Icetips Magic Buttons 1.100

Add a touch of Magic to your applications

Version 1.100

Table of contents

Welcome

Welcome to the Icetips Magic Buttons. This product is the resuls of our work on our Icetips Professional Wizards, where we developed this technique to change the standard look and feel of the buttons. We decided to take this to a bit more professional level and you can see the results in this sharp product, which we call the Icetips Magic Buttons!

Introduction

The Icetips Magic Buttons consist of templates and classes. The template generate all the code necessary to access and use the classes. This allows you to put images behind the buttons in your application and create a very sharp and professional look and feel. There are no black boxes to worry about when using the Magic Buttons, as all the code that makes the Magic work is generated by the template and the supplied classes.

There is a Global Extension template, which sets up the default image for buttons. There is also a Procedure Extension template which allows you to set the image for the buttons, include and exclude buttons as well as override settings for individual buttons if needed. There are also a few code templates that allow you more control in really cool ways, if you want to be very specific and picky with your button images. At the end of this document there is also a comprehensive documentation of the classes and how they can be used in handcoded applications.

Demo Applications

We supply two simple demo applications with the Icetips Magic Buttons. The applications, dictionary and all related image and data files are placed in the 3rdParty\Examples\ITMagicButtons directory.

MagicButtons.app is a ABC application written in Clarion5B. The other is mlegacy.app which is a very simply Clarion application, also written in Clarion 5B. The ABC is also supplied as a Clarion 5B locally compiled EXE in the Examples\ITMagicButtons directory. The ABC application demonstrates use of the extension templates, use of the code templates and provides a nice view utility for the images that we supply with it.

Installation

The Icetips Magic Buttons are provided as a single ITMagicButtons.exe install set. It will install the template files and class files and any additional images that we may provide with it into a single directory of your choice and update the redirection file (RED file) of your Clarion IDE with the path to the images. The images we provide are BMP and ICO images and the *.ico and *.bmp= lines in your redirection file will possibly be modified by our post-install program, as well as *.clw and *.inc entries.

The installation will prompt you to register the templates. If you do not want to do that at installation time, you can always do it later. The templates are located in 3rdParty\Template directory. The template files installed are:

```
ITMagicButtons.tpl          - ABC templates
ITMagicButtonsC.tpl         - Clarion templates
```

There is a number of icons and image files installed both in the 3rdParty\Images\ITMagicButtons directory and the 3rdParty\Examples\ITMagicButtons

There are two simple demo applications supplied in the 3rdParty\Examples\ITMagicButtons directory.

Documentation is provided as Acrobat PDF file in 3rdParty\Docs\ITMagicButtons.

A Post-Installer utility is included in 3rdParty\Tools\ITInstall which makes certain that paths are set correctly in the RED file etc.

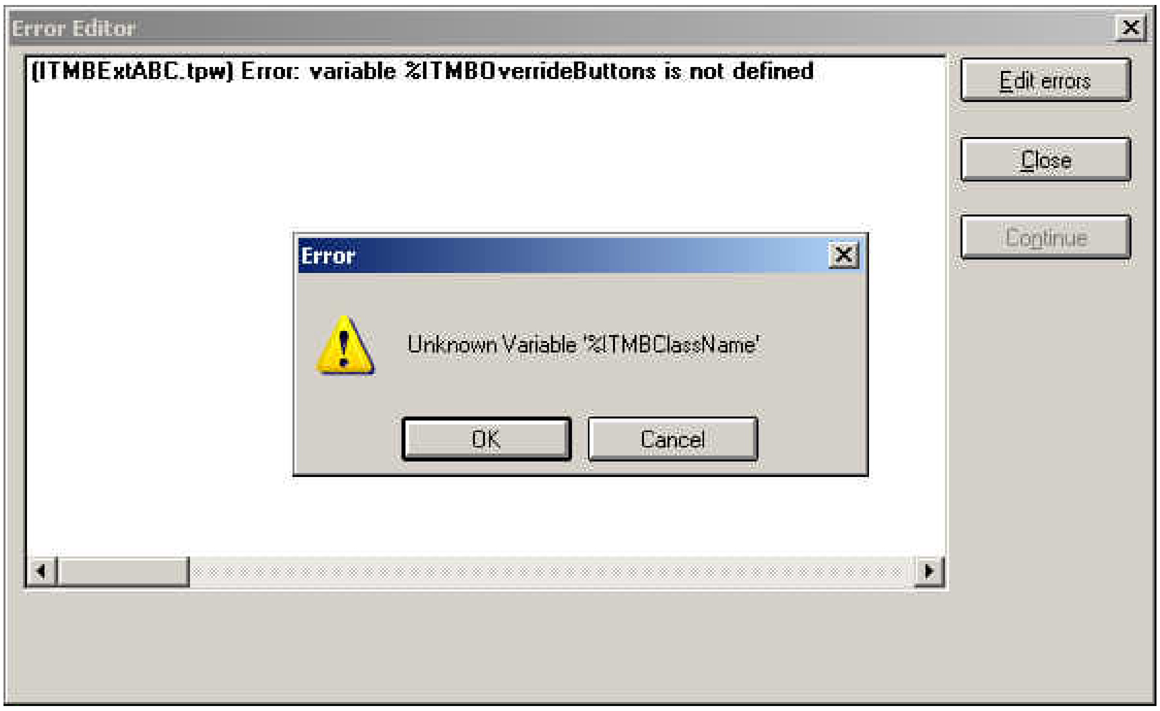
At the end of this document is a complete list of the installed files.

Upgrading from version 1.000 to 1.100

With version 1.1, released on August 9th, 2002, there are some very drastic changes in the way the Magic Buttons work. In the previous release, templates generated all the code. In this release, the templates only generate calls to the MagicButtonsClass methods and set related properties. This reduced the generated code by about 85% and simplifies any handcoding that might need to be done in a Magic Button application.

Applications with the earlier template set should work without problems. However, if the code templates have been used, they may throw errors when generating. It seems that the Clarion IDE still holds on to the old template symbols and get's confused. The new template was applied to several very big projects (upto 15 apps) which had the old templates without any problems at all. If you have applied any of the code templates, you will need to do some work to remove them and reapply them. Unfortunately we have not found any way to make this transition fully automatic.

One problem that we ran into was when we had used the Refresh Buttons template, it would complain about one symbol not being defined. The fix was to remove the template where it had been applied and re-apply it.



```
(ITMBExtABC.tpw) Error: variable %ITMBOverrideButtons is not defined
```

```
Unknown Variable '%ITMBClassName'
```

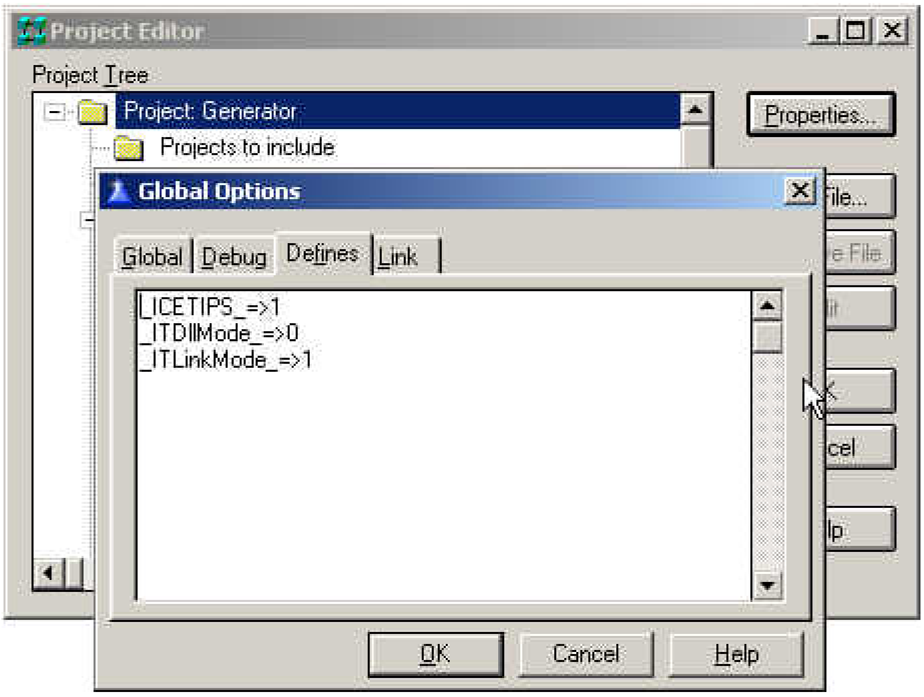
Error when applying the new templates to an app with the old version

Unfortunately the Generator does not tell you where the template has been used. Also refer to the next section for information about defines needed in version 1.1.

Adding the Magic Buttons for the first time

The Magic Buttons template need 3 defines to be able to determine what kind of target file is being compiled, exe or dll. This information is used for the Magic Button class when it it linked into the project. These 3 defines are:

```
_ICETIPS_=>1
_ITDllMode_=>0
_ITLinkMode_=>1
```



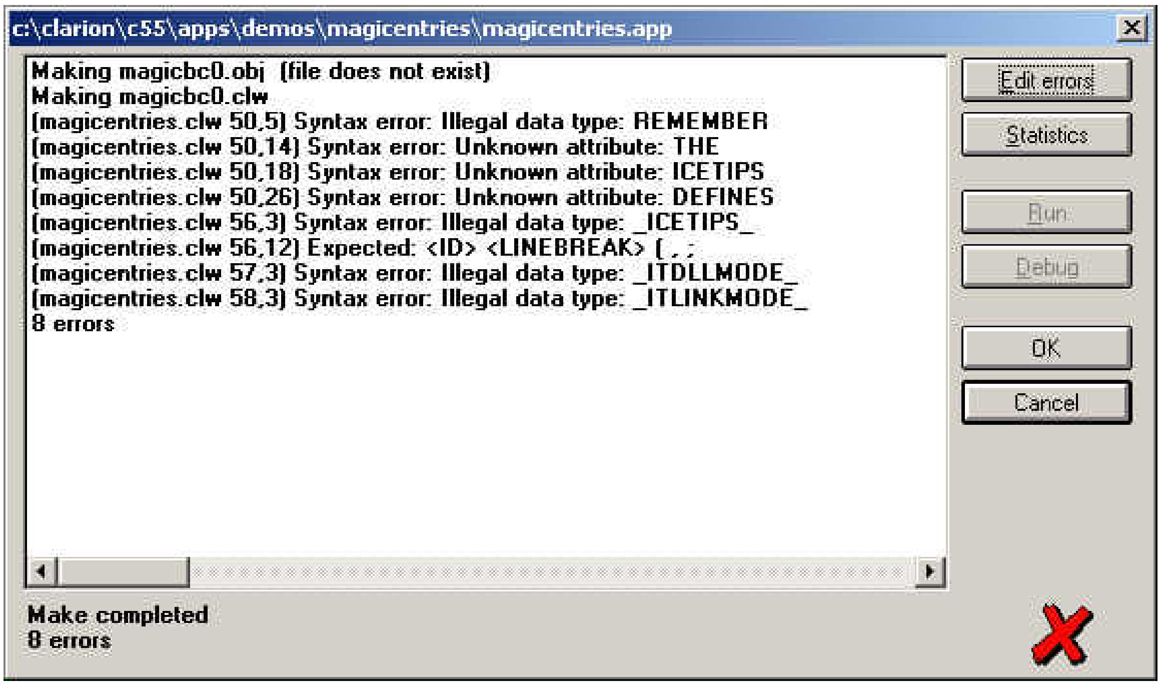
```
_ICETIPS_=>1
_ITDllMode_=>0
_ITLinkMode_=>1
```

Adding Icetips Defines to the Project

To add these defines, you must open your application, then select "Project|Edit" from the main menu in the Clarion IDE. Click on the "Properties" button and then on the "Defines" tab.

