Icetips Outlookbar

10/14/2018

Icetips OutlookBar

Published: October 2018

Publisher

Icetips Creative, Inc.

Managing Editor

Arnor Baldvinsson

Table of Contents

Foreword 0

Part I Introduction 2

1 Welcome ... 2
2 Quickstart ... 3
3 Contact us ... 6

Part II Templates 8

1 About the templates ... 8
2 Adding the templates ... 9
3 Template settings ... 10
- Control Template ... 10
- Header Settings ... 11
- Task Settings ... 12

Part III Misc 16

1 Multi-DLL applications ... 16
2 msigm32.dll ... 17
3 Installed files ... 18
4 Upgrade Issues ... 19
5 License Agreement ... 20
6 Version history ... 25

Part IV Reference 36

1 Embed points ... 36
2 Methods by category ... 38
3 Methods by alphabet ... 40
4 Methods ... 41
- AddHeader ... 41
- AddTask ... 41
- DeleteHeader ... 42
- DeleteTask ... 42
- ExecuteAction ... 43
- GetAlignAllHeaders ... 44
- GetHeaderIcon ... 44
- GetTaskInfo ... 44
- GetTaskTitle ... 45
- GetTaskToolTip ... 45
- GetTaskUserData ... 46
- GetTitle ... 46
- GetTooltipMaxWidth ... 46
- GetWizardMode ... 47

Init 47
LoadExpanded 48
ODS 48
Refresh 48
RGB 48
SaveExpanded 49
SetActiveWizardTask 49
SetAlignAllHeaders 49
SetAppName 50
SetBackgroundColors 50
SetBackgroundGradient 51
SetBorderColors 51
SetColorScheme 51
SetEnabled 52
SetExpanded 52
SetFont 53
SetHeaderBorderColors 53
SetHeaderColors 53
SetHeaderHeight 54
SetHeaderIcon 54
SetHoveredHeaderColors 54
SetIcon 55
SetShowHeaders 55
SetStyle 55
SetTaskColors 56
SetTaskFont 56
SetTaskFontWeight 56
SetTaskInfo 57
SetTaskTitle 57
SetTaskToolTip 57
SetTaskUserData 58
SetTitle 58
SetTooltipMaxWidth 58
SetTooltipMode 59
SetVisible 59
SetWizardMode 59

5 Structures 61

POB:TaskInfo 61

Index 62

Part I

Chapter 1 - Introduction

1 Introduction

1.1 Welcome

Providing easy and intuitive navigation options within an application is the goal of every software developer. Icetips OutlookBar is a native Clarion component that enables you to write applications that has a navigation bar very similar to the famous navigation bar known from applications like Microsoft™ OutlookXP and Outlook 2003.

Features:
- Supports icons in headers and tasks
- OutlookXP and Outlook2003 styles
- "Easy-to-use" Control Template
- Multiple controls per window
- Mimic buttons
- Dynamically create and delete headers and tasks
- Hide/Unhide headers and tasks
- Disable/Enable headers and tasks
- Get and Set header and task info at runtime
- Real, resizable gradients
- Customizable colors
- Multi-DLL support.
- Clarion 5.5 and Clarion 6
- ABC and Legacy
- Full source code (no black-box DLL's)
- Windows 95/98/ME/NT/2000/XP

If you are upgrading from a previous version, please read Upgrade issues.

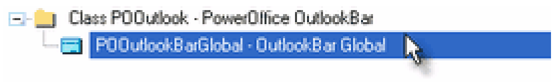
Note that Icetips Creative, Inc. took this product over from PowerOffice in Norway in December 2008. It is possible that there are some references to PowerOffice in this text or in the source codes. If you find references to PowerOffice, please let us know.

1.2 Quickstart

Follow these steps to implement an Icetips OutlookBar in your application:

Add the global template

- Open your application (or create a new one)
- Press the "Global"-options button ( )
- Click the "Extensions"-button
- Click the "Insert"-button
- Choose the "POOutlookBarGlobal - Icetips OutlookBar Global"-template

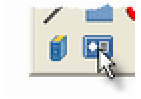


Class POOutlook - PowerOffice OutlookBar
POOutlookBarGlobal - OutlookBar Global

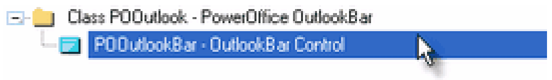
- Press "Ok" two times to get back to the application

Populate the control

- Create a Window
- Populate "OutlookBar" control template by clicking the

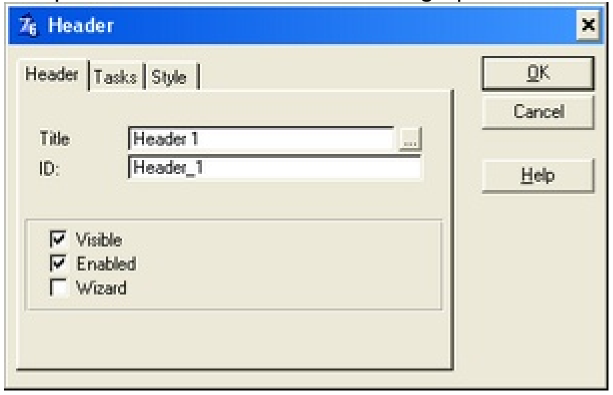


- Choose the "POOutlookBar - Icetips OutlookBar Control"-template



Class POOutlook - PowerOffice OutlookBar
POOutlookBar - OutlookBar Control

- Place the control somewhere on you window and adjust the size.
- Right-click on the control and choose "Actions".
- Now, press the "Insert"-button to bring up the "Header Editor".



Header

Header | Tasks | Style

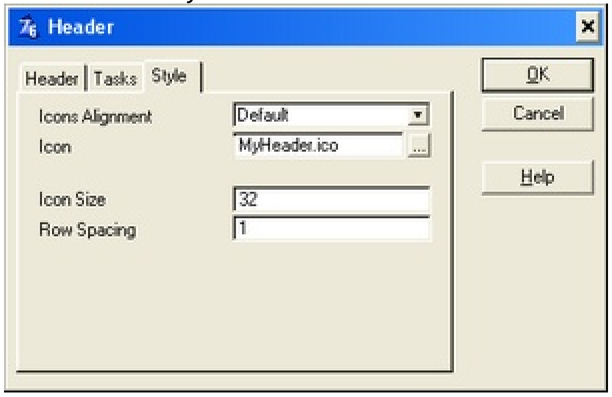
Title: Header 1
ID: Header_1

[✓] Visible
[✓] Enabled
[ ] Wizard

(Header settings)

- In the "Title"-field, enter "Header 1" (without the quotes)

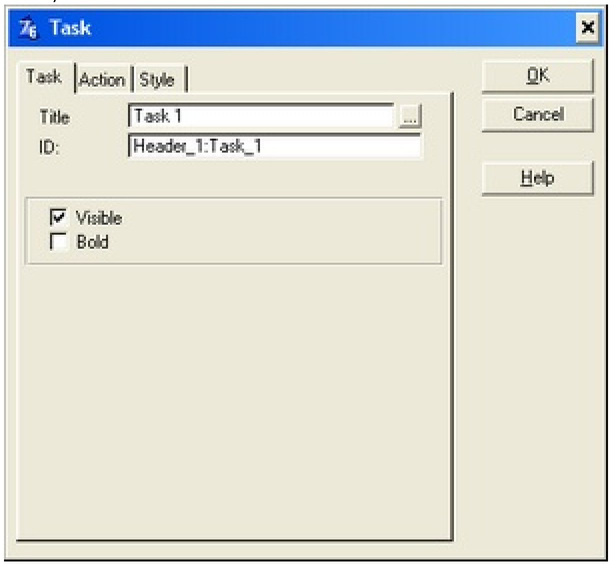
▶ Switch to the "Style tab"



Icons Alignment: Default
Icon: MyHeader.ico
Icon Size: 32
Row Spacing: 1

- If you'd like an icon in the header, press the elipsis-button (...) to right of the icon-field, and choose an .ico-file.

▶ Now, switch to the "Tasks" tab and click "Insert" to add a task.



Title: Task 1
ID: Header_1:Task_1
[✓] Visible
[ ] Bold

(Task settings)

- Enter "Task 1" in the "Title"-field
- For this task, choose "No Special Action" as "Action" on the "Action"-tab
- Click "Ok" to add the Task

▶ Click "Insert" to add another task
- Title: Quit
- Action: Post Event
- Event: EVENT:CloseWindow
- Control:

- Click "Ok" to add the Task

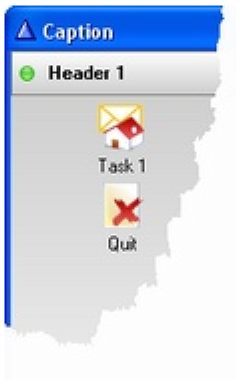
▶ Click "Ok" three times to get back to the window formatter.
- Save the window

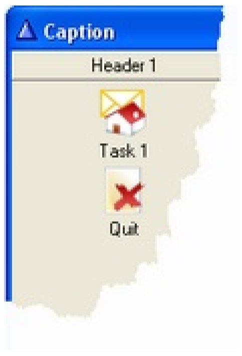
- Now, go to the embed-point "Local Objects → Outlookbar1 → Header_1 → My_First_Header:Task_1 → Before Generated Code", and enter the following code:

```
Message('You selected task 1')
```

- Save, compile and run your application.

The result should look something like this (the left screenshot is with the style set to "Outlook2003", the right screenshot has style set to "OutlookXP"):





When you click on "Task 1" a message saying "You selected task 1" should appear.

1.3 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 Icetips OutlookBar, 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 (OutlookBar) 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

Part II

Chapter 2 - Templates

2 Templates

2.1 About the templates

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

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

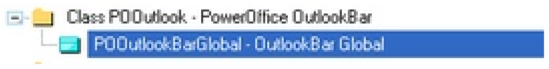
Control Template
The control template generates the actual OutlookBar-control.

Have a look at Adding the templates for more info on how to add the templates.
For a detailed description of the various settings, read Control Template, Header Settings and Task Settings.

