Icetips Magic Entries

Add a touch of Magic to your applications

Table of contents

Welcome

Welcome to the Icetips Magic Enries. This product is the results of our Magic Buttons, which have become quite popular in the Clarion world.

Introduction

The Icetips Magic Entries are a template supported technology that creates boxes behind entry fields, makes the entry field transparent and the results are a flatter, more attractive window design, than the three dimensional controls sometimes allow. The template calls methods in our MagicEntry class which does all the work.

There is a Global Extension template, which sets up the default Class name and all the defaults that are used. It also does some housekeeping to make sure that the classes are linked in correctly for exes and dlls.

There is also a Procedure Extension template which allows you to set properties for the entries, as well as include, exclude and override individual entries. You can take care of everything yourself if you want to and make the calls to the class methods in embedded code.

This product is compatible with, and has been tested with, Clarion 4B, Clarion 5B and Clarion 5.5 and both ABC and Clarion template chains are supported.

Installation

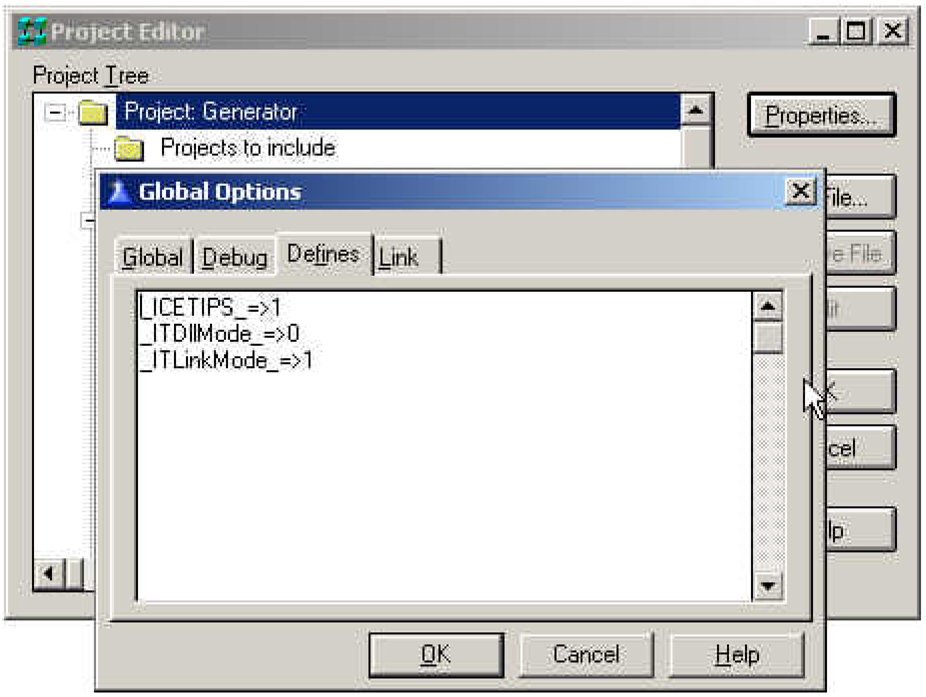
The Icetips Magic Entries are provided as a single ITMagicEntries.exe install set. It will install the template file and any additional images that we may provide with it into a single directory of your choice and update the redirection file (RED file) of your Clarion IDE with the path to the class files.

The install program should register the templates automatically. However, if that fails, please register the templates manually. Run the Clarion environment and select "Setup|Template Registry" from the main menu in Clarion and click on the [Register] button. The install program will place the template files into the 3rdParty\Template directory under the Clarion directory.

Adding the Magic Entries for the first time

The Magic Entries 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 Entries class when it it linked into the project. These 3 defines are:

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



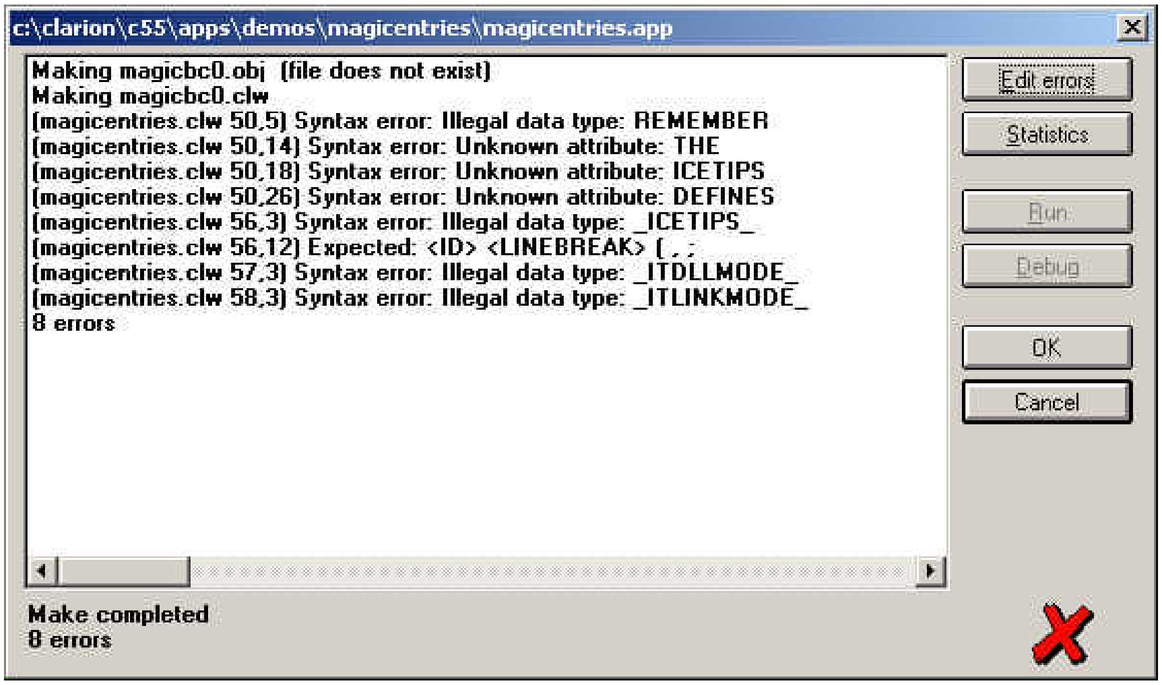
```
_ICETIPS_=>1
_ITDllMode_=>0
_ITLinkMode_=>1
```

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 Entries global extension template.