You only need to do this the first time you compile your application after you apply the Icetips Magic Button global extension template. This is also necessary to do for each application using the Magic Buttons after you have upgraded from version 1.0.

What happens is that if you forget to add these defines is that the templates will generate code that will not compile:

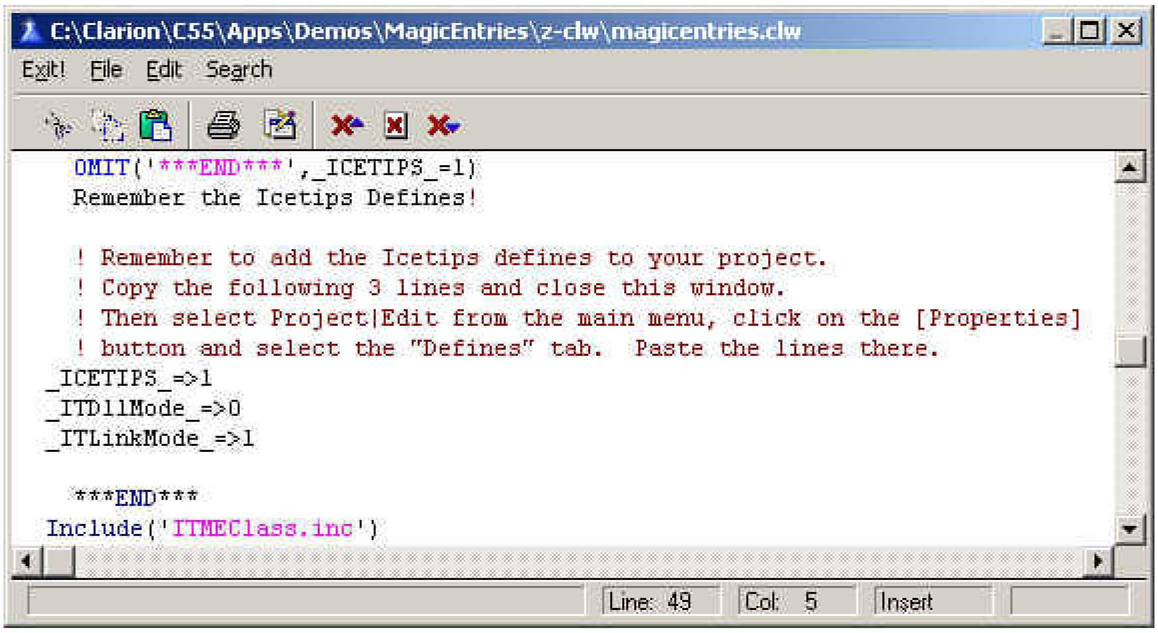


```
Making magicbc0.obj  (file does not exist)
Making magicbc0.clw
(magicentries.clw 50,5) Syntax error: Illegal data type: REMEMBER
(magicentries.clw 50,14) Syntax error: Unknown attribute: THE
(magicentries.clw 50,18) Syntax error: Unknown attribute: ICETIPS
(magicentries.clw 50,26) Syntax error: Unknown attribute: DEFINES
(magicentries.clw 56,3) Syntax error: Illegal data type: _ICETIPS_
(magicentries.clw 56,12) Expected: <ID> <LINEBREAK> ( , ;
(magicentries.clw 57,3) Syntax error: Illegal data type: _ITDLLMODE_
(magicentries.clw 58,3) Syntax error: Illegal data type: _ITLINKMODE_
8 errors
```

```
Make completed
8 errors
```

Compile errors because defines have not been added to the project

If you click on the [Edit error] button, you will see this, reminding you what is missing:


C:\Clarion\C55\Apps\Demos\MagicEntries\z-clw\magicentries.clw
Exit!
File
Edit
Search
Line: 49
Col: 5
Insert

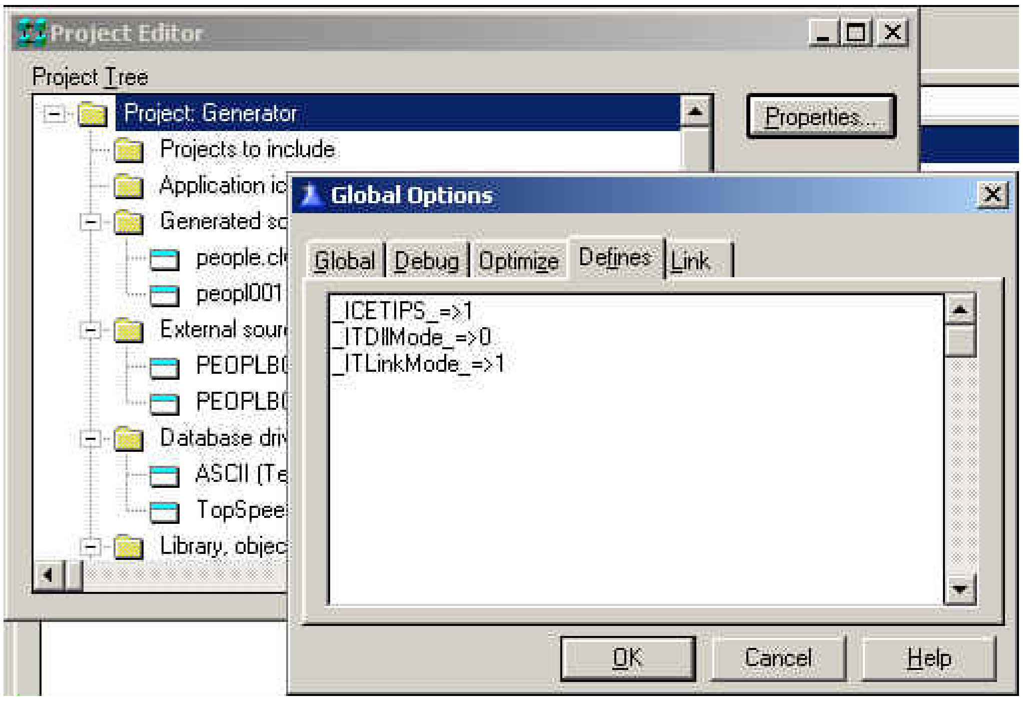
```
    OMIT('***END***',_ICETIPS_=1)
    Remember the Icetips Defines!

    ! Remember to add the Icetips defines to your project.
    ! Copy the following 3 lines and close this window.
    ! Then select Project|Edit from the main menu, click on the [Properties]
    ! button and select the "Defines" tab.  Paste the lines there.
  _ICETIPS_=>1
  _ITDllMode_=>0
  _ITLinkMode_=>1

    ***END***
  Include('ITMEClass.inc')
```

Incorrectly generated code to remind you what to do

Copy the three red lines to the clipboard, and close the editor. The select "Project|Edit" from the main menu, make sure that the top line is selected in the "Project Editor" window and click on the [Properties] button and then click on the "Defines" tab. Paste the three lines in there:



```
_ICETIPS_=>1
_ITDllMode_=>0
_ITLinkMode_=>1
```

Defines added to Project

Once this has been done, the application will compile correctly. If you are working with multi-dll project, it is important to copy the lines as they are generated as they will be generated correctly based on the settings in the project for target compile and export/external declarations. It is also very important that you copy the _ICETIPS_ =>1 line because that is the line that causes the compiler to omit the invalid code when that define is set to 1.

Images for the Magic Buttons

File formats

You can use any kind of images for the buttons that are suppored by Clarion. That includes icons (doesn't really make sense though!), bitmap BMP, GIF, JPG and PCX. Usually BMP or GIF are the best image format to use. BMP is not decoded and is usually very fast. However if you are using linked images, BMPs tend to be big because they are not compressed in any way. JPG/JPEG files are compressed and can be very slow.

Designing new images

No, I'm not going to teach you graphic desing - I'm not qualified for that! But there are some tips we can give you so you know what is a good starting point.

The images that you use for the Magic Buttons are placed under the buttons and sized to the exact same size as the button. This is done when the window opens, after the size and position has been reset from the ini file, if that option is used. This is also done automatically when the window resizes. The image you use, can be larger or smaller than the button. It will simply be stretched to match the button at runtime.

A good starting point for an image for regular buttons is 25x75 pixels. For small square buttons, use 25x25 and for big square buttons use 75x75 pixels.