2.2 Adding the templates

How to add the global template

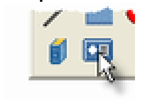
- Open your application (or create a new one)
- Press the "Global"-options button ( )
- Click the "Extensions"-button
- Click the "Insert"-button
- Choose the "POOutlookBarGlobal - Icetips OutlookBar Global"-template


Class POOutlook - PowerOffice OutlookBar
POOutlookBarGlobal - OutlookBar Global

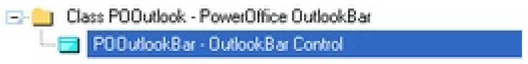
- Press "Ok" two times to get back to the application

Populating an OutlookBar control

- Create a Window
- Populate "OutlookBar" control template by clicking the



- Choose the "POOutlookBar - Icetips OutlookBar Control"-template


Class POOutlook - PowerOffice OutlookBar
POOutlookBar - OutlookBar Control

2.3 Template settings

2.3.1 Control Template — Template settings

Open the template settings:

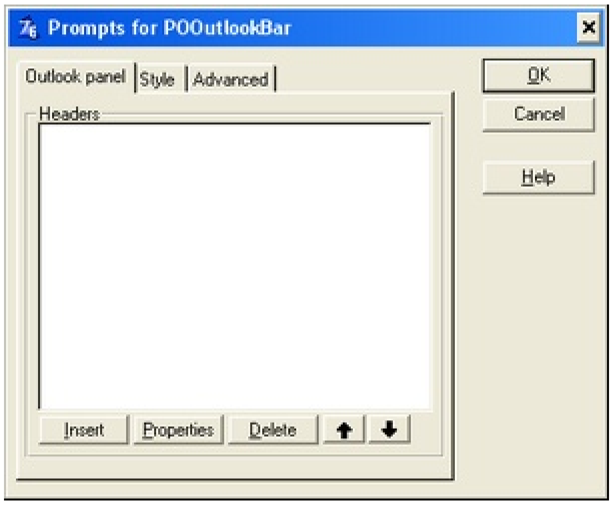
- Right-click the OutlookBar
- Select "Actions"

OutlookBar

Insert
Add new header. This will open the Header Settings

Properties
Change the properties for the selected header.
This will open the Header Settings

Delete
Delete header



Style

OutlookBar style
There are currently two styles. *OutlookXP* which looks like the menu in Outlook XP. *The Outlook 2003* style uses color gradients like the menu in Outlook 2003.

Don't show headers
Hide the header. You must use the SetExpanded method to set which tasks is visible.

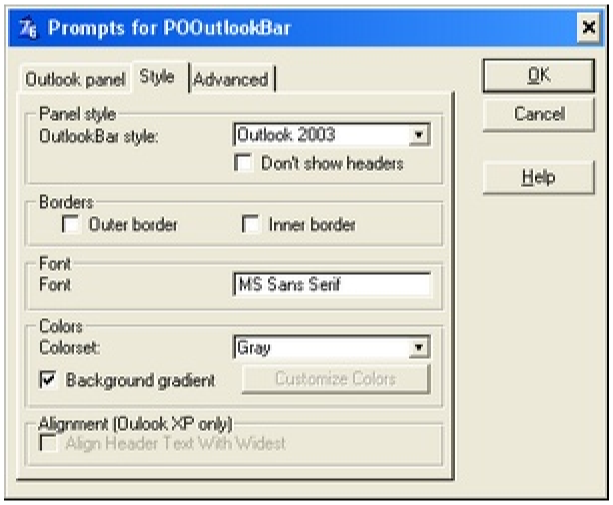
Inner border
Draws a sunken border around the OutlookBar

Outer border
Draws a raised border around the OutlookBar

Font
Enter the name of the font you'd like to use

Colorset
There are two predefined colorsets. Gray and Blue. Set colorset to "Custom" and use the "Customize colors" button to set the colors.

Background gradient



Panel style
OutlookBar style: Outlook 2003
☐ Don't show headers
Borders
☐ Outer border ☐ Inner border
Font
Font: MS Sans Serif
Colors
Colorset: Gray
☑ Background gradient
Alignment (Oulook XP only)
☐ Align Header Text With Widest

If this checkbox is check a gradient will be draw as background for the OutlookBar. If unchecked, plain colors will be used instead.

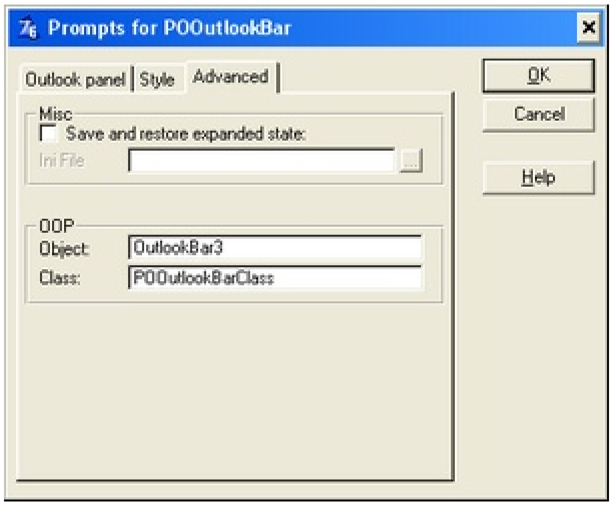
Align Header Text With Widest
If this checkbox is checked the OutlookBar will center the widest header title, and then left align all other header titles with that one.

Advanced

Save and restore expanded state
Check this checkbox and specify a filename (or variable containing the filename. The variable name must be prefixed with '!'). The OutlookBar will now expand the header that was expanded last time the OutlookBar was closed.

Object
Object name used when generating code

Class
Class name used when generating code.



☐ Save and restore expanded state:
Ini File:
Object: OutlookBar3
Class: POOutlookBarClass

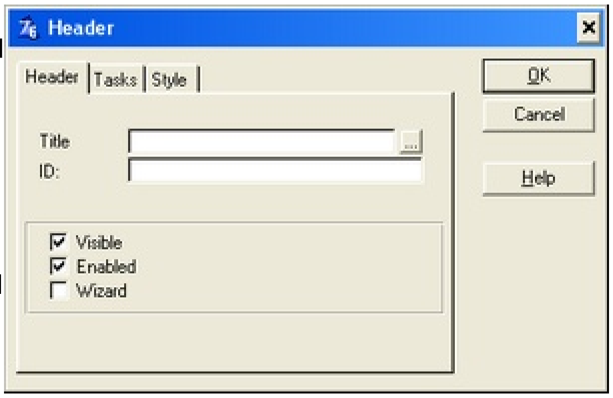
2.3.2 Header Settings — Template settings

Header

Title
This is the text that will be shown on screen. If you would likt to use a variable, just prefix the variable name with '!' (without the quotes)

ID
Each header need a unique identifier. This ID will be used by the template when generating embed-points. If you change the ID after inserting embedded code, this will become orphaned. A LONG-variable in the class will be generated. This LONG will be automatically set runtime, and can be used for calls to class-methods.

Visible
Is this header initially visible upon window open



Title:
ID:
☑ Visible
☑ Enabled
☐ Wizard

Enabled
This header is clickable

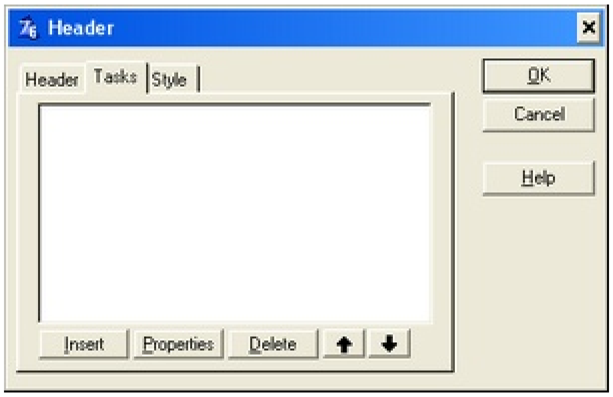
Wizard
Controls if wizard mode should be on (checked) or off. If the wizard is on, and a user clicks a task in this header, the task text will become bold (to indicate which task was last selected).

Tasks

Insert
Add a new task to this header. This will open [Task Settings].

Properties
Change the properties of the selected task. This will open [Task Settings].

Delete
Delete the selected task



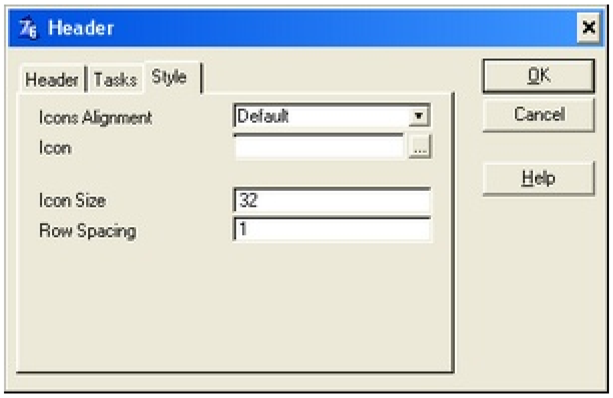
Style

Icons alignment
Controls how task-icons in this header will be aligned. Possible values are *Default*, *Left*, *Right*, *Center.*

Icon
Name of the icon

Icon Size
Size of the icon

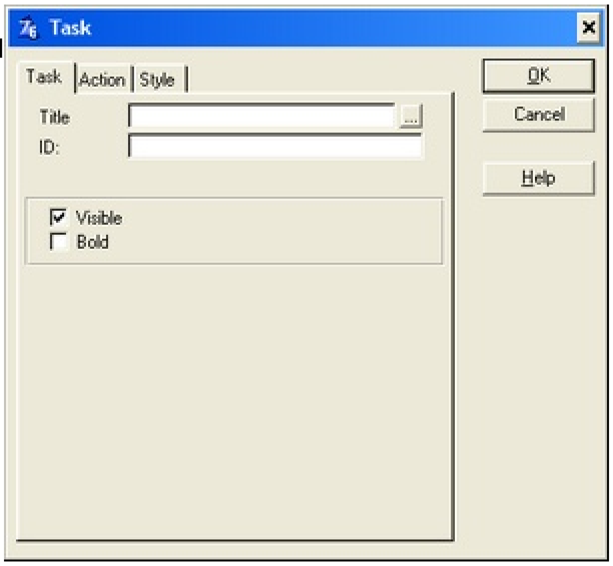
Row Spacing
Vertical spacing between each task