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

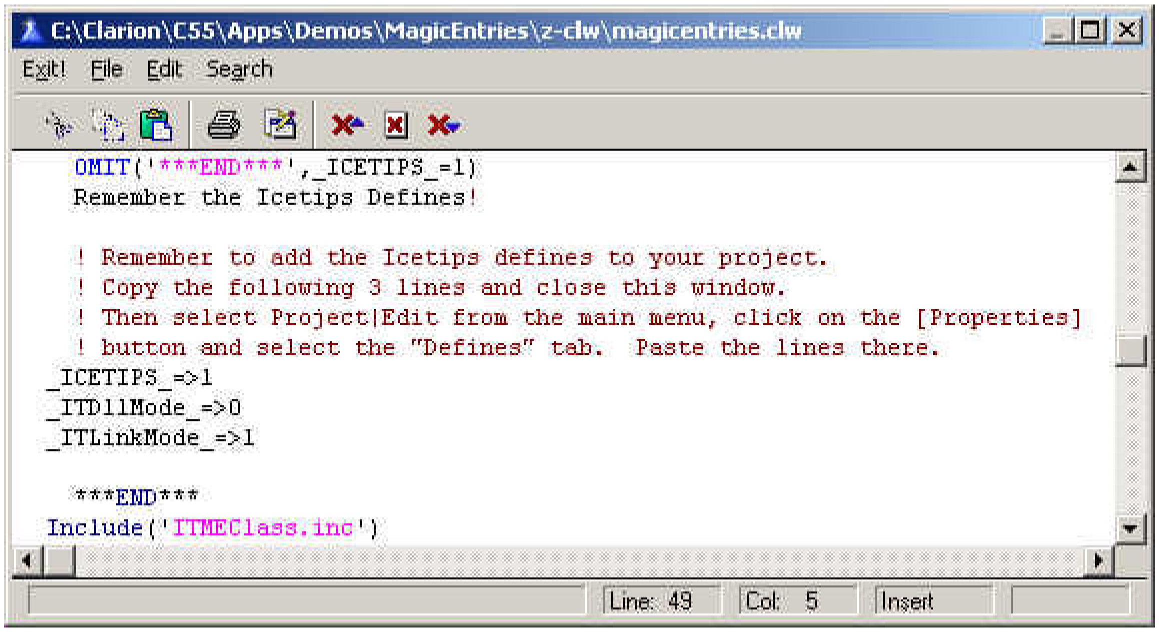

c:\clarion\c55\apps\demos\magicentries\magicentries.app

```
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

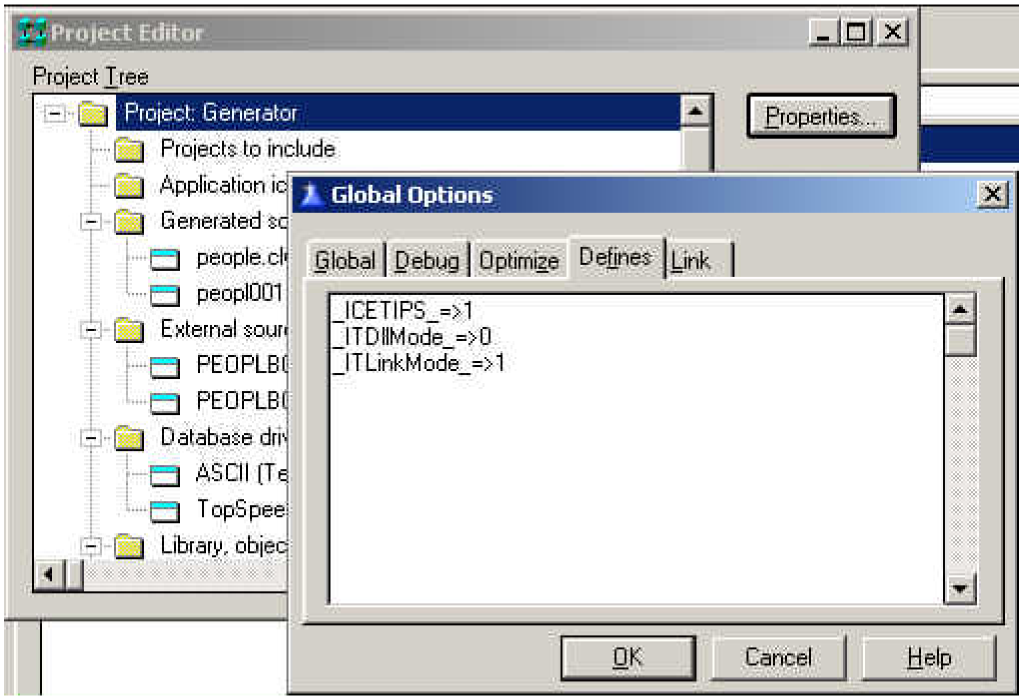
```
    OMIT('***END***',_ICETIPS_=1)
    Remember the Icetips Defines!

     ! 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:



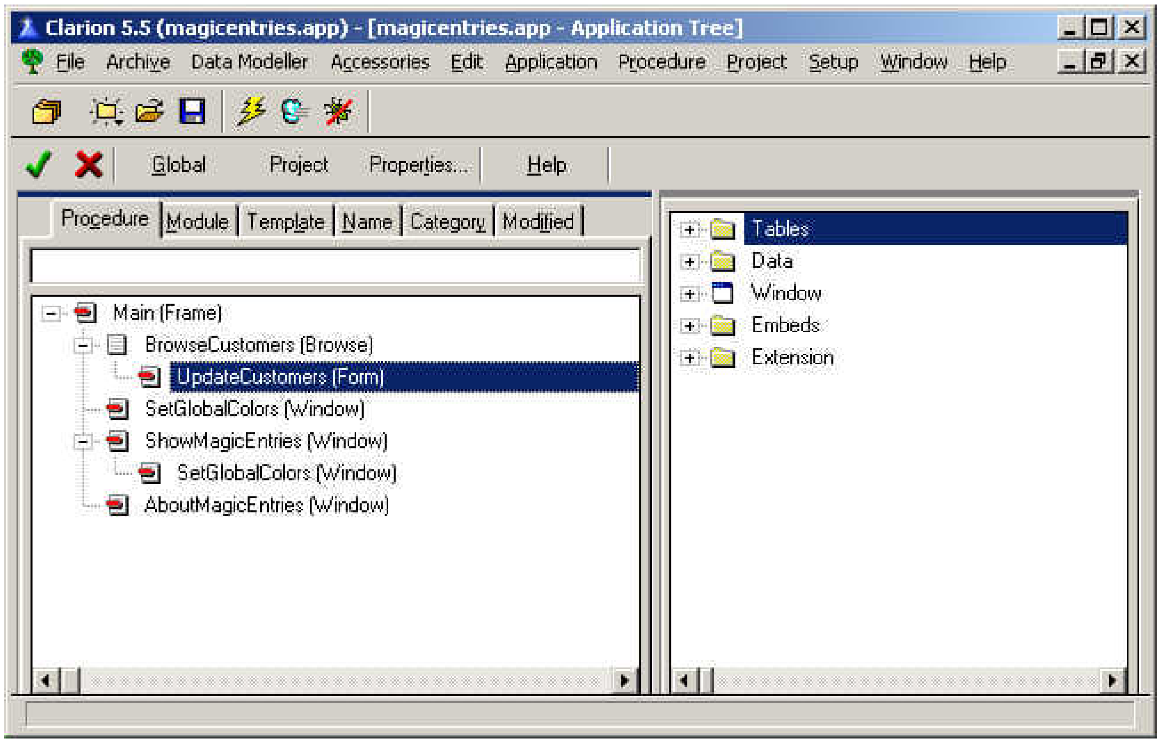
```
_ICETIPS_=>1
_ITDllMode_=>0
_ITLinkMode_=>1
```