Using Paint Shop Pro

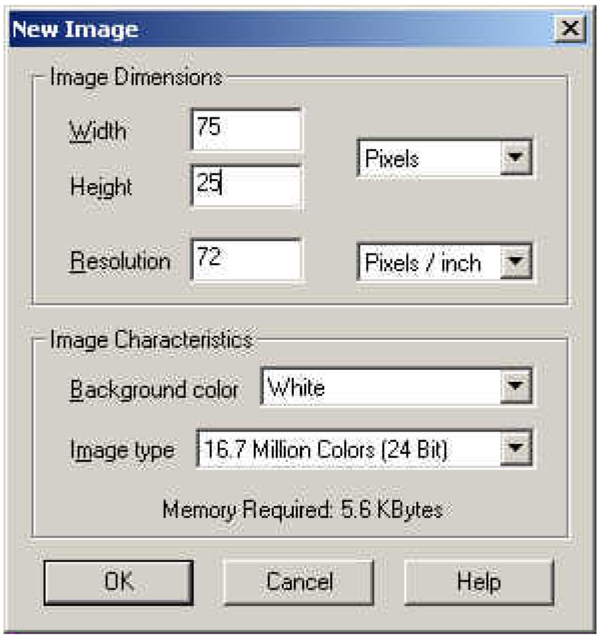

New Image

Image Dimensions
Width: 75
Height: 25
Pixels
Resolution: 72 Pixels / inch

Image Characteristics
Background color: White
Image type: 16.7 Million Colors (24 Bit)
Memory Required: 5.6 KBytes

Creating a new button image in Paint Shop Pro

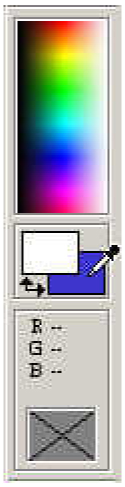
A lot of people have Paint Shop Pro and it can be used to create some beautiful buttons. In fact all the images that are supplied with the Magic Buttons are created with PaintShop Pro 6.0.

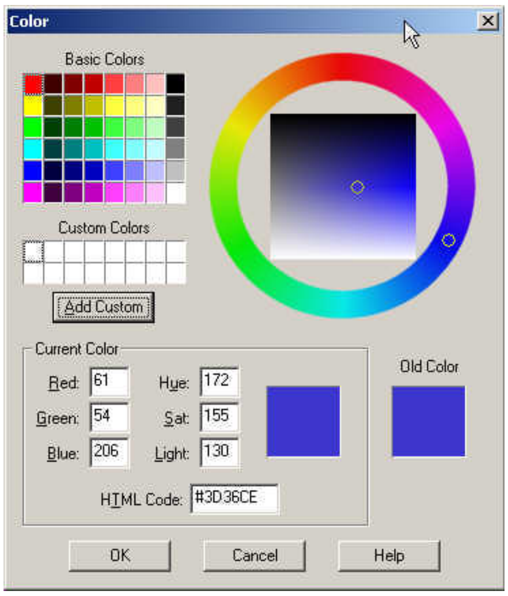
We are going to show you in a few simple steps, the best way to create a button image for the Magic Buttons.

First step is to create an empty image. Select "File|New" from the main menu in Paint Shop Pro (PSP) and enter the width of 75, height of 25, background color White and use 24bit image type. Then hit the OK button. The background color you select controls the main face color of the button.

We are going to use the Buttonize

special effect in PSP to create a nice button for us.
The first thing we should do is to select a color for the border of the button. We use the Color Palette toolbar to do that and we want to change the background color to the color that we want to use for our border color. When we use the Buttonize effect, the color we pick is smooted out to white in the middle of the button and grayed on the bottom and right of the button. You will probably need to experiment with this a bit to get the right color and buttonize settings for your needs. When you are satisfied with your selection, click the OK button and we can go on to the next step.





Color

Basic Colors

Custom Colors

Add Custom

Current Color

Red: 61 Hue: 172

Green: 54 Sat: 155

Blue: 206 Light: 130

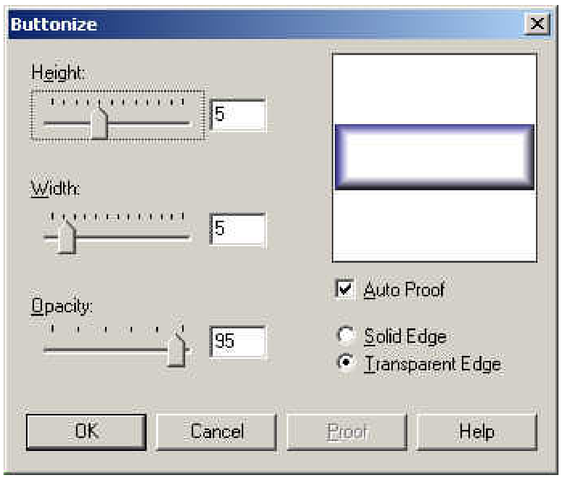
Old Color

HTML Code: #3D36CE

OK Cancel Help

Select the background color you want to use

To use the Buttonize effect, select "Image|Effects|Buttonize" from the main menu in Pain Shop Pro.



Buttonize

Height: 5

Width: 5

Opacity: 95

[✓] Auto Proof

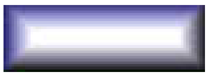
Solid Edge [✓ Transparent Edge]

OK Cancel Proof Help

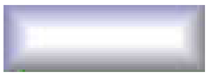
Buttonize the image

The buttonize effect window has some options that you can use to modify the buttonize effects. The height changes how far in from the top and bottom the "buttonizing" appears. The width controls the same from the sides. To the left you see two buttons with the height/width set to 5 and the other one where height and width are set to 10. The opacity controls how faded the image appears. The two first images were created with opacity set to 95, but the last one was created with opacity set to 50. The Copper button image was created starting with an image with a html background color of #FF9999 (Red=255, Green=153, Blue=153) The Golden button image was created starting with #FBB113 (251,177,19 RGB)

Button H=5, W=5



Button H=10, W=10



Button Opacit=50y

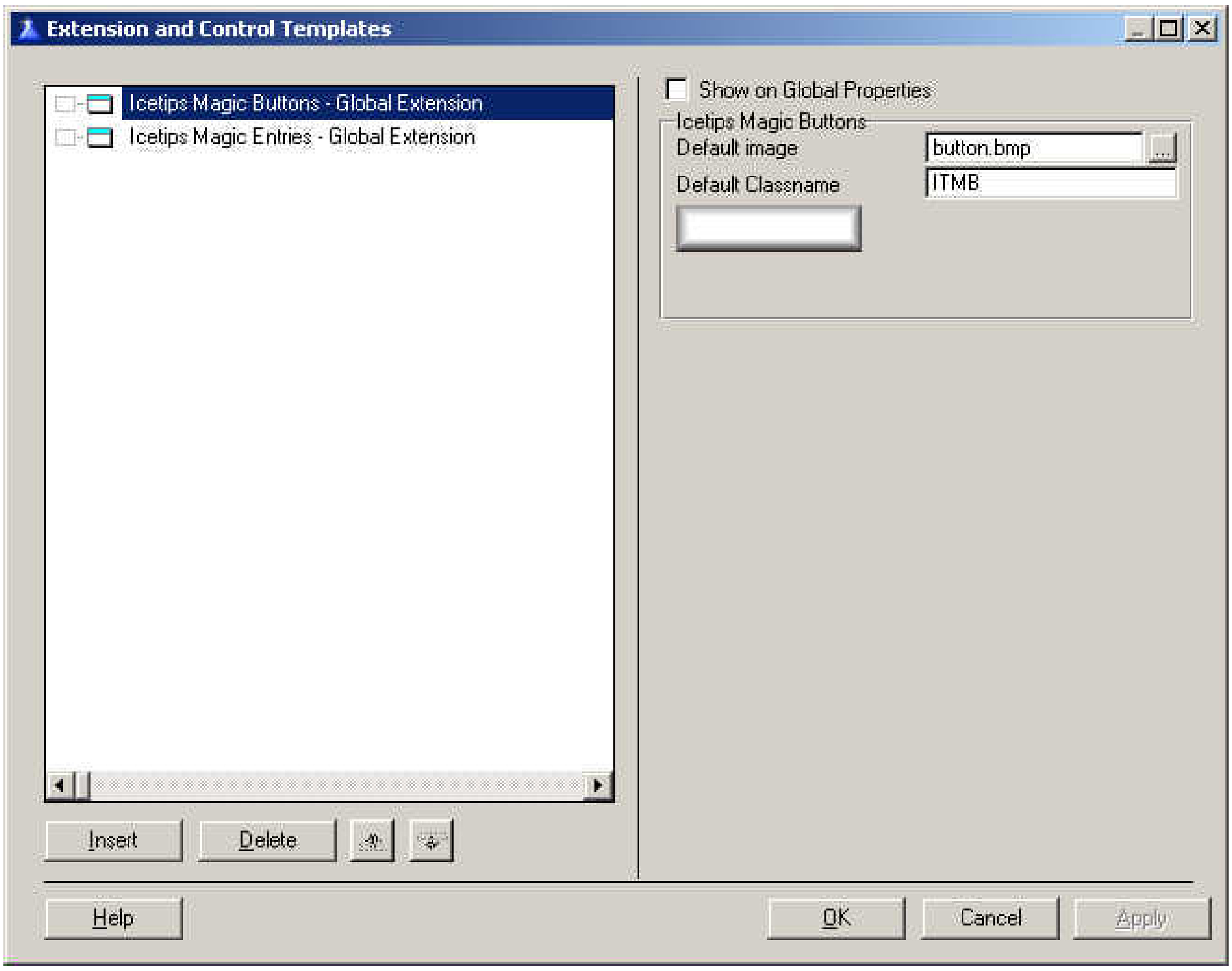
Copper button...

Golden button...

Global Extension Template

First, you populate the Icetips Magic Buttons Global Extension template into your app.