Icons Alignment: Default
Icon:
Icon Size: 32
Row Spacing: 1

2.3.3 Task Settings — Template settings

Task settings



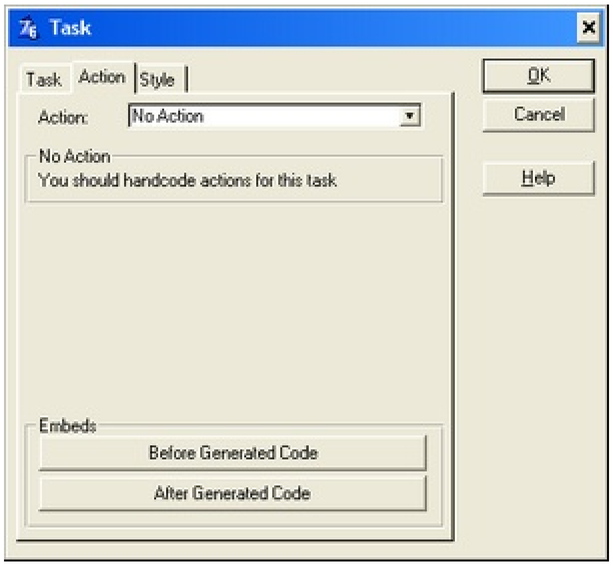
Title
This is the text that will be shown on screen. If you would likt to use a variable, just prefix the variable name with '!' (without the quotes)

ID
Each task need a unique identifier. This ID will be used by the template when generating embed-points. If you change the ID after inserting embedded code, this will become orphaned. A LONG-variable in the class will be generated. This LONG will be automatically set runtime, and can be used for calls to class-methods.

Visible
Controls the visibillity upon window open.

Bold
If checked the task text will be draw bold.

Action



Action
Specifies the type of action for this task. Each type brings up different options:

No Special Action
Do nothing. You should use one (or several) of the embed-points to hand code some action.

Call Procedure
Call a prodecure directly. The settings are just as the standard Clarion "Call Procedure" .

Mimic button
Hides the task when the button specified in "Control" is disabled. If the button is enabled, then the task is visible. If the PROP:Text of the button is changed, this will also be reflected in the taskpanel. Use "mimic button" to add browse-control in your taskpanel (refer the example-application).

When "Mimic Button" is selected, the tasks title will be extracted from the button (using PROP:Text). If "Mimic Button" is not selected, the title specified for the task will be used.

If the "Don't mimic icon" option is checked and the mimiced button has no compile time icon name, Button{PROP:Icon} is used instead. If "Don't mimic icon" is unchecked the icon-name is extracted from the button (STD-icons are not supported). Otherwise the icon specified for the task is used.

Post Event
Posts the specified event. If you specify "Control" the event will be posted to this event. You can also specify

which thread should reveice the event.

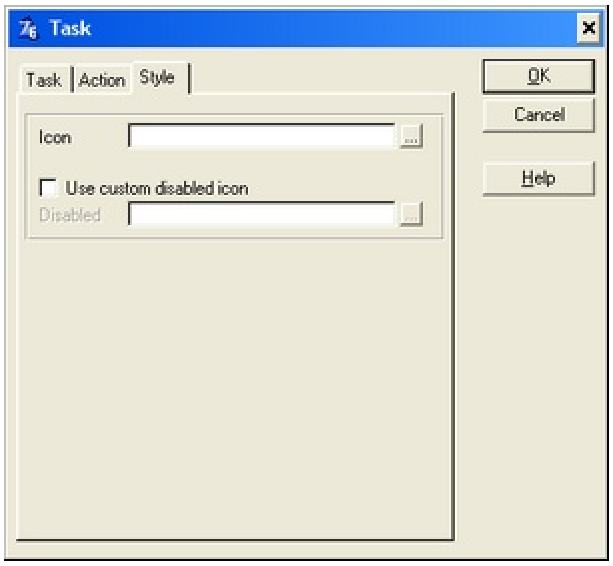
Select Tab
Activate a Tab-control. Just choose the Tab-control you want to activate (bring to front). When this action is executed, EVENT:TabChanging will be posted to the parent Sheet.

Set Field Value
Assign a value to a variable. The Value field could be any valid Clarion expression/assignment.

Style

Icon
Name of the icon. You can use a variable by prefixing the variablename with '!' (wihtout the quotes)

Use custom disabled icon
If you check this option, you can specify an icon to use when the task is disabled. Otherwise a "grayed" version of the normal icon will be used.



Part III

Chapter 3 - Misc

3 Misc

3.1 Multi-DLL applications

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

Filenames NOTE!
DLL's with a OutlookBar control must not be renamed after compilation. If the dll is not found by its compile-time name, in the current dir or path, the OutlookBar will be unable to load the icons from that 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 responisble for setting the correct filename, at runtime, with the method SetAppName.

3.2 msigm32.dll

The API calls for gradients resideds in msimg32.dll.
If this dll is not present on the client system, no gradients will be drawn. Plain colors will be used instead.

msimg32.dll is included by default in Windows 98/2000 and later, and is subject of copyright by Microsoft Corp.

3.3 Installed files

The following files is installed on your system:

- 3rdParty\Libsrc\pooutlookbar.inc
- 3rdParty\Libsrc\pooutlookbar.clw
- 3rdParty\Libsrc\pooutlookbareq.inc
- 3rdParty\Template\pooutlookbar.tpl
- 3rdParty\Docs\OutlookBar\Outloobar.chm
- 3rdParty\Examples\OutlookBar\outlookdemo55.app
- 3rdParty\Examples\OutlookBar\outlookdemo60.app
- 3rdParty\Examples\OutlookBar\flash.ico
- 3rdParty\Examples\OutlookBar\money.ico
- 3rdParty\Examples\OutlookBar\note.ico
- 3rdParty\Examples\OutlookBar\pie-chart.ico
- 3rdParty\Examples\OutlookBar\presentation_chart.ico
- 3rdParty\Examples\OutlookBar\node.ico
- 3rdParty\Examples\OutlookBar\drink_blue.ico
- 3rdParty\Examples\OutlookBar\drink_green.ico
- 3rdParty\Examples\OutlookBar\drink_red.ico
- 3rdParty\Examples\OutlookBar\drink_yellow.ico
- 3rdParty\Examples\OutlookBar\dude2.ico
- 3rdParty\Examples\OutlookBar\dude4.ico
- 3rdParty\Examples\OutlookBar\dude5.ico
- 3rdParty\Examples\OutlookBar\pens.ico
- 3rdParty\Examples\OutlookBar\element.ico
- 3rdParty\Examples\OutlookBar\element_replace.ico
- 3rdParty\Uninstall\OutlookbarUninstall.bat
- 3rdParty\Uninstall\OutlookbarInstall.LOG

For an upto date list of the files being installed and where they are installed, please see:

http://www.icetips.com/productbuilds
[****]

3.4 Upgrade Issues

In this section you'll find various issues that needs to be considered when upgrading from earlier versions.

Upgrading to version 2.0 (Icetips)

All files are installed into the 3rdParty folder or accessory folder for Clarion 7 and newer. Older installs from PowerOffice may have installed files into the Clarion\LibSrc folder in which case you may experience compile errors or weird issues because the files in LibSrc and 3rdParty\LibSrc may be different and incompatible.

Upgrading from version 1.1

One of the AddTask prototypes has been changed to avoid compiler confusion. The prototype that didn't require a header-ID had to be changed. This method does no longer support omitted arugments. All handcode using this method must be changed.

Upgrading from version 1.0

All API-prototypes and some group-definitions has been been prefixed with POB: (PowerOffice OutlookBar) to avoid conflicts with other templates. If you have handcode using any of the below groups you must change it to the new name

Old name	New name
POOutlookColorGrp	POB:ColorGrp
POOutlookTaskInfo	POB:TaskInfo
RECT	POB:RECT

3.5 License Agreement

Icetips "Icetips Outlookbar" 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

3.6 Version history

Version 2018.10.186.115 [October 14, 2018]

July 8, 2016:

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

June 16, 2015:

■ Classes	Resize and loss/gain of focus could cause the Outlookbar control to stop painting after it was erased.

Version 2.0.172 February 24, 2015]

February 24, 2015:

■ Installer	Installer is now compatible with Clarion 10.

July 2, 2014:

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

June 25, 2014:

■ Classes	Two implicit variables in the TakePaintEvent method had been missed when removing implicit variables in September 2012. This could cause problems with the vertical scrollbar not working. Fixed.

Version 2018.10.186.115 [October 14, 2018]

January 28, 2014:

■ Installer	Installer is now compatible with Clarion 9.1

Version 2.0.168.67 [August 21, 2013]

August 21, 2013:

■ Template	Global class was not exported correctly. Fixed.

▪ Template	"Use Global Class for resizing" defaulted to ON. This could cause problems on windows that had Outlookbar controls that weren't a side panel windows and had multiple instances of the Outlookbar control. Fixed. **Note: This must ONLY be checked on an Outlookbar control on a window that should resize to the application frame window size.**
▪ Classes	Subclass testing code was accidentally left in the code. This caused a crash on windows with multiple Outlookbar controls. Code removed. Fixed.
▪ Classes	Icon size for tasks within a header did not reflect the icon size for the header. Fixed

Version 2.0.167.64 [August 14, 2013]

August 13, 2013:

▪ Installer	The POOBWIZ.TPW file was not installed for the older Clarion IDE (Clarion 6.x) only for the new IDE (Clarion 7 and newer) Fixed.