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.

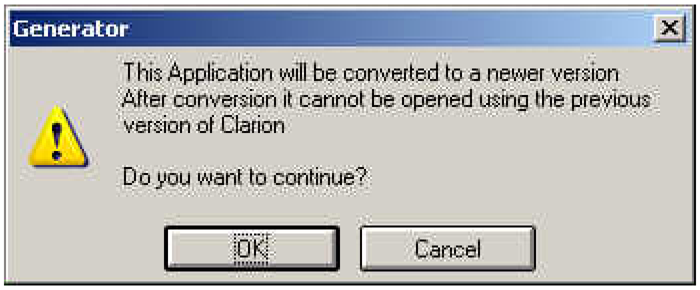
Demo application

The product ships with a rather simple demo application which you can find in the 3rdParty\Examples\ITMagicEntries directory. The Demo is a Clarion 4B, ABC


Clarion 5.5 (magicentries.app) - [magicentries.app - Application Tree]
File Archive Data Modeller Accessories Edit Application Procedure Project Setup Window Help
Global Project Properties... Help
Procedure Module Template Name Category Modified
Main (Frame)
BrowseCustomers (Browse)
UpdateCustomers (Form)
SetGlobalColors (Window)
ShowMagicEntries (Window)
SetGlobalColors (Window)
AboutMagicEntries (Window)
Tables
Data
Window
Embeds
Extension

The demo application open in C5.5H

application and should open and compile without any problems in any flavour of Clarion since Clarion 4B. It has been compiled and run and tested in Clarion 4B, Clarion 5B, Clarion 5.5Gold and Clarion 5.5H (5508)


Generator

This Application will be converted to a newer version
After conversion it cannot be opened using the previous version of Clarion

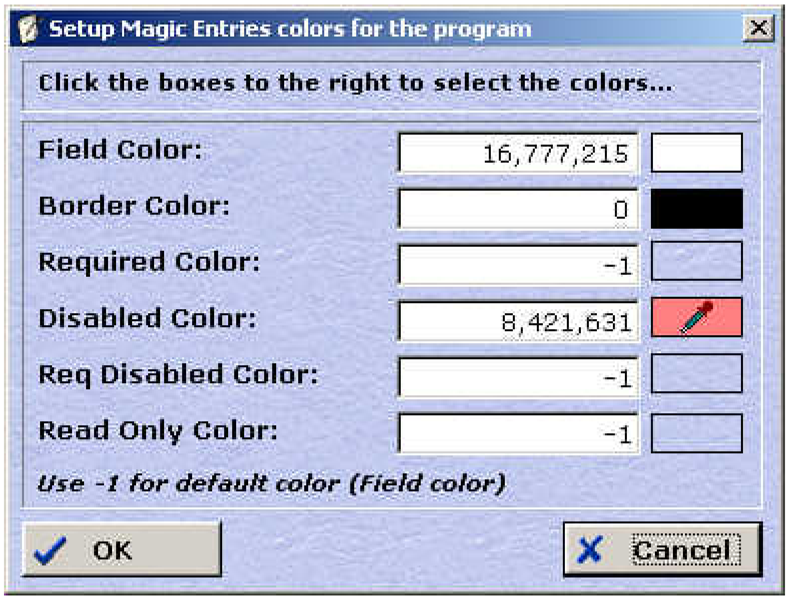
Do you want to continue?