The Global Extension template allows you to set a default button image to use when you populate the Procedure Extension template. Click on the ellipsis button [...] to select an image file. The image you select will be visible immediately after you select it and return to the Global Extension template. The image you select here will be used on all


Extension and Control Templates
Icetips Magic Buttons - Global Extension
Icetips Magic Entries - Global Extension
Insert
Delete
Help
OK
Cancel
Apply

Show on Global Properties

Icetips Magic Buttons

Default image	button.bmp
Default Classname	ITMB

Global Extension Template

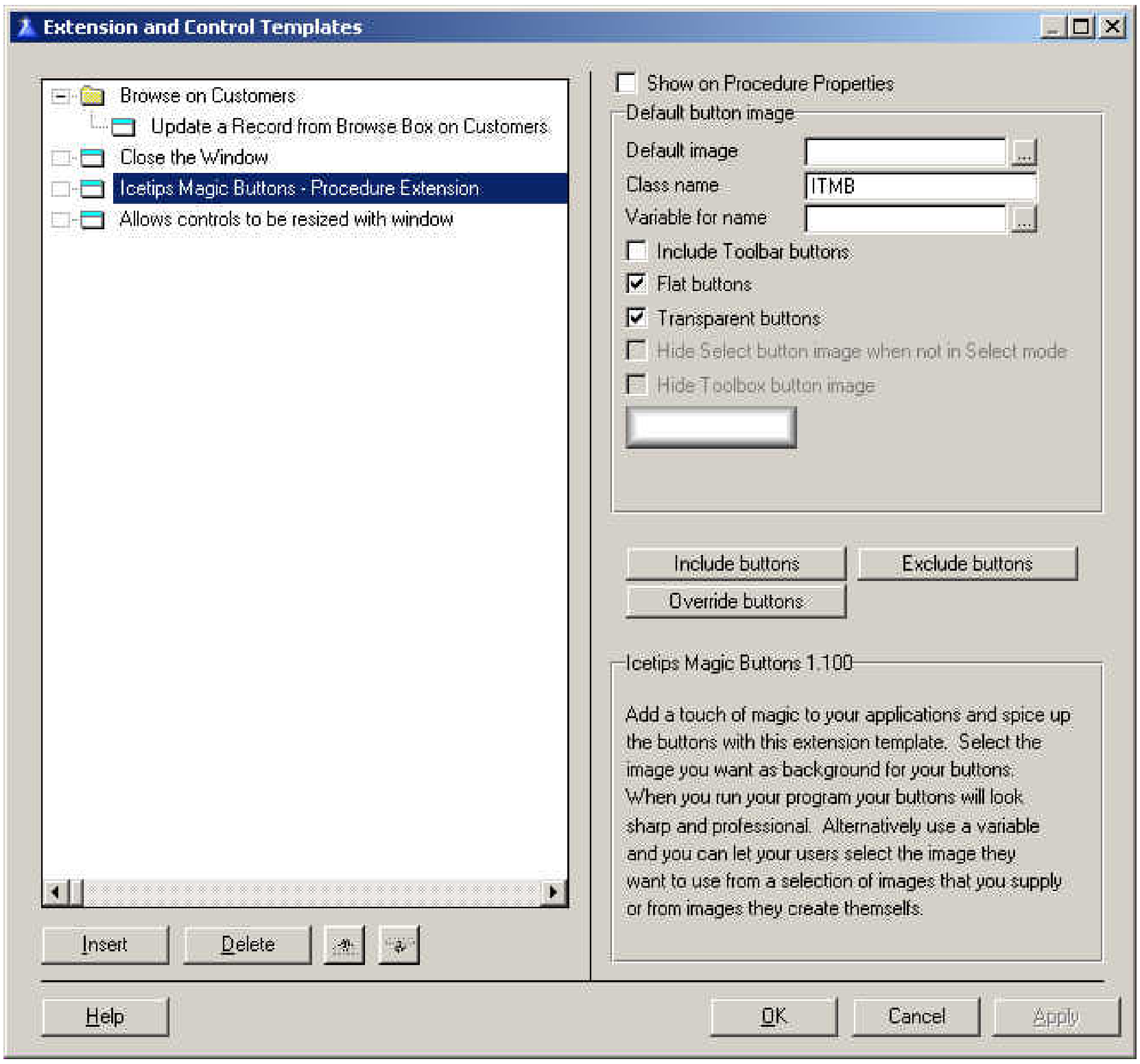
procedures with the Magic Button Procedure Extension template unless you select an image in the Procedure Extension. This allows you to quickly add a button image to your application and very quickly change it if you want to. The Default classname is the name that will be suggested when you apply the procedure extension template.

Procedure Extension Template

Then, you add the Icetips Magic Buttons Procedure Extension template to any procedures in your app that you want to use the Icetips Magic Buttons. (The Procedure Extension template will need to be added to each procedure using the Icetips Magic Buttons.)

The Procedure Extension template controls how the buttons will look, which button will get what image, what buttons will be excluded from getting images, etc.

When it is first populated, it will contain the image name that is populated in the Global Extension template. If the Global Extension template has not been populated, you will not be able to populate the Procedure Extension template.



Extension and Control Templates

- Browse on Customers
  - Update a Record from Browse Box on Customers
- Close the Window
- Icetips Magic Buttons - Procedure Extension
- Allows controls to be resized with window

[ ] Show on Procedure Properties

Default button image

Default image:
Class name: ITMB
Variable for name:

[ ] Include Toolbar buttons
[✓] Flat buttons
[✓] Transparent buttons
[ ] Hide Select button image when not in Select mode
[ ] Hide Toolbox button image

Icetips Magic Buttons 1.100

Add a touch of magic to your applications and spice up the buttons with this extension template. Select the image you want as background for your buttons. When you run your program your buttons will look sharp and professional. Alternatively use a variable and you can let your users select the image they want to use from a selection of images that you supply or from images they create themselfs.

Procedure Extension Template applied

Default image

The Default image is populated from the Global Extension template. This image is linked into your software, so if you use a lot of different button images in your app, it may put strain on the resources in your application. If you leave this empty, the template will use the image selected in the Global Template.

If you want to refer to this image in your code, you need to use the tilde, like:

```
'~button.bmp'
```

Class name

The classname is the name that is used for the MagicButtonsClass instance in the procedure. This defaults to ITMB (IceTips Magic Buttons). You can use this name to access all the methods and properties in the class from anywhere in your code.

Variable for name

This allows you to use a variable for the image file name and construct it into this field. If the variable is filled, it will be used, even if the default image is filled too. The variable has a higher priority when it comes to the code generation.

Note that this is an expression field, so you can type into it as well as select variables to build up the expression that is used to construct the image name. You can use internal, linked resources by using a tilde in front of the name. Example:

```
'~' & Clip(Loc:ImageName) & '.bmp'
```

If you do not use the tilde, your program will look for:

```
Clip(Loc:ImageName) & '.bmp'
```

on the computer drives instead of in your application. This only applies to the variable name.

Include Toolbar Buttons

If this is checked, the template will also change buttons on toolbars. The default is unchecked. Use this if you want to put images on toolbar buttons, or override them.

Flat Buttons

This sets the FLAT attribute on the buttons at runtime. Checked by default

Transparent Buttons

This sets the TRN attribute on the buttons at runtime. Checked by default. Note that in Clarion 5, this attribute only has effects if the button has an icon.

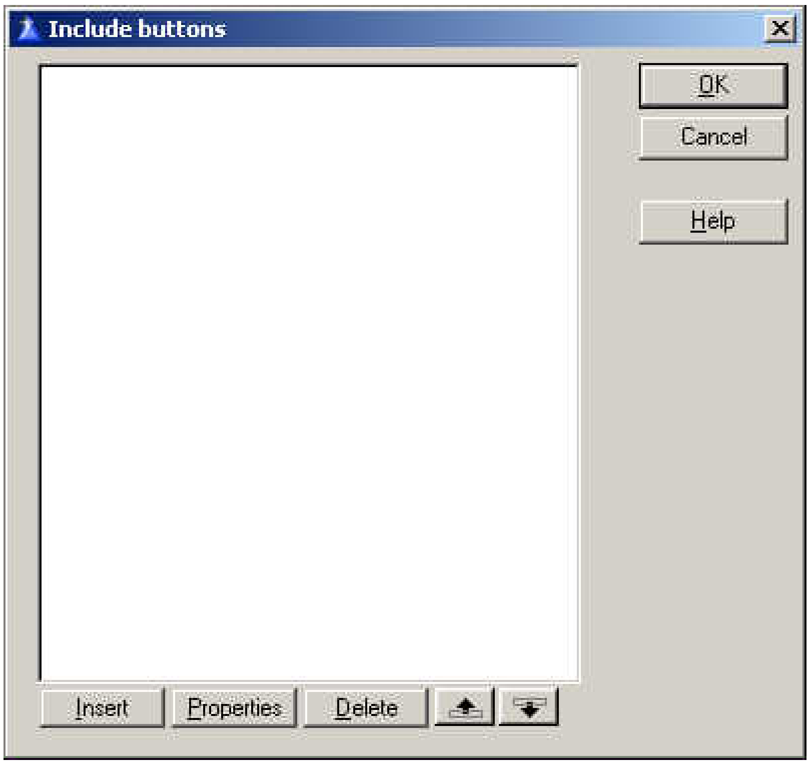
Hide Select Button image when not in select mode

This is enabled if a Select button control template is found on the window. If it is not checked the image behind the select button will not be hidden if the window is in SelectRecord mode. If it is checked, the image is hidden.

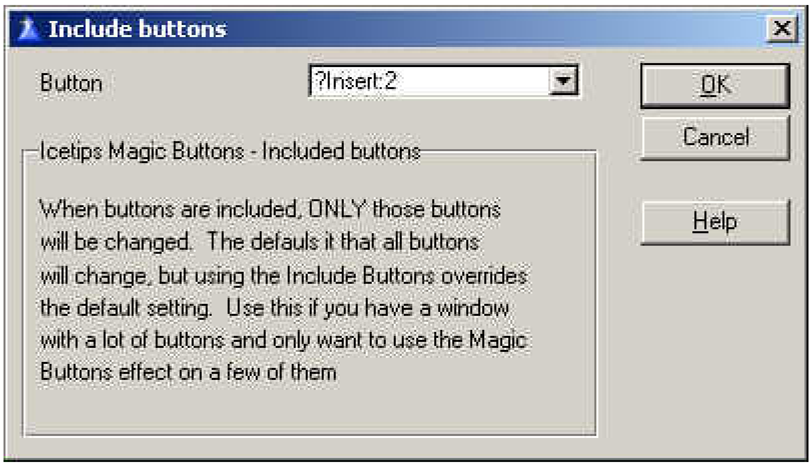
Hide toolbox image

This was a user request for applications generated by the Wizatrons. Similar to the Select button option, this is enabled only if a toolbox control template is found on the window.

