Toolbar Methods

Change the prompt width of an entry or dropcombo control

Prototype:
SetPromptWidth(LONG ControlID, LONG nWidth)

Arguments:

ControlID	ID of the control
nWidth	New prompt width. Set to PTB:BTN_RESIZE if you want the size to be equal to the size of the prompt-text.

5.5.85 SetPXP — Toolbar Methods

Used to connect PowerToolbar and PowerXP-Theme.

Prototype:
SetPXP(LONG pPXPGlobal)

Arguments:

pPXPGlobal	Address of the PXPGlobal object

Example:
```
 Toolbar1.SetPXP(Address( PXPGlobal ))
```

5.5.86 SetRadio — Toolbar Methods

Controls whether or not a control is a radiobutton.

Prototype:
SetRadio(LONG ControlID, BYTE IsRadio)

Arguments:

ControlID	ID of the control
IsRadio	Set to True to turn the control into a Radiobutton, False to make it a regular button.

5.5.87 SetReadOnly — Toolbar Methods

Controls read-only for a DropEntry or Entry.

Prototype:
SetReadOnly(LONG ControlID, BYTE bReadOnly)

Arguments:

ControlID	ID of the control
bReadOnly	True sets the control to read-only. False makes it read/write.

5.5.88 SetStyle Toolbar Methods

Change the toolbar style

Prototype:
SetStyle(LONG Style)

Arguments:

Style	Use one of the following equates: PTB:STYLE_OFFICE2000 PTB:STYLE_OFFICEXP PTB:STYLE_OFFICE2003 PTB:STYLE_NATIVEXP

5.5.89 SetText Toolbar Methods

Change the text of a control

Prototype:
SetText(LONG ControlID, STRING Txt)

Arguments:

ControlID	ID of the control
Txt	New text for the control

5.5.90 SetTooltip Toolbar Methods

Set tooltip for a control

Prototype:
SetTooltip(LONG ControlID, STRING Tooltip)

Arguments:

ControlID	ID of the control

Tooltip	String containing the tooltip text

5.5.91 SetTooltipMaxWidth Toolbar Methods

Set maximum width of a tooltip. Default is 300px.

Prototype:
SetTooltipMaxWidth(LONG TooltipMaxWidth)

Arguments:

TooltipMaxWidth	Maximum width of tooltip in pixels

5.5.92 SetTooltipMode Toolbar Methods

Change tooltip mode for the toolbar

Prototype:
SetTooltipMode(BYTE TooltipMode)

Arguments:

TooltipMode	0 - Disabled 1 - Normal 2 - Balloon

5.5.93 SetVisible Toolbar Methods

Set visibility of a control

Prototype:
SetVisible(LONG ControlID, BYTE bVisible)

Arguments:

ControlID	ID of the control
bVisible	If set to True the control is visible. Set to False to hide control.

5.5.94 SetWidth Toolbar Methods

Set fixed width of a control

Prototype:
SetWidth(LONG ControlID, LONG nWidth)

Arguments:

ControlID	ID of the control
nWidth	The new fixed width for the control (in pixels)

5.5.95 UncheckAll Toolbar Methods

Uncheck all checkboxes and radios in a band

Prototype:
UncheckAll(LONG BandID)

Arguments:

BandID	ID of the band

5.6 Global Methods

5.6.1 GetBandHidden — Global Methods

Returns if a band is visible or not and if it's visible in the Customize menu. See also GetBandVisible

Prototype:
GetBandHidden(LONG BandID),BYTE

Arguments:

BandID	ID of the band

Return value:
Returns True if band is hidden.

5.6.2 GetBandVisible — Global Methods

Returns if a band is visible or not.

Prototype:
GetBandVisible(LONG BandID),BYTE

Arguments:

BandID	ID of the band

Return value:
Returns True if band is visible

5.6.3 GetChecked — Global Methods

Test if a control is checked

Prototype:
GetChecked(LONG ControlID),BYTE

Arguments:

ControlID	ID of the control

Return value:
Returns True if the control is checked, False otherwise.

5.6.4 GetEnabled Global Methods

Check if a control is enabled

Prototype:
GetEnabled(LONG ControlID),BYTE

Arguments:

ControlID	ID of the control

Return value:
Returns True if the control is enabled, and False if it's disabled.

5.6.5 GetText Global Methods

Returns the display text of a control

Prototype:
GetText(LONG ControlID),STRING

Arguments:

ControlID	ID of the control

Return value:
Returns a STRING containing the text of this control.

5.6.6 GetVisible Global Methods

Returns is a control is visible

Prototype:
GetVisible(LONG ControlID),BYTE

Arguments:

ControlID	ID of the control

Return value:
Returns True if control is visible, False otherwise.

5.6.7 SetBandControlsEnabled — Global Methods

Enable all controls in a band.

Prototype:
SetBandControlsEnabled(LONG BandID, BYTE bEnabled=1)

Arguments:

BandID	ID of a band
bEnabled	Optional. Specify True to enable or False to disable. Defaults to True.

5.6.8 SetBandHidden — Global Methods

Hides or unhides a specified band and removes it from the customize menu. See also SetBandVisible

Prototype:
SetBandHidden(LONG BandID, BYTE IsHidden)

Arguments:

BandID	ID of a band
IsHidden	True or false. If True, the band is hidden and removed from the customize menu. If false it is unhidden and added to the customize menu.

.

5.6.9 SetBandVisible — Global Methods

Controls the visibility of a band (and it's child controls)

Prototype:
SetBandVisible(LONG BandID, BYTE IsVisible)

Arguments:

BandID	ID of a band
IsVisible	Set to False to hide band, True to show band.

5.6.10 SetChecked Global Methods

Check or uncheck a checkbox control

Prototype:
SetChecked(LONG ControlID, BYTE bChecked)

Arguments:

ControlID	ID of the control
bChecked	Specify True to set the control checked, False to uncheck.

5.6.11 SetEnabled Global Methods

Disable or enable a control

Prototype:
SetEnabled(LONG ControlID, BYTE bEnabled)

Arguments:

ControlID	ID of the control
bEnabled	True enables the control, False disables.

5.6.12 SetText Global Methods

Change the text of a control

Prototype:
SetText(LONG ControlID, STRING Txt)

Arguments:

ControlID	ID of the control
Txt	New text for the control

5.6.13 SetVisible Global Methods

Set visibility of a control

Prototype:

SetVisible(LONG ControlID, BYTE bVisible)

Arguments:

ControlID	ID of the control
bVisible	If set to True the control is visible. Set to False to hide control.

Index