Version 2.0.166 [August 1, 2013]

July 1, 2013:

▪ Template	Added calls to ITRun32C7.dll to show message boxes with information from the wizard. This was later removed, seemed to cause instability in Clarion 8.
▪ Template	No search/replace for WIN32/PASCAL was performed on POOBWIZ.TPW. Fixed.
▪ Classes/Template	Problem with resizing not working when the appframe window opened initially. Fixed.

June 30, 2013:

▪ Install	POOBExports.TPW template file was not included in the install. Fixed.

June 29, 2013:

▪ Release	Released 2.0.160 into beta testing.

June 29, 2013:

▪ Classes	Added 4 new methods, ShellExec, OBShellExec, OpenURL and StartEmail. Those methods add the ability to call ShellExecute directly from the OutlookBar class. OpenURL takes a URL string as parameter that it opens

in the default browser. StartEmail starts and email in the default email client. It takes a email string as parameter that it then adds "mailto:" in front of and starts it in the default email client. This makes it very easy to add buttons to call open websites etc.

- Template/Classes

Outlookbar wizard working and into beta testing. It does NOT import the TXA it generates as the #IMPORT template statement in Clarion 6.3 does not handle it properly. Importing the .txa via "File | Import Text" works without problems. After the wizard generated txa is imported, the AppFrame template needs to be applied to the appframe for control of resizing of the OutlookBar window

June 18, 2013:

- Classes

Added methods to handle saving and restoring of the styles and schemes. Two virtual methods have been added:

```
SaveTheme                 PROCEDURE,VIRTUAL
       !! AB 2013-06-18 - Add virtual method that can
be used in the program
RestoreTheme              PROCEDURE,VIRTUAL
       !! AB 2013-06-18 - Add virtual method that can
be used in the program
```

Code needs to be placed in those methods to save and restore the data. Two variables need to be saved or primed, Style and Scheme.

Two additional methods were also added:

```
GetColorScheme            PROCEDURE(),LONG
       !! AB 2012-12-26:  Added to get current color
scheme
SetColorSchemeAndStyle   PROCEDURE(LONG SchemeID, BYTE
Style)  !! AB 2013-06-18 - combine setting scheme and
color.
```

The first one gets the current color scheme and the second sets both color scheme and style. The schemes and color styles available are:

Schemes:
```
PSTYLE_OUTLOOKXP    EQUATE(0)
PSTYLE_OUTLOOK2003  EQUATE(1)
```

Color styles:
```
PCOLOR_Silver       EQUATE(1)
PCOLOR_Blue         EQUATE(2)
PCOLOR_Gray         EQUATE(3)
```

By default the scheme is set to PSTYLE_OUTLOOK2003. The color style is set to whatever is chosen in the template and passed in through the call to the INIT method. This can now be changed, saved and restored by calling the SaveTheme and RestoreTheme methods. Below is a simple example of how to save and restore the scheme and color in Clarion 8. In SaveTheme the only code added is the IniMgr.Update lines. In RestoreTheme the assignment to Style and Scheme and INIMgr.Fetch are also added before the call to SetColorSchemeAndStyle.

```
OutlookBar1.SaveTheme       Procedure
  Style    Byte
  Scheme   Long
      Code
      Style  = SELF.GetStyle()
      Scheme = SELF.GetColorScheme()
      ! Start of "Save the currrent Theme"
      ! [Priority 5000]
      INIMgr.Update('SidePanel','OBStyle',Style)
      INIMgr.Update('SidePanel','OBScheme',Scheme)

      ! End of "Save the currrent Theme"

OutlookBar1.RestoreTheme    Procedure
  Style    Byte
  Scheme   Long
      Code
      ! Start of "Restore the currrent Theme"
      ! [Priority 4000]
      Style  = PSTYLE_OUTLOOK2003
      Scheme = PCOLOR_Blue
      INIMgr.Fetch('SidePanel','OBStyle',Style)
      INIMgr.Fetch('SidePanel','OBScheme',Scheme)

      ! End of "Restore the currrent Theme"
      SELF.SetColorSchemeAndStyle(Scheme, Style)
```

- Template — On the Styles tab there is now a new checkbox, "Add SaveTheme and RestoreTheme methods" If this is checked, the SaveTheme and RestoreTheme methods will be called. You need to place code in those two methods to actually save the values. It is up to you to provide storage for it, which can be anything, INI file, registry or a database table. See above.

May 20, 2013:

- Template/Classes — Work on wizard template to create an Outlookbar panel window and relate it to the application frame using a global class.

- Classes — Added methods to change header and task fonts:

```
SetHeaderFont             PROCEDURE(String FontName,
Long FontSize, Long FontColor, Long FontStyle)
SetTaskFont               PROCEDURE(String FontName,
Long FontSize, Long FontColor, Long FontStyle)
SetFont                   PROCEDURE(STRING fName)

GetHeaderFont             PROCEDURE(),STRING
SetHeaderFontSize         PROCEDURE(Long FontSize)
GetHeaderFontSize         PROCEDURE(),LONG

SetTaskFontWeight         PROCEDURE(<LONG HeaderID>,
<LONG TaskID>, BYTE bIsBold)
SetTaskFontSize           PROCEDURE(Long FontSize)
GetTaskFontSize           PROCEDURE(),LONG
SetTaskFont               PROCEDURE(STRING FontName)
```

```
GetTaskFont              PROCEDURE(),STRING
```

Some are still undocumented. Some existed prior to this update but the methods above are all about the font settings for the headers and tasks.

March 15, 2013:

- Documentation — Verified that the links in the online manual (http://www.icetips.com/manuals/outlookbar) work. In last upload it did not work.

December 26, 2012:

- Classes — Added GetColorScheme method to get the current color scheme. Added ColorSchema to store the schema value.
- Classes — Added:

```
PCOLOR_Silver        EQUATE(1)
PCOLOR_Blue          EQUATE(2)
PCOLOR_Gray          EQUATE(3)
```

  Equates for color schemes.
- Classes — Added GetStyle method that returns the value of the Style property.

September 12, 2012:

- Classes — Header height was calculated based on each header icon size, font size etc. Now all headers will have the same height, whatever is needed for the maximum size of icons etc. on all the headers. This makes the headers look uniform and avoids issues with headers below the expanded header not filling the entire panel. Height set by SetHeaderHeight() applies to all headers, but if there are headers that need more room than accounted for by the SetHeaderHeight, the height will be increased.

September 11, 2012:

- Classes — Icons in headers were always sized at 16x16 pixels, ignoring settings in template. Fixed.

September 10, 2012:

- Classes — Reformatting code in the .clw file.
  Removed all implicit variables.
  All "." replaced with END.
  All "~=" replaced with "<>"
  All "~" replaced with NOT
  All NOT uppercased.

Changed all checks for Error() to use ErrorCode() instead.
Added space around all arithmetic symbols (i.e. +, -, * and /) to improve code readability.
File headers had old company name. Fixed.
Some equates were still left in the pooutlookbar.inc. Moved to pooutlookbareq.inc.

June 25, 2012:

■ Classes	Code formatting fixed (lot of white space removed).

October 2, 2011:

■ Classes	If icon-height was specified but no icon was specified for a header, the header would still be adjusted to the icon height. Fixed. Use SetHeaderHeight() to set the height for the headers. It can be done at any time. Default height is 32 pixels.

September 16, 2011:

■ Template	Added hot-keys for all template prompts and tabs to make access easier.
■ Classes	Scrollbar height inside a header was not calculated correctly. Hidden task items were still calculated into the total height for the scrollbar. Fixed.

August 24, 2011:

■ Template	Template exported class methods for standalone exe apps. This causes "Unknown directive, ignoring" link errors in Clarion 7 and Clarion 8. Fixed.

June 10, 2011:

■ Classes	Item height for left aligned icons was calculated incorrectly. Fixed.

Version 2.0.139 [May 31, 2011]

May 31, 2011:

■ Installer	Installer suggested "Clarion7" in the examples path. Fixed.
■ Template	%EnableRunTimeTranslator was unknown in Legacy apps. Fixed.

Version 2.0.137 [May 4, 2011]

May 4, 2011:

■ Installer	Installer built for Clarion 8.0.

April 6, 2011:

▪ Template	Embed points for header clicks were not correct causing the code to be "lost" or orphaned. Fixed. Note that this fix ADDS two new embed points for the header and the old embeds are left intact but the name of the old embed point is appended with "DO NOT USE" You should move any code that you see in the "DO NOT USE" embeds to the new embeds using the embeditor.
▪ Template	Added new embed in the CODE section of the ExecuteAction generated method.
▪ Classes	Added new method called GetHeaderHeight. It returns the calculated height of headers in pixels.
▪ Template	Added option to hide/reveal the entire Outlookbar if there is just one header defined. It is only enabled if there is just one header. It will remove the Outlookbar except for the header and make it act kind of like a drop down menu.

February 21, 2011:

▪ Classes	Save and restore expanded state did not work. Fixed.

January 11, 2011:

▪ Template	Added 3 new embed points into "Local Objects | OutlookBar | INIT Method" called "Before CODE", "Beginning of INIT" and "End of INIT" Also moved "Control Initialization | Control Init Code" into "Local Objects | OutlookBar | INIT Method" so all the embeds inside that methods are in this node in the embed tree.
▪ Classes	Fontsize in pixels was not being calculated correctly. Fixed.
▪ Template	Export section was missing methods. Fixed.
▪ Classes	Added methods to set header and task fonts: TaskFontSize, TaskFontName, hTaskFont, SetHeaderFontSize, GetHeaderFontSize, SetTaskFontSize, GetTaskFontSize, , GetHeaderFont, SetTaskFont, GetTaskFont, PointsToPixels. <u>**These are not yet documented.**</u>

To change the fonts for the outlookbar, code like this can be used **before the INIT method is called**:

```
OutlookBar3.SetHeaderFontSize(12)
OutlookBar3.SetTaskFont('Times New Roman')
OutlookBar3.SetTaskFontSize(12)
```

Version 2.0.134 [December 21, 2010]

February 7, 2010:

■ Documentation	Information about the MimicButton option was confusing. Hopefully fixed.