Include Buttons



Adding buttons to Include



Button ?Insert:2

Icetips Magic Buttons - Included buttons

When buttons are included, ONLY those buttons will be changed. The defauls it that all buttons will change, but using the Include Buttons overrides the default setting. Use this if you have a window with a lot of buttons and only want to use the Magic Buttons effect on a few of them

Including a button by selecting from a list of buttons

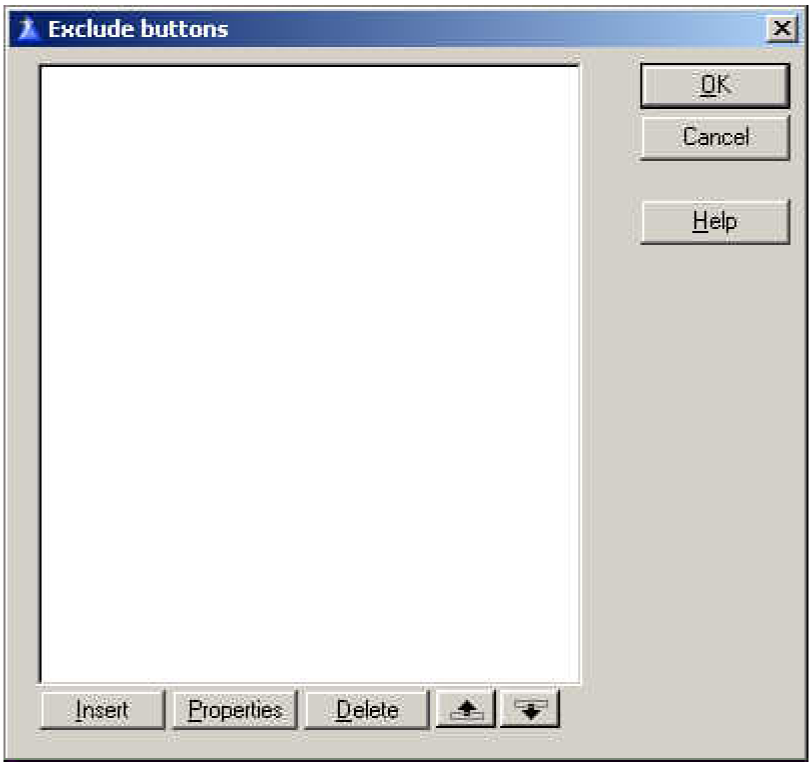
Included buttons is used when you want only some of the buttons on your window to get the image buttons effect.

Normally the extension template changes all the buttons on the window, but if you use the Include Buttons option, ONLY the buttons that you specify here will be included.

Note: If a button that is included is also in the Excluded buttons list (see the following page), it will be excluded, i.e. the Exclude has priority over Include.

This option is very useful if you have a window that has a lot of buttons, but you only want to add the Magic Buttons effects to a few of them. That way you can tell the template what buttons to include.

Exclude Buttons

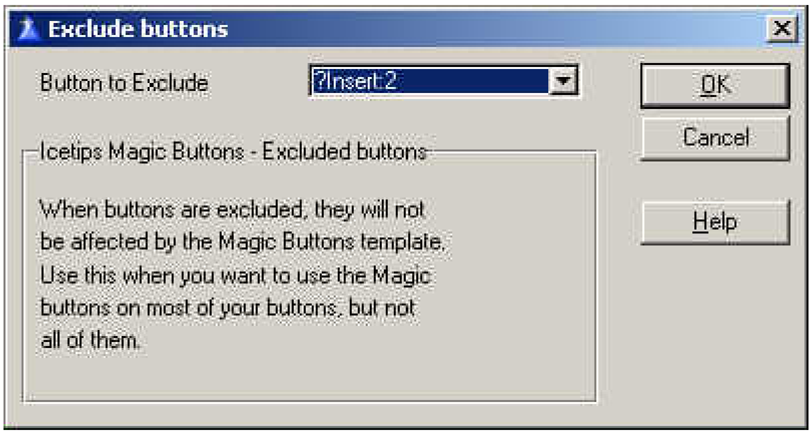


Adding Buttons to Exclude

Excluding buttons is where you would specify ONLY the buttons to be excluded, and all other buttons on the window will get the Magic Buttons effect.

This is the reverse approace to using the "Include Buttons", noted on the previous page.

This is very useful when you have a window where you want to leave some buttons untouched.



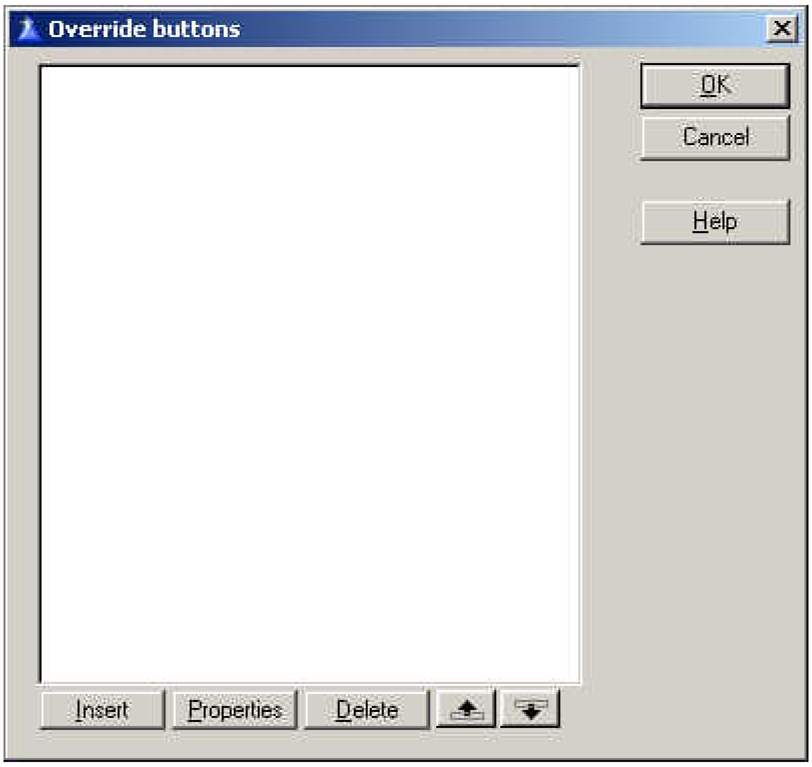
Button to Exclude ?Insert:2

Icetips Magic Buttons - Excluded buttons

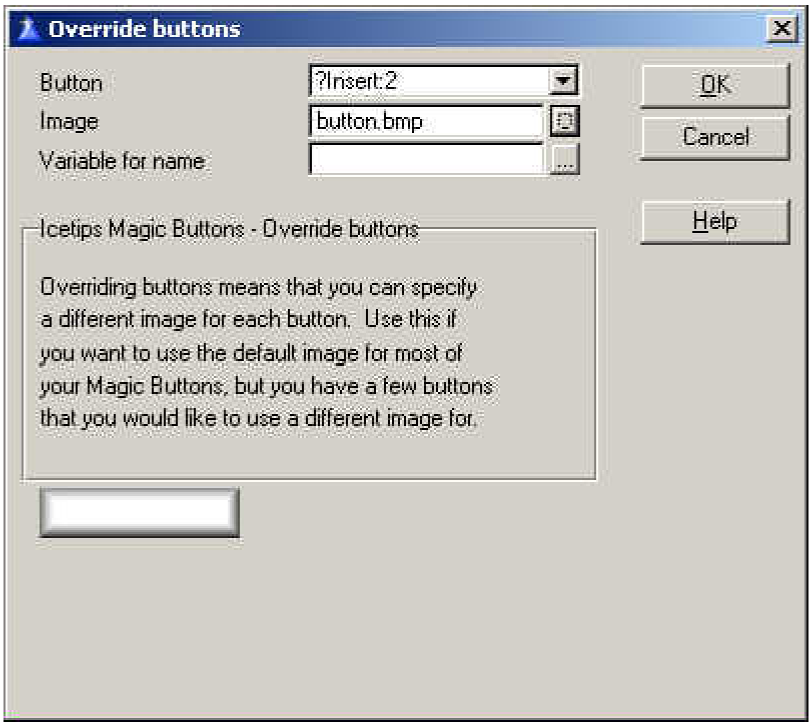
When buttons are excluded, they will not be affected by the Magic Buttons template. Use this when you want to use the Magic buttons on most of your buttons, but not all of them.

Excluding a button by selecting from a list of buttons

Override Buttons



Adding buttons to override



Button: ?Insert:2
Image: button.bmp
Variable for name:

Icetips Magic Buttons - Override buttons

Overriding buttons means that you can specify a different image for each button. Use this if you want to use the default image for most of your Magic Buttons, but you have a few buttons that you would like to use a different image for.

Overriding the default settings for a button

The Override option allows you to override the settings for buttons individual buttons. With this, you can select a different image for the selected button (instead of using the default image). Or, you can set a new/different expression for the variable filename.

This is very useful when you need to change the settings for one or more buttons.

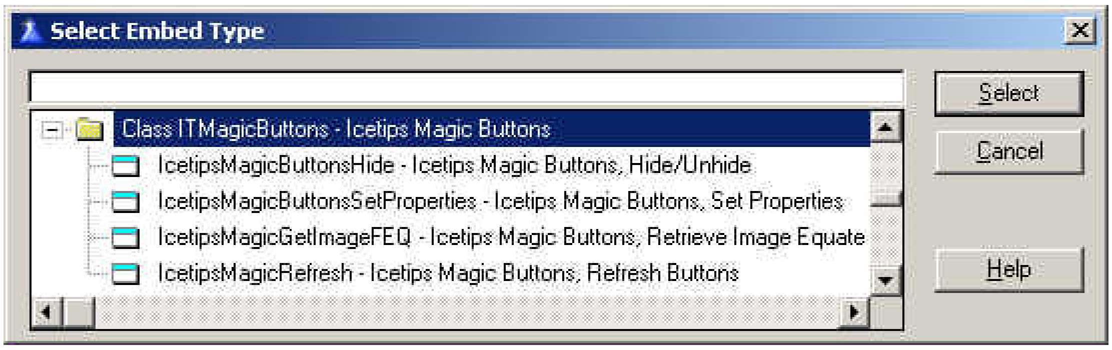
When you select an image to display, it will automatically update the image displayed on the Override window so you can see what image you just picked.

The same applies here as with the default settings. You can use an expression to use a filename variable and build up an expression if needed.

When you pick an image, the image at the bottom of the window will automatically update to show the image just picked.

Code templates

The Icetips Magic Buttons include 5 code templates to make life easier if you need to manipulate the image control for a particular button. These code templates allow you to hide/unhide the image or both the image and the button. This section will show how each code template operates.