December 16, 2009:

■ Template	Outlookbar template did not support runtime translation. Fixed.

Version 2.0.116 [December 13, 2009]

December 13, 2009:

■ Install	When installing for Clarion 7 no demo app was installed! Fixed.

Version 2.0.113 [December 3, 2009]

December 3, 2009:

■ Classes	ODS method caused problems in Clarion 5.5. Fixed

Version 2.0.112 [September 22, 2009]

September 20, 2009:

■ Classes	Added ODS (OutputDebugString) method to class.
■ Classes	Added SetTitle and GetTitle methods to class to easily set the title of headers and items.

Version 2.0.111 [May 27, 2009]

- Added SetEnable(HeaderID, IsEnabled) method to make it easy to enable/disable all items in a given header.
- Bug tracking set up at http://icetips.fogbugz.com
- Added link to bug tracking to Global template

Version 2.0 [December 15, 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, undocumented methods, such as SetHeaderIcon, GetHeaderIcon, SetTooltipMode, SetTaskToolTip, GetTaskToolTip, SetTooltipMaxWidth and SetTooltipMaxWidth.
- Previous beta changes are all implemented in public release - see below

1.3 BETA 6 - not released publicly before

- Fixed: C55 all modules generate
- Change: Some prototypes has changed from ULONG to LONG to support COLOR:None

- Fixed: GPF in C55 build A-F
- Fixed: non-dropping DropLists
- Fixed: missing template symbol %TmpSaveExpandedIniFile
- New: XP-Theme color support (requires latest PowerXP-Theme)
- New methods: SetHeaderIcon, GetHeaderIcon

1.3 BETA 5 - not released publicly before

- Typo caused Subscript out of range when compiling in debugmode
- Added tooltip-support
- New methods: SetTooltipMode, SetTaskToolTip, GetTaskToolTip, SetTooltipMaxWidth, SetTooltipMaxWidth
- Methods changed: AddTask (new arugment: Tooltip)
- Template now shows header and task ID's in lists

1.3 BETA 4 - not released publicly before

- Mimiced buttons wasn't unsubclassed when calling DeleteTask
- Added flags to avoid WM_LBUTTONUP to fire when closing another window by double-clicking
- Added a call to Update just before calling embed-code

1.3 BETA 3 - not released publicly before

- Scrollbar didn't hide if a header was deleted or hidden
- Scrollbar height wasn't properly calculated
- Position for headers after an expaned header where wrong
- Change: DeleteHeader() and SetVisible(HeaderID,True|False) now automatically calls Refresh()

1.3 BETA 2 - not released publicly before

- Mouse-clicks are now sent to ExecuteAction also when no header or task has been clicked (id's=0)
- Added a new border type
- Changed prototype: Init() now accepts a new optional border argument

1.3 BETA 1 - not released publicly before

- Added right mousebutton support
- Fixed a bug that would reserve space for hidden headers at the bottom of the panel
- Fix bug causing wrong position of scrollbar when changing panel-style runtime
- Changed the basecontrol from Region to Image to remove flicker
- Renamed the Gray-colorset to Silver, and added a new Gray set
- Added option to set header height
- New method: SetHeaderHeight

Version 1.2 [July 9, 2004]

- Added Bold property for tasks
- Added icon aligment
- Added icon size setting
- Added new task action: Set Field Value
- Added real scrollbars
- Added RowSpacing prompt for each header
- Added template buttons to open embed-editor
- Added wizard-mode

- AddHeader prototype has changed
- Changed one of the AddTask prototypes to avoid confusing the compiler
- Fixed bug causing the variable to be added to the project if iconname is variable
- Fixed bug in selection-rectangle calculation when there's no icon
- Fixed bug resulting in GPF on NT4 if style was Outlook-2003
- Fixed bug that could execute a task when a header was clicked and the task came into view
- Fixed bug with ID-generation
- Moved template generated init-code into an Init class method
- New embed-points: Before and After execute-action
- New embed-points: Before and After task logic
- New methods: SetAlignAllHeaders, GetAlignAllHeaders
- New metod: SetTaskFontWeight
- New template option: Align Header Text With Widest
- Optimized mouse handling routines
- Redesigned the template-GUI
- 'Save and restore expanded state' now accepts a variable as filename

Version 1.1 [March 6, 2004]

- Fixed a memory leak happening on the Win9x-platform
- Added missing refresh event on window-maximize
- Added prefix to all prototypes to avoid conflicts with other templates
- Added clippping region to avoid the borders being overdraw by task icons
- Added GetTaskTitle and SetTaskTitle functions
- Added GetTaskUserData and SetTaskUserData functions
- Changed the window subclassing to use a safer way of storing class references

Version 1.0 [April 27, 2004]

- Initial version

Part IV

Chapter 4 - Reference

4 Reference

4.1 Embed points

The OutlookBar control-template generates several embed points. With theese embed-points you can control actions and the behaviour of the control with your own customized sourcecode.

Have a look at the class reference for a list of available functions.

All embed points is generated under "Local Objects-><object name>". For instance "Local Objects->Outlookbar1".

Init embeds

Embed point
`Local Objects->Outlookbar1->Control Initialization->Control Init Code`

This embed-point is executed right before the Outlookbar.Init() metod is called.

Header embeds

Embed points
```
Local Objects->Outlookbar1->Header_1->Header Clicked->Before Generated Code
Local Objects->Outlookbar1->Header_1->Header Clicked->After Generated Code
```

Available properites

`HeaderID`	- The unique ID of the header beeing clicked
`MouseBtn`	- Mouse button that activated the header

Theese embed points are executed everytime a header is clicked. If you want to avoid the header beeing expaned, you can execute a `return` in the "Before Generated Code"-embed.

Task embeds

Embed points
```
Local Objects->Outlookbar1->Header_1->Task_1->Task Clicked->Before Generated Code
Local Objects->Outlookbar1->Header_1->Task_1->Task Clicked->After Generated Code
```

Available properites

`HeaderID`	- The unique ID of the parent header of the task beeing clicked
`TaskID`	- The unique ID of the task beeing clicked
`MouseBtn`	- Mouse button that activated the header

Embeds are executed whenever a task is clicked. You can avoid any action beeing executed by doing a `return` in the "Before Generated Code"-embed.

Execute Action embeds

Embed points:
```
Local Objects->Outlookbar1->Execute Action->Before Generated Code
Local Objects->Outlookbar1->Execute Action->After Generated Code
```

Availale properties:

`HeaderID`	- The unique ID of the parent header of the task beeing clicked
`TaskID`	- The unique ID of the task beeing clicked
`MouseBtn`	- Mouse button that activated the header

Theese embeds are executed every time a task or header is clicked.

Task logic embeds

Embed points:
```
Local Objects->Outlookbar1->Header1->Task Logic->Before Generated Code
Local Objects->Outlookbar1->Header1->Task Logic->After Generated Code
```

Availale properties:

`HeaderID`	- The unique ID of the parent header of the task beeing clicked
`TaskID`	- The unique ID of the task beeing clicked
`MouseBtn`	- Mouse button that activated the header

Embedpoints are called just before and after a task in this header has been clicked.

4.2 Methods by category

POOutlookBarClass contains the following public methods:

Click here for a alphabetized list of functions.

Header functions

- AddHeader
- DeleteHeader
- GetWizardMode
- SetWizardMode
- SetVisible
- SetExpanded

Task functions

- AddTask
- DeleteTask
- GetTaskInfo
- GetTaskTitle
- GetTaskUserData
- SetActiveWizardTask
- SetEnabled
- SetIcon
- SetTaskFontWeight
- SetTaskInfo
- SetTaskTitle
- SetTaskUserData
- SetVisible

Color functions

- SetBackgroundColors
- SetBackgroundGradient
- SetBorderColors
- SetColorScheme
- SetHeaderBorderColors
- SetHeaderColors
- SetHoveredHeaderColors
- RGB

Style functions

- SetFont
- GetAlignAllHeaders
- SetAlignAllHeaders
- SetShowHeaders
- SetStyle
- SetTaskColors

Misc functions

- Init
- ExecuteAction
- LoadExpanded
- SaveExpanded
- Refresh
- SetAppName

Wizard functions

- GetWizardMode

- SetWizardMode
- SetActiveWizardTask

4.3 Methods by alphabet

POOutlookBarClass contains the following public methods:

Click here for a categorized list of functions.

- AddHeader
- AddTask
- DeleteHeader
- DeleteTask
- ExecuteAction
- GetAlignAllHeaders
- GetTaskInfo
- GetTaskTitle
- GetTaskUserData
- GetWizardMode
- Init
- LoadExpanded
- Refresh
- RGB
- SaveExpanded
- SetActiveWizardTask
- SetAlignAllHeaders
- SetAppName
- SetBackgroundColors
- SetBackgroundGradient
- SetBorderColors
- SetColorScheme
- SetEnabled
- SetExpanded
- SetFont
- SetHeaderBorderColors
- SetHeaderColors
- SetHoveredHeaderColors
- SetIcon
- SetShowHeaders
- SetStyle
- SetTaskColors
- SetTaskFontWeight
- SetTaskInfo
- SetTaskTitle
- SetTaskUserData
- SetVisible
- SetWizardMode

4.4 Methods

- Methods by category
- Methods by alphabet

4.4.1 AddHeader Methods

This function appends a header to the OutlookBar. The header will be visible at the next redraw.

Prototype:
AddHeader(STRING Title, BYTE bSmallIcons, <STRING IconName>),LONG,PROC

Arguments:

Title	This is the text that the header will display
bSmallIcons	Set to True if you'd like to use small icons (16x16pixels) instead of the default big ones (32x32pixels)
IconName	Optional. The name of the icon that should be shown in the header. If ommited, no icon is shown. NOTE: The icon MUST be linked into your application.

Return value:
This function returns a LONG, representing an unique ID for this header.