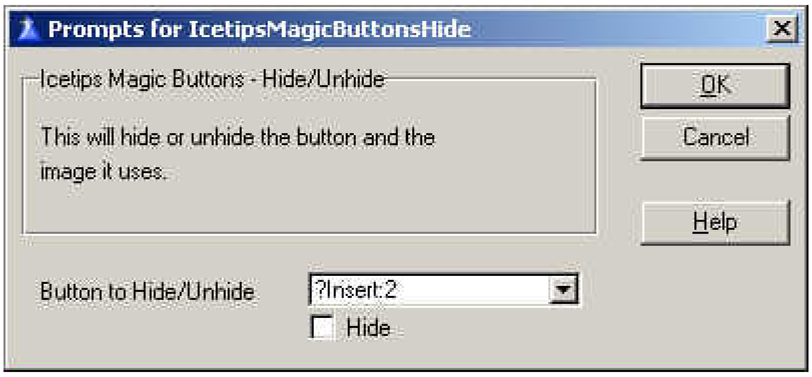


Class ITMagicButtons - Icetips Magic Buttons
IcetipsMagicButtonsHide - Icetips Magic Buttons, Hide/Unhide
IcetipsMagicButtonsSetProperties - Icetips Magic Buttons, Set Properties
IcetipsMagicGetImageFEQ - Icetips Magic Buttons, Retrieve Image Equate
IcetipsMagicRefresh - Icetips Magic Buttons, Refresh Buttons

Selecting any of the Icetips Magic Buttons code tamplates

Hide

When you hide a button that uses the Icetips Magic Buttons, you need to hide the image that is used to create the button's background. This code template allows you to do that very easily. Select the button control you want to hide or unhide. If you want to hide it, check the Hide checkbox and if you want to unhide the button, leave the



Icetips Magic Buttons - Hide/Unhide

This will hide or unhide the button and the image it uses.

Button to Hide/Unhide: ?Insert:2

[ ] Hide

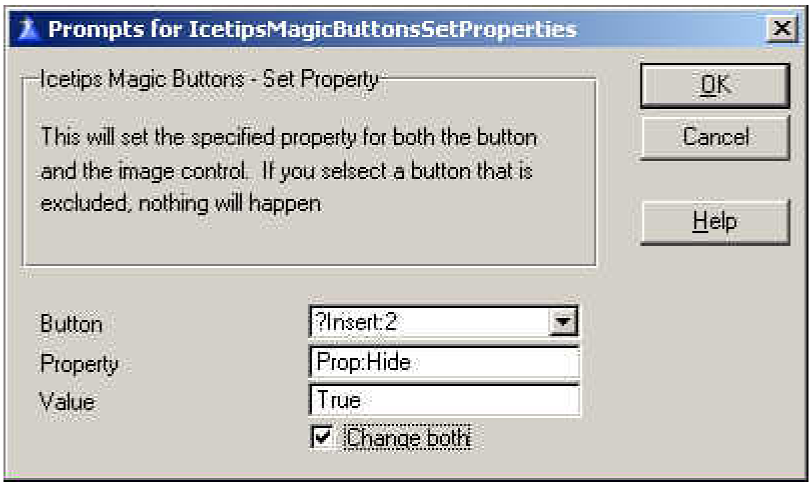
Selecting a button control to hide/unhide with the image

Hide checkbox unchecked. That is all there is to it! You can also do this with the next code template if you want to, but it can also do much more than the Hide code template. Alternatively you could also use the HideButton and UnhideButton methods in the MagicButtonClass, for example:

```
ITMB.HideButton(?Insert:2)
```

SetProperties

This template allows you to set whatever property you want for the button and/or the image behind it. Select the button whose image you want to set properties for, type in the property and the value and optionally check the "Change both" checkbox if you want this property to be used on both the image and the button.


Prompts for IcetipsMagicButtonsSetProperties

Icetips Magic Buttons - Set Property

This will set the specified property for both the button and the image control. If you select a button that is excluded, nothing will happen

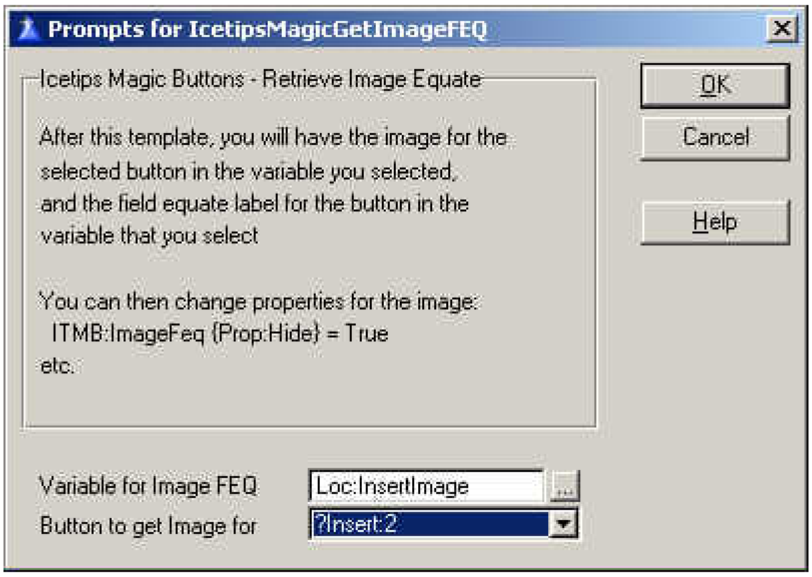
Button	?Insert:2
Property	Prop:Hide
Value	True
	☑ Change both

Using the SetProperties code template

In this case, it is the Prop:Hide, it's set to true and so it should change both controls. The results would be that the button is hidden along with the button image. To accomplish the same thing in handcode, you would need to use something like the following:

```
If ITMB.RetrieveButton(?Insert:2)
  ITMB.Buttons.ButtonFEQ {Prop:Hide} = True
  ITMB.Buttons.ImageFEQ {Prop:Hide} = True
End
```

GetImageFEQ


Prompts for IcetipsMagicGetImageFEQ

Icetips Magic Buttons - Retrieve Image Equate

After this template, you will have the image for the selected button in the variable you selected, and the field equate label for the button in the variable that you select

You can then change properties for the image:
ITMB:ImageFeq {Prop:Hide} = True
etc.

Variable for Image FEQ	Loc:InsertImage
Button to get Image for	?Insert:2

Retrieving the Field Equate label for the button image

This code template will retrieve the Field EQuate (FEQ) label of the button image, so you can modify it seperately by code.

Please also see technical issues on next page for some more in-depth information about how this can be accomplished directly in source code. Note that in version 1.100 you must select a variable to receive the FEQ of the image.

SetMultiProperties

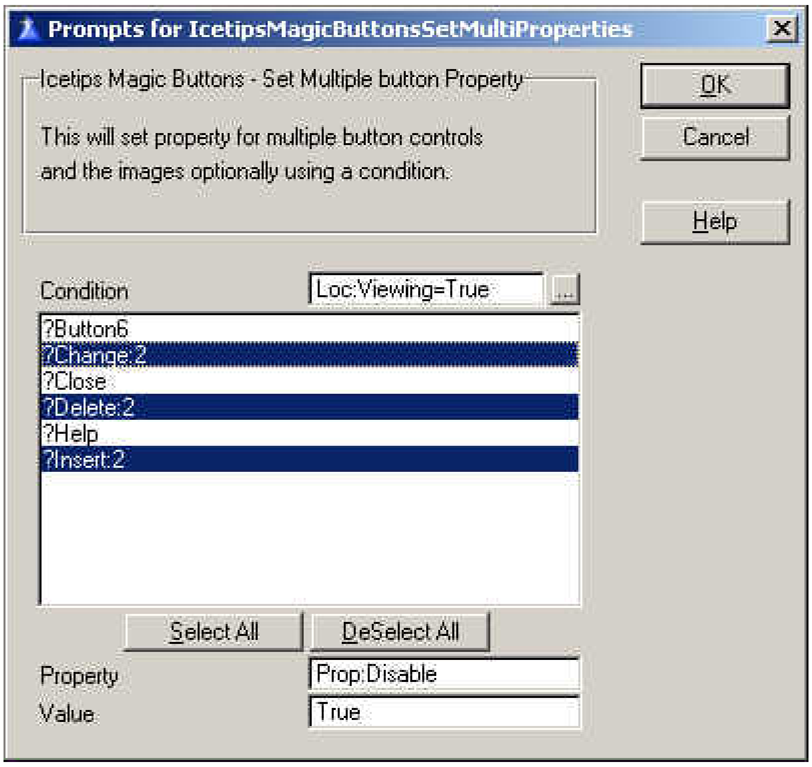
This is a very powerful code template, that allows you to change a property for multiple controls and even add a condition for the property setting as well. The optional condition is used in an IF/END statement, i.e. in the example in the screenshot it would generate

```
If Loc:Viewing=True
  ! Code setting properties here
End
```

The property specified is set for both the image controls and the buttons.

This would normally be of great value to set multiple controls as hidden. This will generate property setting for each button and each image.

The code for the screenshot to the left will be generated as shown below.



Prompts for IcetipsMagicButtonsSetMultiProperties

Icetips Magic Buttons - Set Multiple button Property

This will set property for multiple button controls and the images optionally using a condition.