See also:
AddTask, DeleteHeader, DeleteTask

Example:
```
 HID:Header1 = OutlookBar5.AddHeader('OutlookBarb', False, 'project.ico')
```

4.4.2 AddTask Methods

Adds a task to a header. The task will be visible on the next redraw.

Prototypes:
AddTask(STRING Title, LONG ActionFeq, BYTE MimicButton, STRING IconName, BYTE DontMimicIcon),LONG,PROC
AddTask(LONG HeaderID, STRING Title, LONG ActionFeq, <BYTE MimicButton>, <STRING IconName>, <BYTE DontMimicIcon>),LONG,PROC

Arguments:

HeaderID	A previously obtained ID for parent header the task should be added to. If you ommit this value (i.e. use the first prototype) the task will be added to the last added header.
Title	This is the screentext for this task
ActionFeq	The FEQ (Field Equate Label) for the control that should receive an EVENT:Accepted, when this task is clicked. Set this to zero if you don't want the task to activate a control.
MimicButton	Optional. If a button was specifed as the ActionFeq, you could set MimicButton

	to True. The task will then be hidden whenever the button is disabled. When the button is enabled, the task will be visible to the user. (Defaults to false)
IconName	Optional. The name of the icon that should be shown to the left of the task. If ommited, no icon is shown. NOTE: The icon MUST be linked into your application. Also note that the icon name should NOT be prefixed with a tilde (~) even if it is linked in!
DontMimicIcon	Optional, If set to True and MimicButton is specified, the icon set in the template settings will be used instead of the icon of the mimiced button.

Return value:
This function returns a LONG, representing an ID for this task. The ID is only uniqe within its header. Tasks in different headers could have the same ID.

See also:
AddHeader, DeleteHeader, DeleteTask

Example:
```
  OutlookBar5.AddTask('Introduction', ?Button1{PROP:Feq}, False, 'user.ico')
```

4.4.3 DeleteHeader Methods

This function completely removes a header (and all it's tasks) from the OutlookBar.

Prototype:
DeleteHeader(LONG headerID)

Arguments:

HeaderID	The unique ID of the header to remove

See also:
DeleteTask

Example:
```
  OutlookBar5.DeleteHeader(OutlookBar5.Header_1)
```

4.4.4 DeleteTask Methods

This function completely removes a task from the OutlookBar.

Prototype:
DeleteTask(LONG HeaderID, LONG TaskID)

Arguments:

HeaderID	The unique ID of the task's parent header
TaskID	The unique ID of the task to remove

See also:

DeleteHeader

Example:

```
OutlookBar5.DeleteTask(OutlookBar5.Header_1, OutlookBar5.Header_1:Task_1)
```

4.4.5 ExecuteAction Methods

Every class based on the POOutlookBarClass must implement this method.
Whenever a task is clicked, this method will be called, and you use this method to implement actions for your tasks.

Prototype:
ExecuteAction(Long HeaderID, Long TaskID, Long MouseBtn, Long Feq),VIRTUAL

Arguments:

HeaderID	This is the uniqe ID of the header in which the clicked task resides.
TaskID	The ID of the task. This value is only unique within it's header
MouseBtn	Which mousebutten was used to activate the task (currently only MouseLeft is supported)
Feq	The "Field Equate Lable" of the control that should be executed.

Example:

```
OutlookBar5      Class(POOutlookBarClass)
ExecuteAction       Procedure(Long HeaderID, Long TaskID, Long MouseBtn,
Long Feq),VIRTUAL
                End

!! Handle Mouse-clicks on tasks
OutlookBar5.ExecuteAction   Procedure(Long HeaderID, Long TaskID, Long
MouseBtn, Long Feq)
    Code
    If MouseBtn = MouseLeft
        Case HeaderID
        Of HID:XpOutlookBar
            Select(?Sheet1, TaskID)
            Post(EVENT:NewSelection, ?Sheet1)
            Post(EVENT:Accepted, ?Sheet1)

        Of HID:Demos
            Select(?Sheet1, 2+TaskID)
            Post(EVENT:NewSelection, ?Sheet1)
            Post(EVENT:Accepted, ?Sheet1)

        Of HID:HideUnhide
            Message('You selected Task ' & TaskID,'Hide/Unhide
Demo',ICON:Asterisk,'Ok',1)
        End
    End
```

4.4.6 GetAlignAllHeaders — Methods

Retrive info about a task.

Prototype:
GetAllignAllHeaders(),BYTE

Returns
Returns True if headers are set to align with the one with the widest title.

See also:
SetAlignAllHeaders

4.4.7 GetHeaderIcon — Methods

Retrieves the name of the icon used in the specified Header ID.

Prototype:
GetHeaderIcon(LONG HeaderID)

Arguments:

HeaderID	The unique ID of the header in wich the task resides.

Returns
Returns a STRING containing the name of the icon.

See also:
SetHeaderIcon

4.4.8 GetTaskInfo — Methods

Retrive info about a task.

Prototype:
GetTaskInfo(LONG HeaderID, LONG TaskID),POB:TaskInfo

Arguments:

HeaderID	The unique ID of the header in wich the task resides.
TaskID	The unique ID of the task you want info from (if set to 0, header info for HeaderID is returned instead)

Returns
Upon succesfull return, this function will return a POB:TaskInfo-group.

See also:
SetTaskInfo

Example:
```
HeaderInfo           GROUP(POB:TaskInfo),PRE(HIG) .
```

```
TaskInfo              GROUP(POB:TaskInfo),PRE(TIG) .

  Code
  HeaderInfo = Self.GetTaskInfo(HeaderID, 0)
  If HeaderInfo = '' Then Return .

  TaskInfo = Self.GetTaskInfo(HeaderID, TaskID)
  If TaskInfo = '' Then Return .

  Message('Header: ' & HeaderInfo.Title & '<13,10><13,10>' & |
          'Title: ' & TaskInfo.Title & '<13,10>' & |
          'Icon: ' & TaskInfo.IconName & '<13,10>' & |
          'Visible: ' & TaskInfo.IsVisible, 'You clicked a
task',ICON:Asterisk,'Ok',1)
```

4.4.9 GetTaskTitle — Methods

Retrieves the title-text of a task.

Prototype:
GetTaskTitle(LONG HeaderID, LONG TaskID),STRING

Arguments:

HeaderID	The unique ID of the task's parent header
TaskID	The unique ID of the task you want change

Returns
Returns a STRING containing the task's title. If the task was not found, an empty string is returned.

See also:
SetTaskTitle

4.4.10 GetTaskToolTip — Methods

Retrieves the tooltip of a Task.

Prototype:
GetTaskToolTip(LONG HeaderID, LONG TaskID),STRING

Arguments:

HeaderID	The unique ID of the task's parent header
TaskID	The unique ID of the task you want to retrieve

Returns
Returns a STRING containing the task's tooltip. If the task was not found, an empty string is returned.

See also:
SetTaskTooltip

4.4.11 GetTaskUserData Methods

Retrieves the userdata of a task. Userdata is a ULONG variable stored with each task. In this variable you can store whatever value you'd like. For instace an ID of which procedure that should be started (in case of a enduser-defiend dynamic menu), or maybe a reference address of an object or group.

Prototype:
GetTaskUserData(LONG HeaderID, LONG TaskID),ULONG

Arguments:

HeaderID	The unique ID of the task's parent header
TaskID	The unique ID of the task you want change

Returns
Returns an ULONG containing the task's userdata. If the task was not found, this function returns 0.

See also:
SetTaskUserData

4.4.12 GetTitle Methods

Retrive the title of a header or a task.

Prototype:
GetTaskInfo(LONG HeaderID, LONG TaskID),STRING

Arguments:

HeaderID	The unique ID of the header in wich the task resides.
TaskID	The unique ID of the task you want info from (if set to 0, header title for HeaderID is returned instead)

Returns
Upon successful return, this function will return a string with the title of the header or item.

See also:
SetTitle

Example:
```
HeaderTitle

  Code
  HeaderTitle = Self.GetTitle(HeaderID, 0)
```

4.4.13 GetTooltipMaxWidth Methods

Retrieves the maximum width for a tooltip in pixels

Prototype:
GetTooltipMaxWidth(),LONG

Returns
Returns a LONG containing the maximum width of a tooltip in pixels.

See also:
SetTooltipMaxWidth

4.4.14 GetWizardMode Methods

Returns if wizard-mode is enabled or not.

Prototype:
GetWizardMode(LONG HeaderID),BYTE

Arguments

HeaderID	The unique ID of the header you want to check.

Returns:
Returns True if "Wizard mode" is enabled for this header, False otherwise.

See also:
SetWizardMode, SetActiveWizardTask

4.4.15 Init Methods

Call this method to initialize the control. The control will not be draw until this function has been called.
NOTE: Never call this function before the window has been opened (i.e. "Local Objects|Abc Objects|Init|Open Window"-embed is a nice place).

Prototype:
Init(LONG nControl, BYTE Style, <BYTE InnerBorder>, <BYTE OuterBorder>)

Arguments:

nControl	The control to be used.
Style	Specifies the style of the OutlookBar. Currently the following styles are supported: PSTYLE_OUTLOOKXP PSTYLE_OUTLOOK2003
InnerBorder	Set to True to get a inset border around the panel
OuterBorder	Set to True to get a border around the panel

See also
SetBorderColors

Example:
```
OutlookBar5.Init(?OutlookBar, PSTYLE_OUTLOOK2003)
```

4.4.16 LoadExpanded Methods

Load and sets which header that was be expanded when SaveExpanded was called.

Prototype:
LoadExpanded(String sSection, String sIniFile)

Arguments:

sSection	Name of the section you want to use to store this infornation.
sIniFile	The name of the ini-file used to store the information in.