Condition: Loc:Viewing=True

?Button6
?Change:2
?Close
?Delete:2
?Help
?Insert:2

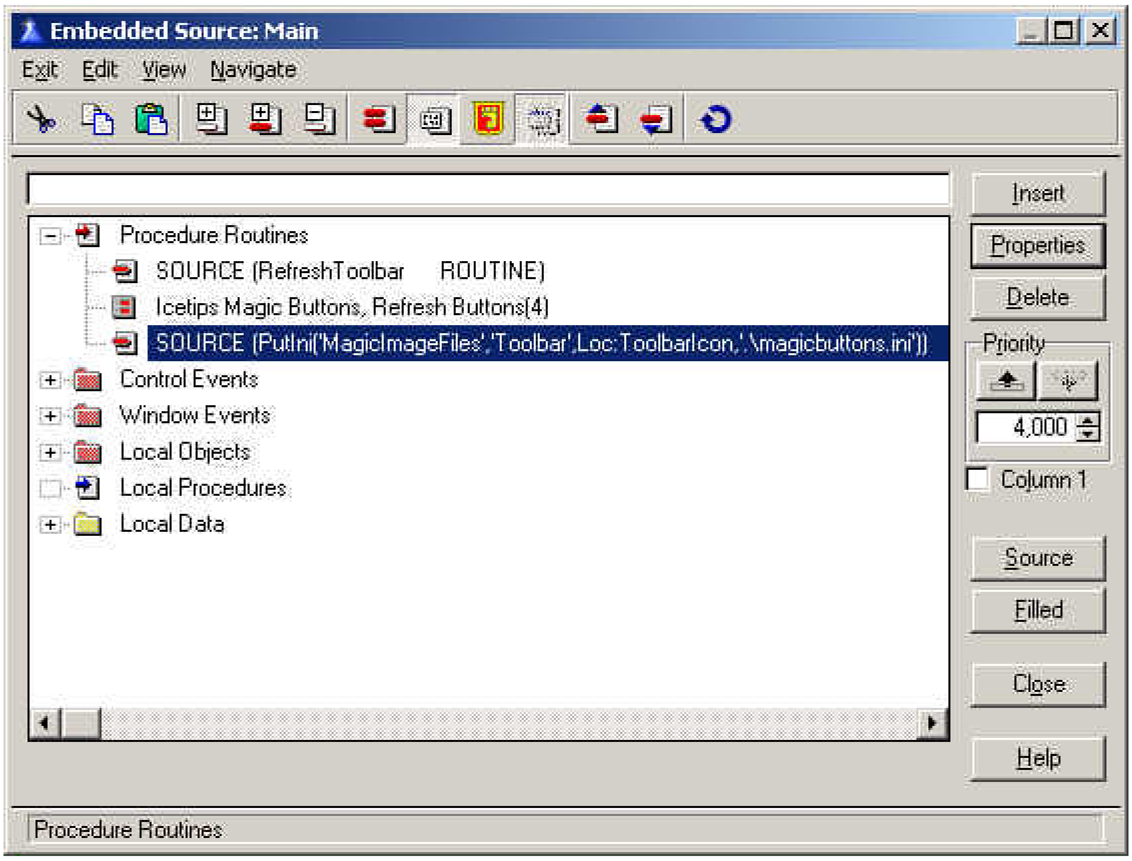
Property: Prop:Disable
Value: True

Set properties for multiple controls

```
If Loc:Viewing=True
  If ITMB.RetrieveButton(?Change:2)
    ITMB.Buttons.ImageFEQ  {Prop:Disable} = True
    ITMB.Buttons.ButtonFEQ {Prop:Disable} = True
  End
  If ITMB.RetrieveButton(?Delete:2)
    ITMB.Buttons.ImageFEQ  {Prop:Disable} = True
    ITMB.Buttons.ButtonFEQ {Prop:Disable} = True
  End
  If ITMB.RetrieveButton(?Insert:2)
    ITMB.Buttons.ImageFEQ  {Prop:Disable} = True
    ITMB.Buttons.ButtonFEQ {Prop:Disable} = True
  End
End
```

Refresh

This template resets all the images that use a variable image name. This is necessary to use when you have selected a new image for the buttons. Simplest way to do this is to create a routine, that way you only need to populate this template once in your procedure.



```
Procedure Routines
    SOURCE (RefreshToolbar      ROUTINE)
    Icetips Magic Buttons, Refresh Buttons(4)
    SOURCE (PutIni('MagicImageFiles','Toolbar',Loc:ToolbarIcon,'.\magicbuttons.ini'))
Control Events
Window Events
Local Objects
Local Procedures
Local Data
```

Procedure Routines

In the image above you can see the embedded code that is needed to do this as well as save the image selected to an ini file for easy retrival next time the window is opened.

Optionally you can use the RefreshAll method:

```
ITMB.RefreshAll
```

Class reference

The Icetips Magic Buttons are based on the ITMagicButtonsClass class. This allows you to implement and use the Magic Buttons in hand coded projects as well as application based projects. The class is stored in ITMBClass.inc and the code in ITMBClass.clw which are both installed into the 3rdParty\LibSrc directory.

Class Properties

Buttons	**&MBFeqQ**

The Buttons property is a reference to a queue:

```
MBFeqQ   Queue,Type
ButtonFEQ  Long
ImageFEQ   Long
ButtonTrn  Byte
ButtonFlat Byte
ImageFile  Cstring(1025)
         End
```

This queue is instanciated in the Init method and used to keep track of the buttons and the images that are created behind the buttons.

ButtonImage	**Cstring(1025)**

The ButtonImage is the default image used for the buttons. Each button get's individual image assigned so it can be changed. The ButtonImage is assigned in the Init method from pImage parameter passed to the method.

Initialized	**Byte**

This variable is set to True in the Init method if an image is passed to it. The image parameter is required and if it is empty the class methods will not do anything.

ButtonsFlat	**Byte**

Specifies if the buttons should be turned flat by default.

ButtonsTrn	**Byte**

Specified is the buttons should be turned transparent by default.

Class Methods

Init **Procedure(String pImage,Byte pFlat,Byte pTrn)**

The Init method instanciates the Buttons queue and sets the default image, as well as the default flat and transparent attributes.

RegisterWindow **Procedure(Byte pToolbar,Byte pIsFrame)**

This registers all buttons on a window to be used with the Magic Buttons. Registering simply means that the button is added to the Buttons queue, and some attributes set.

The pToolbar parameter tells the method if toolbar buttons should be included or not. The pIsFrame tells the method if the window is an application frame window or not. This is important as the toolbar controls are handled differently on frames than on any other window.

If the toolbar is set and this is a not a frame procedure, all toolbar buttons are ignored by this method. Instead they are registered individually. The reason is that when the controls are scanned on a non-frame window, all controls from the frame are visible as well. There is no difference in the properties returned by appframe toolbar controls and the window toolbar controls, so it is impossible to distinguish one from the other.

RegisterButton **Procedure(Long pFEQ,<String pImage>)**

This method registers one button to be added to the Magic Buttons queue. It takes the button label and optionally an image for the button. If the pImage parameter is omitted the default image passed to the Init method is used. This method is called from the RegisterWindow method for each button on the window.

ApplyWindowMagic **Procedure**

This method loops through all the registered buttons and calls the ApplyButtonMagic method for each button.

ApplyButtonMagic **Procedure(Long pFEQ)**

This method creates the image control behind the button, applies the image specified in the Self.Buttons.ImageFile field, sets the button's transparent and flat attributes, positions the image and unhides it. This method is called from the ApplyWindowMagic for each registered button.

RetrieveButton **Procedure(Long pFEQ),Byte**

This method retrieves the record from the Buttons queue corresponding to the passed button label. If the button is found the method returns true, otherwise it returns false.

HideButton **Procedure(Long pFEQ)**

This method hides the button and the associated image.

UnhideButton **Procedure(Long pFEQ)**

This method unhides the button and the associated image.

SetImagePos **Procedure(Long pFEQ)**

This method sets the image for the button to the same position as the button. This method is called from the SetAllImagePos method which sets the positions for all registered buttons on the window.

SetAllImagePos **Procedure**

This method synchronizes the positions of the images and the buttons for all the registered buttons. This method calls the SetImagePos method for each registered button on the window.

ChangeImage **Procedure(Long pFEQ,String pImage)**

This method retrieves the Buttons record for the passed button label and assigns a new image to use for it. It does not reset the image on the window, only the image name in the Buttons queue.

SetImage **Procedure(Long pFEQ,String pImage)**

This method retrieves the Buttons record for the passed button label and assigns a new image to use for it and saves it back to the queue. It refreshes the image on the window as well.

RefreshImage **Procedure(Long pFEQ)**

This method refreshes the image for the passed button label to what it is in the Buttons queue. This method is called from the RefreshAll method for each registered button.

RefreshAll **Procedure**

This method refreshes all the button images. It uses Prop:LazyDisplay to speed the redraws up as much as possible.

RemoveButton **Procedure(Long pFEQ)**

This method removes the button from the Buttons queue and destroys the image associated with it.

Kill **Procedure**

This method destroys all the button images on the window, frees the Buttons queue and disposes of it.

Handcoding

Because of the classes in version 1.1, it is now easy to use the Magic Buttons in handcoded projects. It must be noted that the Magic Buttons classes are NOT ABC compliant.

Include classfile

To use the classes in a handcoded project the first thing to do is to include the inc file in the main source module:

 Include('ITMBclass.inc')

Project defines

It is also required that you put a couple of defines into the project.   The following is correct based on the target and if the project exports data or where data is external:

DLL

Exporting:

```
_ITDllMode_=>0
_ITLinkMode_=>1
```

External:

```
_ITDllMode_=>1
_ITLinkMode_=>0
```

EXE

Standalone:

```
_ITDllMode_=>0
_ITLinkMode_=>1
```

External:

```
_ITDllMode_=>1
_ITLinkMode_=>0
```

Adding to the procedure

To add the class to your procedure, add the following to the datasection of your procedure:

```
ITMB   ITMagicButtonsClass
```

This creates an object names ITMB which you can then use to access the class methods. You should not use any of the methods until after you have opened the window in the procedure. Appropriate coding would be along these lines:

```
W    Window...
       ...
     End
ITMB ITMagicButtonsClass

 Code
 Open(W)
 Display

 ITMB.Init('~button.bmp',1,1)
 ITMB.RegisterWindow(1,0)
 ITMB.ApplyWindowMagic

 Accept
   ...
 End

 ITMB.Kill
 Close(W)
```

The Magic Buttons classes can be used in ABC applications as well as Clarion template applications in exactly the same way.

The above code obviously applies to both hand coded projects as well as hand coded procedures (source procedures).

Compatibility and Technical issues

The Icetips Magic Buttons are compatible with both the ABC and Clarion template chains, in Clarion versions 5 and 5.5. (The templates were tested in Clarion 5B and Clarion 5.5E.)

There is one notable difference between Clarion 5 and Clarion 5.5 that affects the Magic Buttons and that is the Transparent property on buttons. In Clarion 5.5 it affects all buttons, but in Clarion 5 it only affects buttons that use an icon. Therefor you may not get exactly the same visual effects in Clarion 5 as you can in Clarion 5.5. This is a limitation of the Clarion runtime in Clarion 5, and has nothing to do with the Icetips Magic Buttons.

There have been requests for utility that could add the Magic Buttons automatically to all or selected procedures in applications. Such a utility exists, but it is not supplied with the product. The reason is that it exports the procedures to TXA, makes changes to the TXA and then imports the TXA back into the application, replacing the original procedure. In most cases this works without problems. But in some cases the export/import process loses contact with some template symbols that are not exported correctly and are thereby lost on the import. This is not a problem with our utility, but a problem in the Clarion TXA export/import process. But it also means that we can not rely on our utility to work. If you are in dire need for it, we will supply it to you for your own use at your own risk. Usually if you can export the application to TXA, create a new empty application and import the TXA and have it work without any problems, this utility should work without problems also. But we are not taking the chances of it accidentally corrupting your applications!