See also:
SaveExpanded, SetExpanded

Example:
OutlookBar5.LoadExpanded('MyTaskpanel','myprogram.ini')

4.4.17 ODS Methods

Sends a string to debug viewer

Prototype:
ODS(String pS)

Arguments:

pS	String to send to debug viewer, such as DebugView from Microsoft/SysInternals (http://technet.microsoft.com/en-us/sysinternals/bb896647.aspx)

Example:
```
Outlookbar1.ODS('Testing')
```

4.4.18 Refresh Methods

Redraws the control

Prototype:
Refresh()

4.4.19 RGB Methods

Creates a ULONG color value based on the R,G,B-values

Prototype:
RGB(Byte R, Byte G, Byte B),ULONG

Arguments:
R - The value for red (0-255)
G - The green value
B - The blue value

Return value:
This function returns a color value that can be used as an argument for the color-functions.

See also:

4.4.20 SaveExpanded Methods

Save which header is currently expanded.

Prototype:
SaveExpanded(String sSection, String sIniFile)

Arguments:

sSection	Name of the section you want to use to store this infornation.
sIniFile	The name of the ini-file used to store the information in.

See also:
LoadExpanded, SetExpanded

Example:
OutlookBar5.SaveExpanded('MyTaskpanel','myprogram.ini')

4.4.21 SetActiveWizardTask Methods

Calling this method will result in the task being drawn bold, and the last selected wizard-task (if any) is set to normal.

Prototypes:
SetActiveWizardTask(LONG HeaderID, LONG TaskID)

Arguments:

HeaderID	A previously obtained header-ID
TaskID	A previously obtained task-ID

See also:
SetWizardMode, GetWizardMode

4.4.22 SetAlignAllHeaders Methods

Control if the OutlookBar will center the widest header title, and then left align all other header titles with that one.

Prototypes:
SetAlighAllHeaders(BYTE AlignAllHeaders)

Arguments:

AlignAllHeaders	If set to *True* the OutlookBar will center the widest header title, and then left align all other header titles with that one.

See also:
GetAlignAllHeaders

4.4.23 SetAppName — Methods

This method is only used with Multi-DLL apps. If you want to load icons from one of your DLL's, the DLL's name must be set with this method. Failing to do so will result in the OutlookBar trying to load icons from the .exe file.

Prototype:
SetAppName(STRING szAppName)

Arguments:

szAppName	The name (casesensitive) of the DLL, including .dll

Example:
```
  OutlookBar5.SetAppName('MyMultiApp.dll')       !Icons is linked into
MyMultiApp.dll
```

4.4.24 SetBackgroundColors — Methods

With this method you can specify the background colors of the OutlookBar.

Prototype:
SetBackgroundColors(ULONG cBackground, ULONG cGradientFrom, ULONG cGradientTo)

Arguments:

cBackground	A plain background color (used when "Background-gradient" is turned off, or the panel style is OutlookXP)
cGradientFrom	The gradient start color
cGradientTo	The gradient end color

See also
RGB, SetBorderColors, SetColorScheme, SetHeaderColors, SetHeaderBorderColors, SetHoveredHeaderColors, SetTaskColors

Example
```
  OutlookBar5.SetBackgroundColors(COLOR:BtnFace, COLOR:White, COLOR:Gray)
```

4.4.25 SetBackgroundGradient — Methods

Use this method to turn on/off the background gradient (only applicable when the OutlookBar-style is Outlook2003).

Prototype:
SetBackgroundGradient(BYTE bUseGradient)

Arguments:

bUseGradient	Set to True to turn gradient on, set to False to turn it off

See also
RGB, SetBackgroundColors

Example
```
OutlookBar5.SetBackgroundGradient(True)
```

4.4.26 SetBorderColors — Methods

With this method you can specify the colors of the panel borders.

Prototype:
SetBorderColors(ULONG cLight, ULONG cDark, ULONG cBackground)

Arguments:

cLight	The light color of the border
cDark	The dark color of the border
cBackground	The background color of the border (when both inner and outer border is turned on)

See also
Init, RGB, SetBackgroundColors, SetColorScheme, SetHeaderColors, SetHeaderBorderColors, SetHoveredHeaderColors, SetTaskColors

Example
```
OutlookBar5.SetBorderColors(COLOR:White, 0000000h, COLOR:BtnFace)
```

4.4.27 SetColorScheme — Methods

Load one of the two predefined color-schemas.

Prototype:
SetColorScheme(Long SchemaID)

Arguments:

SchemaID	The number of the color schema to use. 1 = Silver/Lilac 2 = Blue/Yellow 3 = Gray/Blue

See also
RGB, SetBackgroundColors, SetBorderColors, SetHeaderColors, SetHeaderBorderColors, SetHoveredHeaderColors, SetTaskColors

Example
```
OutlookBar5.SetColorScheme(1)
```

4.4.28 SetEnabled Methods

Enable or disable a task

Prototype:
SetEnabled(BYTE IsVisible)
SetEnabled(LONG HeaderID, BYTE IsVisible)
SetEnabled(LONG HeaderID, LONG TaskID, BYTE IsVisible)

Arguments:

HeaderID	The unique ID of the header you want to change
TaskID	The ID of the task you want to change
IsEnabled	If you set this value to true the item will become enabled. If false, the item will be disabled.

Note:
If you use the first prototype, the last added header or task will be affected. The second affects the header and all items. The third affects only one item.

4.4.29 SetExpanded Methods

Use this method to expand or contract a header.

Prototype:
SetExpanded(LONG HeaderID, <BYTE IsExpanded>)

Arguments:

HeaderID	The unique header ID of the header beeing modified
IsExpanded	(Optional) If true the header will be expanded, false will contract the header. If this argument is omitted the header will be expanded.

See also
SaveExpanded, LoadExpanded

Example:
```
OutlookBar5.SetExpanded(Self.MyFirstHeader, False) !Contracts MyFirstHeader
```

4.4.30 SetFont Methods

Change the current font for the taskpanel. The default font is MS Sans Serif, 8pt.

Prototype:
SetFont(STRING fName)

Arguments:

fName	The name of the new font.

4.4.31 SetHeaderBorderColors Methods

Specify header border-colors.

Prototype:
SetHeaderBorderColors(ULONG cTopLeft, ULONG BottomRight, ULONG BottomRightLight, ULONG BottomRightHovered)

Arguments:

cTopLeft	Color of the top-left border (ligth color)
cBottomRight	Color of the bottom-right border (dark color)
cBottomRightLight	Color of the bottom-right border which is placed one pixel offset of the bottom-right border (a bit lighter then cBottomRight)
cBottomRightHovered	Bottom-right border when mouse is hovering over the header

Note
This method only works when the OutlookBar-style is OutlookXP.

See also
RGB, SetBackgroundColors, SetBorderColors, SetColorScheme, SetHeaderColors, SetHoveredHeaderColors, SetTaskColors

4.4.32 SetHeaderColors Methods

Specify header colors.

Prototype:
SetHeaderColors(ULONG cBackground, ULONG cGradientFrom, ULONG cGradientTo, ULONG cText, ULONG cBorder)

Arguments:

cBackground	Background color of the header (only used when OutlookXP-style)
cGradientFrom	The gradient start color (only for Outlook2003-style)
cGradientTo	The gradient end color (only for Outlook2003-style)
cText	Textcolor
cBorder	The border color (only for Outlook2003-style)

See also
RGB, SetBackgroundColors, SetBorderColors, SetColorScheme, SetHeaderBorderColors, SetHoveredHeaderColors, SetTaskColors

4.4.33 SetHeaderHeight Methods

Sets the header height in pixels. Note that if icons or font settings require some headers to be taller, then all headers will be set to that height. See version history on September 12, 2012

Prototype:
SetHeaderHeight(LONG nHeight)

Arguments:

nHeight	The height of the header in pixels

4.4.34 SetHeaderIcon Methods

Sets the icon to use in the specified header.

Prototype:
SetHeaderIcon(LONG HeaderID, STRING IconName)

Arguments:

HeaderID	The unique ID of the header in wich the task resides
IconName	The name of the icon you want to change to (the icon must be linked into your project)

See also:
GetHeaderIcon

4.4.35 SetHoveredHeaderColors Methods

Same function as SetHeaderColors except that this method specifies the colors when the mouse is hovering over the header.

Prototype:
SetHeaderColors(ULONG cBackground, ULONG cGradientFrom, ULONG cGradientTo, ULONG cText, ULONG cBorder)

Arguments:

cBackground	Background color of the header (only used when OutlookXP-style)
cGradientFrom	The gradient start color (only for Outlook2003-style)
cGradientTo	The gradient end color (only for Outlook2003-style)
cText	Textcolor
cBorder	The border color (only for Outlook2003-style)

See also
RGB, SetBackgroundColors, SetBorderColors, SetColorScheme, SetHeaderBorderColors, SetHeaderColors, SetTaskColors

4.4.36 SetIcon Methods

Change the icon of a task. You can also use this method to an icon to use when the task is disabled.

Prototype:
SetIcon Procedure(<STRING IconName>, <STRING DisabledIcon>)
SetIcon Procedure(LONG HeaderID, LONG TaskID, <STRING IconName>, <STRING DisabledIcon>)

Arguments:

HeaderID	The unique ID of the header in wich the task resides
TaskID	The uniduq ID of the task you want info from
IconName	The name of the icon you want to change to (the icon must be linked into your project)
DisabledIcon	If you'd like to set an icon to use when the task is disabled, specify the name as this argument.

Example
```
OutlookBar5.SetIcon(,'MyDisabledIcon.ico')
```

4.4.37 SetShowHeaders Methods

You can use this method to turn of headers. If you do so, you must use the SetExpanded method to expand the wanted header (when more than one is applicable).

Prototype:
SetShowHeaders Procedure(BYTE ShowHeaders)

Arguments:

ShowHeaders	Set to False to turn of headers

See also
SetExpanded

Example
```
OutlookBar5.SetShowHeaders(False)
```

4.4.38 SetStyle Methods

With this method you can change the OutlookBar style at runtime.

Prototype:
SetStyle(BYTE Style)

Arguments:

Style	Specifies the style of the OutlookBar. Currently the following styles are supported: PSTYLE_OUTLOOKXP PSTYLE_OUTLOOK2003

See also
Init, SetColorScheme

Example:
```
  OutlookBar5.SetStyle(PSTYLE_OUTLOOK2003)
```

4.4.39 SetTaskColors Methods

Specify task colors.

Prototype:
SetTaskColors(ULONG cText, ULONG cHoveredText, ULONG cDisabledText, ULONG cSelection, ULONG cSelectionBorder)

Arguments:

cText	Textcolor
cHoveredText	Textcolor when mouse is hovering over the task
cDisabledText	Textcolor when task is disabled
cSelection	Selection rectangle background color
cSelectionBorder	Selection rectangle border color

See also
RGB, SetBackgroundColors, SetBorderColors, SetColorScheme, SetHeaderBorderColors, SetHeaderColors, SetHoveredHeaderColors

4.4.40 SetTaskFont Methods

Change the current font for the tasks in the Outlookbar . The default font is the font used for the Outlookbar control. The default font is MS Sans Serif, 8pt.

Prototype:
SetTaskFont(STRING FontName)

Arguments:

FontName	The name of the new font.

4.4.41 SetTaskFontWeight Methods

With this method you can set a task to be draw with bold text.

Prototypes:
SetTaskFontWeight(<LONG HeaderID>, <LONG TaskID>, BYTE bIsBold)

Arguments:

HeaderID	A previously obtained ID for parent header the task should be added to. If you ommit this value the justification will be set for the last added task .
TaskID	The unique ID of the task you want to set justification for. If you ommit this value the justification will be set for the last added task .

bIsBold	If this value is True, the task will be drawn with bold text. If False, the task will be drawn as normal.

Example:
```
  OutlookBar1.SetTaskFontWeight(OutlookBar1.Header1, OutlookBar1.Header1_Task1,
TRUE)
```

4.4.42 SetTaskInfo — Methods

Change the properties of a task at runtime

Prototype:
SetTaskInfo(LONG HeaderID, LONG TaskID, POB:TaskInfo TaskInfo)

Arguments:

HeaderID	The unique ID of the header in wich the task resides
TaskID	The uniduq ID of the task you want info from
TaskInfo	A POB:TaskInfo-group with the task-properties

See also:
GetTaskInfo

4.4.43 SetTaskTitle — Methods

Change the title-text of a task.

Prototype:
SetTaskTitle(LONG HeaderID, LONG TaskID, STRING Title)

Arguments:

HeaderID	The unique ID of the task's parent header
TaskID	The unique ID of the task you want change
Title	New task title

See also:
GetTaskTitle

4.4.44 SetTaskToolTip — Methods

Sets the tooltip text to use for the specified header and task.

Prototype:
SetTaskToolTip(LONG HeaderID, LONG TaskID, STRING ToolTip)

Arguments:

HeaderID	The unique ID of the task's parent header
TaskID	The unique ID of the task you want to retrieve
ToolTip	The tooltip string

See also:
SetTaskTooltip

4.4.45 SetTaskUserData Methods

Sets the userdata of a task. Userdata is a ULONG variable stored with each task. In this variable you can store whatever value you'd like. For instace an ID of which procedure that should be started (in case of a enduser-defiend dynamic menu), or maybe a reference address of an object or group.

Prototype:
SetTaskUserData(LONG HeaderID, LONG TaskID, ULONG UserData)

Arguments:

HeaderID	The unique ID of the task's parent header
TaskID	The unique ID of the task you want change
UserData	Any ulong value

See also:
GetTaskUserData

4.4.46 SetTitle Methods

Change the title of a header or task at runtime.

Prototype:
SetTitle(LONG HeaderID, LONG TaskID, STRING pTitle)

Arguments:

HeaderID	The unique ID of the header in which the task resides
TaskID	The uniduq ID of the task you want to change. When changing a header only, set this to 0 (zero)
pTitle	The new text for the header or task.

See also:
GetTitle

Example:
```
OutlookBar1.SetTitle(OutlookBar1.Outlookpanel,0,'Changed Outlookbar Title')
```

4.4.47 SetTooltipMaxWidth Methods

Sets the maximum width of a tooltip in pixels. By default it is set to 300 pixels.

Prototype:
SetTooltipMaxWidth(LONG TooltipMaxWidth)

Arguments:

TooltipMaxWidth	The maximum width of a tooltip in pixels.

See also:
GetTooltipMaxWidth

4.4.48 SetTooltipMode Methods

Sets the tooltip mode as disabled, normal or balloon.

Prototype:
SetTooltipMode(BYTE TooltipMode)

Arguments:

TooltipMode	Specifies tooltip mode. 0 is disabled, 1 is normal tooltips and 2 is balloon tooltips.

See also:

4.4.49 SetVisible Methods

Changes the visibillity of a header or task.

Prototype:
SetVisible(BYTE IsVisible)
SetVisible(LONG HeaderID, BYTE IsVisible)
SetVisible(LONG HeaderID, LONG TaskID, BYTE IsVisible)

Arguments:

HeaderID	The unique ID of the header you want to change
TaskID	The ID of the task you want to change
IsVisible	If you set this value to true the item will become visible. If false, the item is hidden.

Note:
If you use the first prototype, the last added header or task will be affected.

4.4.50 SetWizardMode Methods

Switches Wizard-mode on or off for a header.

If Wizard-mode is enabled, the last clicked task is bolded, to indicate the current step in the wizard.

Prototype:
SetWizardMode(LONG HeaderID, BYTE WizardMode)

Arguments:

HeaderID	Unique ID of the header you want to change
WizardMode	If this is set to True, wizard-mode is enabled, otherwise wizard-mode is turned off.

See also:
GetWizardMode, SetActiveWizardTask

4.5 Structures

4.5.1 POB:TaskInfo Structures

```
POB:TaskInfo        GROUP,TYPE
Id                          LONG
Title                       CSTRING(255)
IsVisible                   BYTE
IsEnabled                   BYTE
IconName                    CSTRING(255)
SmallIcons                  BYTE
LeftActionFeq               LONG
                    END
```

Index

- A -

Action 12
ActionFeq 41
Add new header 10
AddHeader 41
Adding the templates 9
AddTask 41
Advanced 10
Align Header Text With Widest 10
AlignAllHeaders 49

- B -

B 48
Background gradient 10
bIsBold 56
blue 48
Bold 12
bSmallIcons 41
bUseGradient 51

- C -

Call Procedure 12
cBackground 50, 51, 53, 54
cBorder 53, 54
cBottomRight 53
cBottomRightLight 53
cDark 51
cDisabledText 56
cGradientFrom 50, 53, 54
cGradientTo 50, 53, 54
cHoveredText 56
Class 10
cLight 51
Colorset 10
Control Template 8
cSelection 56
cSelectionBorder 56
cText 53, 54, 56
cTopLeft 53

- D -

Delete header 10
Delete the selected task 11
DeleteHeader 42
DeleteTask 42
DisabledIcon 55
Don't show headers 10
DontMimicIcon 41

- E -

Embed points 36
Enabled 11
Execute Action embeds 36
ExecuteAction 43

- F -

Feq 43
fName 53
Font 10

- G -

G 48
GetAlignAllHeaders 44
GetHeaderIcon 44
GetTaskInfo 44
GetTaskTitle 45
GetTaskToolTip 45
GetTaskUserData 46
GetTitle 46, 58
GetTooltipMaxWidth 46
GetWizardMode 47
Global Template 8
green 48

- H -

Header 11
Header embeds 36
Header Settings 10
HeaderID 41, 42, 43, 44, 45, 46, 49, 52, 54, 55, 56, 57, 58, 59

How to add the global template 9

- I -

Icon 11
Icon Size 11
IconName 41, 54, 55, 61
Icons alignment 11
Id 11, 12, 61
Init 47
Init embeds 36
Inner border 10
InnerBorder 47
Installed files 18
IsEnabled 52, 61
IsExpanded 52
IsVisible 59, 61

- L -

LeftActionFeq 61
License Agreement 20
LoadExpanded 48

- M -

Methods by alphabet 40
Methods by category 38
Mimic button 12
MimicButton 41
MouseBtn 43
msigm32.dll 17
Multi-DLL applications 16

- N -

nControl 47
nHeight 54
No Special Action 12

- O -

Object 10
ODS 48
Outer border 10
OuterBorder 47
Outlook 2003 10
OutlookBar style 10
OutlookXP 10

- P -

POB:TaskInfo 61
Populating an OutlookBar control 9
Post Event 12
pS 48
PSTYLE_OUTLOOK2003 55
PSTYLE_OUTLOOKXP 55
pTitle 58

- R -

R 48
red 48
Refresh 48
RGB 48
Row Spacing 11

- S -

Save and restore expanded state 10
SaveExpanded 49
SchemaID 51
Select Tab 12
Set Field Value 12
SetActiveWizardTask 49
SetAlighAllHeaders 49
SetAppName 50
SetBackgroundColors 50
SetBackgroundGradient 51
SetBorderColors 51
SetColorScheme 51
SetEnabled 52
SetExpanded 52
SetFont 53
SetHeaderBorderColors 53
SetHeaderColors 53, 54
SetHeaderHeight 54
SetHeaderIcon 54
SetIcon 55
SetShowHeaders 55
SetStyle 55

SetTaskColors 56
SetTaskFontWeight 56
SetTaskInfo 57
SetTaskTitle 57
SetTaskToolTip 57
SetTaskUserData 58
SetTitle 58
SetTooltipMaxWidth 58
SetTooltipMode 59
SetVisible 59
SetWizardMode 59
ShowHeaders 55
sIniFile 48, 49
SmallIcons 61
sSection 48, 49
Style 11, 47, 55
szAppName 50

- T -

Task embeds 36
Task logic embeds 36
Task settings 11, 12
TaskID 42, 43, 44, 45, 46, 49, 52, 55, 56, 57, 58, 59
TaskInfo 57
Tasks 11
Title 11, 12, 41, 57, 61
ToolTip 57
TooltipMaxWidth 58
TooltipMode 59

- U -

Upgrade Issues 19
Upgrading from version 1.0 19
Upgrading from version 1.1 19
Upgrading to version 2.0 (Icetips) 19
UserData 58

- V -

Version history 54
Visible 11, 12

- W -

Wizard 11
WizardMode 59