Technical Support

We offer technical support by email, by newsgroup, or by an internet bulletin board.

Email

Please email your questions to either **`support@icetips.com`** or **`wizard@icetips.com`** and we will get back to you as soon as possible. We usually respond to technical support emails within an hour.

Newsgroups

You can also post questions on the **`Topspeed.Topic.Third_Party`** newsgroup on the **`news.softvelocity.com`** news server or **`comp.lang.clarion`**, which you can get to at that same news server or on the web at:

`http://groups.google.com/groups?hl=is&group=comp.lang.clarion`

Internet Bulletin Board

We have a Internet Bulletin Board at the Icetips website, where you are welcome to post questions. We monitor it regularly, and there are quite a few people who visit it frequently. Go to **`http://www.icetips.com/wwwboard/index.htm`**

Installed files

Following is a complete list of the installed files. Please note that it is possible that the dates/times and file sizes does not match completely with what is installed as this list was created before everything was completed. Also note that the file dates are in dd.mm.yyyy format.

Files in: 3rdParty\Docs\ITMagicButtons

```
Date        Time               Size Filename
-------------------------------------------------------
08.08.2002  16:00         1.595.709 ITmagicbuttons.pdf
(Note that this file is NOT the right size)
```

Files in: 3rdParty\Examples\ITMagicButtons

```
Date        Time               Size Filename
-------------------------------------------------------
08.08.2002  15:47            43.008 mlegacy.app
08.08.2002  15:46            93.184 magicbuttons.app
24.04.2002  17:09             5.754 blue-btn.bmp
24.04.2002  17:05            17.154 blue-wht-btn-lg.bmp
24.04.2002  17:05             1.954 blue-wht-btn-sm.bmp
24.04.2002  17:05             5.754 blue-wht-btn.bmp
24.04.2002  14:23            15.530 blue1-bg.bmp
24.04.2002  14:23           196.662 blue2-bg.bmp
24.04.2002  13:42           196.662 bluegold-bg.bmp
24.04.2002  17:09             1.954 blue-btn-sm.bmp
20.04.2002  16:43             7.554 Button2.bmp
24.04.2002  16:48             5.754 gold-wht-btn.bmp
24.04.2002  16:48             1.954 gold-wht-btn-sm.bmp
24.04.2002  16:48            17.154 gold-wht-btn-lg.bmp
24.04.2002  17:12             5.754 gold-btn.bmp
24.04.2002  17:09            17.154 blue-btn-lg.bmp
24.04.2002  13:46            17.154 Marble7575.bmp
24.04.2002  13:55             5.754 Marble2575.bmp
24.04.2002  13:54             1.954 Marble2525.bmp
24.04.2002  13:42           196.662 gold1-bg.bmp
20.04.2002  16:30             7.554 Button.bmp
24.04.2002  16:29             1.954 gold-btn-sm.bmp
24.04.2002  13:44           921.654 gray3-bg.bmp
24.04.2002  13:43           196.662 gray2-bg.bmp
24.04.2002  13:43           196.662 gray1-bg.bmp
24.04.2002  16:49             5.754 gray-wht-btn.bmp
24.04.2002  16:50             1.954 gray-wht-btn-sm.bmp
24.04.2002  16:50            17.154 gray-wht-btn-lg.bmp
24.04.2002  17:18             5.754 gray-btn.bmp
24.04.2002  17:14            17.154 gold-btn-lg.bmp
24.04.2002  17:19             1.954 gray-btn-sm.bmp
24.04.2002  17:19            17.154 gray-btn-lg.bmp
24.04.2002  13:42           196.662 gold2-bg.bmp
24.04.2002  13:58             2.048 Magicbt.dct
02.08.2002  12:46         1.140.736 magicbuttons.exe
24.04.2002  15:36               984 transparent-btn.gif
24.04.2002  15:38             1.123 transparent-btn-lg.gif
24.04.2002  15:39               932 transparent-btn-sm.gif
15.02.2002  23:18             5.823 b_ITBrowse.gif
12.04.2002  14:25            34.144 b_Frame.gif
15.02.2002  23:23             4.269 b_ITForm.gif
```

```
15.02.2002  23:01            3.020 b_Toolbar.gif
15.02.2002  21:35            2.238 b_IThelporig.ico
18.01.2002  13:14            2.238 b_ITsearch.ico
15.02.2002  18:41            2.238 b_ITPrior.ico
16.02.2002  10:43            2.238 b_ITok.ico
18.01.2002  13:21            2.238 b_ITnoprint.ico
17.01.2002  01:07            2.238 b_ITnextpage.ico
15.02.2002  19:05            2.238 b_ITNext.ico
15.02.2002  19:40            2.238 b_ITmark.ico
17.01.2002  01:07            2.238 b_ITlast.ico
15.02.2002  19:45            2.238 b_ITinsert.ico
17.01.2002  01:08            2.238 b_ITprevpage.ico
16.02.2002  11:14            2.238 b_IThelp.ico
16.02.2002  10:50            2.238 b_ITcancel.ico
17.01.2002  01:05            2.238 b_ITfirst.ico
15.02.2002  19:38            2.238 b_ITedit.ico
15.02.2002  21:32            2.238 b_ITditto.ico
15.02.2002  19:47            2.238 b_ITdelete.ico
18.02.2002  12:41            2.238 b_ITclose.ico
25.04.2002  11:31              243 magicbuttons.ini
25.04.2002  11:32          298.240 customers.tps
              62 File(s)      3.974.592 bytes
```

Files in: 3rdParty\Images

```
Date        Time               Size Filename
--------------------------------------------------------
24.04.2002  17:09           17.154 blue-btn-lg.bmp
24.04.2002  17:09            1.954 blue-btn-sm.bmp
24.04.2002  17:09            5.754 blue-btn.bmp
24.04.2002  17:05           17.154 blue-wht-btn-lg.bmp
24.04.2002  17:05            1.954 blue-wht-btn-sm.bmp
24.04.2002  17:05            5.754 blue-wht-btn.bmp
24.04.2002  14:23           15.530 blue1-bg.bmp
24.04.2002  14:23          196.662 blue2-bg.bmp
24.04.2002  13:42          196.662 bluegold-bg.bmp
20.04.2002  16:30            7.554 Button.bmp
20.04.2002  16:43            7.554 Button2.bmp
24.04.2002  16:48           17.154 gold-wht-btn-lg.bmp
24.04.2002  17:12            5.754 gold-btn.bmp
24.04.2002  16:29            1.954 gold-btn-sm.bmp
24.04.2002  17:14           17.154 gold-btn-lg.bmp
24.04.2002  13:46           17.154 Marble7575.bmp
24.04.2002  13:55            5.754 Marble2575.bmp
24.04.2002  13:54            1.954 Marble2525.bmp
24.04.2002  13:44          921.654 gray3-bg.bmp
24.04.2002  16:48            5.754 gold-wht-btn.bmp
24.04.2002  13:43          196.662 gray2-bg.bmp
24.04.2002  13:43          196.662 gray1-bg.bmp
24.04.2002  16:49            5.754 gray-wht-btn.bmp
24.04.2002  16:50            1.954 gray-wht-btn-sm.bmp
24.04.2002  16:50           17.154 gray-wht-btn-lg.bmp
24.04.2002  17:18            5.754 gray-btn.bmp
24.04.2002  17:19            1.954 gray-btn-sm.bmp
24.04.2002  13:42          196.662 gold1-bg.bmp
24.04.2002  17:19           17.154 gray-btn-lg.bmp
24.04.2002  13:42          196.662 gold2-bg.bmp
24.04.2002  16:48            1.954 gold-wht-btn-sm.bmp
15.02.2002  23:01            3.020 b_Toolbar.gif
24.04.2002  15:38            1.123 transparent-btn-lg.gif
24.04.2002  15:39              932 transparent-btn-sm.gif
15.02.2002  23:18            5.823 b_ITBrowse.gif
12.04.2002  14:25           34.144 b_Frame.gif
```

```
15.02.2002  23:23             4.269 b_ITForm.gif
24.04.2002  15:36               984 transparent-btn.gif
18.01.2002  13:14             2.238 b_ITsearch.ico
15.02.2002  18:41             2.238 b_ITPrior.ico
17.01.2002  01:08             2.238 b_ITprevpage.ico
16.02.2002  10:43             2.238 b_ITok.ico
17.01.2002  01:07             2.238 b_ITnextpage.ico
15.02.2002  19:05             2.238 b_ITNext.ico
15.02.2002  19:40             2.238 b_ITmark.ico
17.01.2002  01:07             2.238 b_ITlast.ico
15.02.2002  19:45             2.238 b_ITinsert.ico
15.02.2002  21:35             2.238 b_IThelporig.ico
16.02.2002  11:14             2.238 b_IThelp.ico
17.01.2002  01:05             2.238 b_ITfirst.ico
15.02.2002  19:38             2.238 b_ITedit.ico
15.02.2002  21:32             2.238 b_ITditto.ico
15.02.2002  19:47             2.238 b_ITdelete.ico
18.02.2002  12:41             2.238 b_ITclose.ico
16.02.2002  10:50             2.238 b_ITcancel.ico
18.01.2002  13:21             2.238 b_ITnoprint.ico
              56 File(s)      2.396.877 bytes
```

Files in: 3rdParty\LibSrc

```
Date        Time               Size Filename
-------------------------------------------------------
01.08.2002  21:03             8.374 ITMBClass.clw
01.08.2002  18:01             2.510 ITMBClass.inc
               2 File(s)         10.884 bytes
```

Files in: 3rdParty\Template

```
Date        Time               Size Filename
-------------------------------------------------------
08.08.2002  14:37            19.902 ITMagicButtons.tpl
08.08.2002  15:28            19.777 ITMagicButtonsC.tpl
9 File(s)         39.679 bytes
```

Files in: 3rdParty\Tools\ITInstall

```
Date        Time               Size Filename
-------------------------------------------------------
02.08.2002  12:52           790.528 itutil.exe
02.08.2002  12:57               514 itmagicbuttons.itc
               2 File(s)        791.042 bytes
```