Opening demo in C5.5

When you open the demo application in C5.5 you will get a message telling you that you are opening the application in a more recent version of clarion and prompts you to decide if you want to continue or not.

The demo application is not exhaustive of all the options for the Magic Entries. Some options can only be set through the classes (special attribute border colors for example, i.e. special border color for required fields or read-only fields). Example of setting the ReadOnly border can be found in the **UpdateCustomers** procedure in the WindowManager.Init method.

The **SetGlobalColors** shows how to select colors. It shows how you can use regions to select color and also allow for entries. To use the Field color for any of the special attributes, simply use -1. Normally the Magic Entries would only be applied to forms or windows, but you can apply it to any procedure. In the **SetGlobalColors** the cursor changes to a dropper over the regions to the right where you click to select the colors.



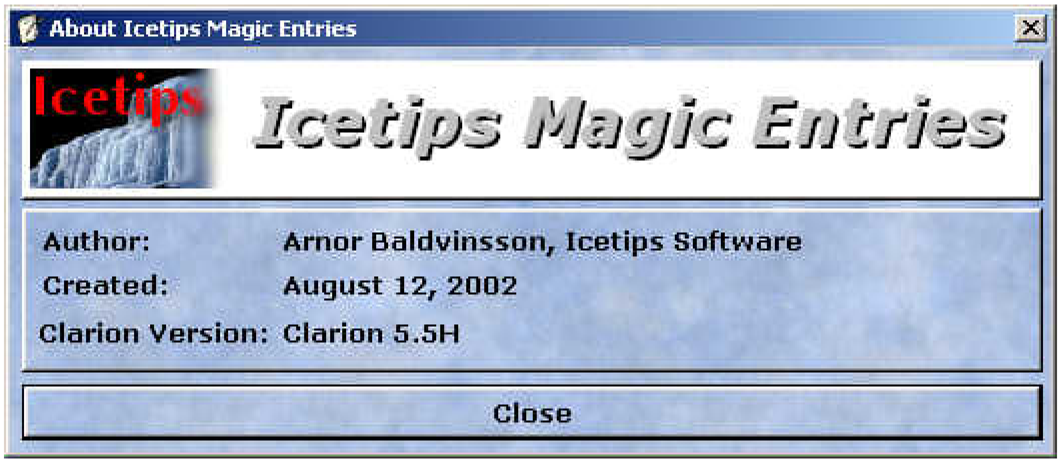
Click the boxes to the right to select the colors...

Field Color:	16,777,215
Border Color:	0
Required Color:	-1
Disabled Color:	8,421,631
Req Disabled Color:	-1
Read Only Color:	-1

Use -1 for default color (Field color)

SetGlobalColors procedure

The **AboutMagicEntries** procedure shows how to use it to retrieve the Clarion build which the application was compiled with. This can come in handy if you have code that



Author:	Arnor Baldvinsson, Icetips Software
Created:	August 12, 2002
Clarion Version:	Clarion 5.5H

AboutMagicEntries window showing the Clarion version used to compile it

needs to be Clarion version depended.

If you only want to gain access to the Clarion build, simply apply the Procedure Extension template to it and at any point after WindowManager.Init, priority 9900 you can use it code like this one, taken from the AboutMagicEntries procedure:

```
If ITME.ClarionBuild < 5500
  Case ITME.ClarionBuild
  Of 4000
    Loc:ClarionBuild = 'Clarion 4 Gold'
  Of 4001
```

```
    Loc:ClarionBuild = 'Clarion 4A'
  Of 4002
    Loc:ClarionBuild = 'Clarion 4B'
  Of 5000
    Loc:ClarionBuild = 'Clarion 5 Gold'
  Of 5001
    Loc:ClarionBuild = 'Clarion 5A'
  Of 5002
    Loc:ClarionBuild = 'Clarion 5B'
  End
Else
  S" = Chr((ITME.ClarionBuild-5500) + 64)
  Loc:ClarionBuild = 'Clarion 5.5' & Clip(S")
End
```

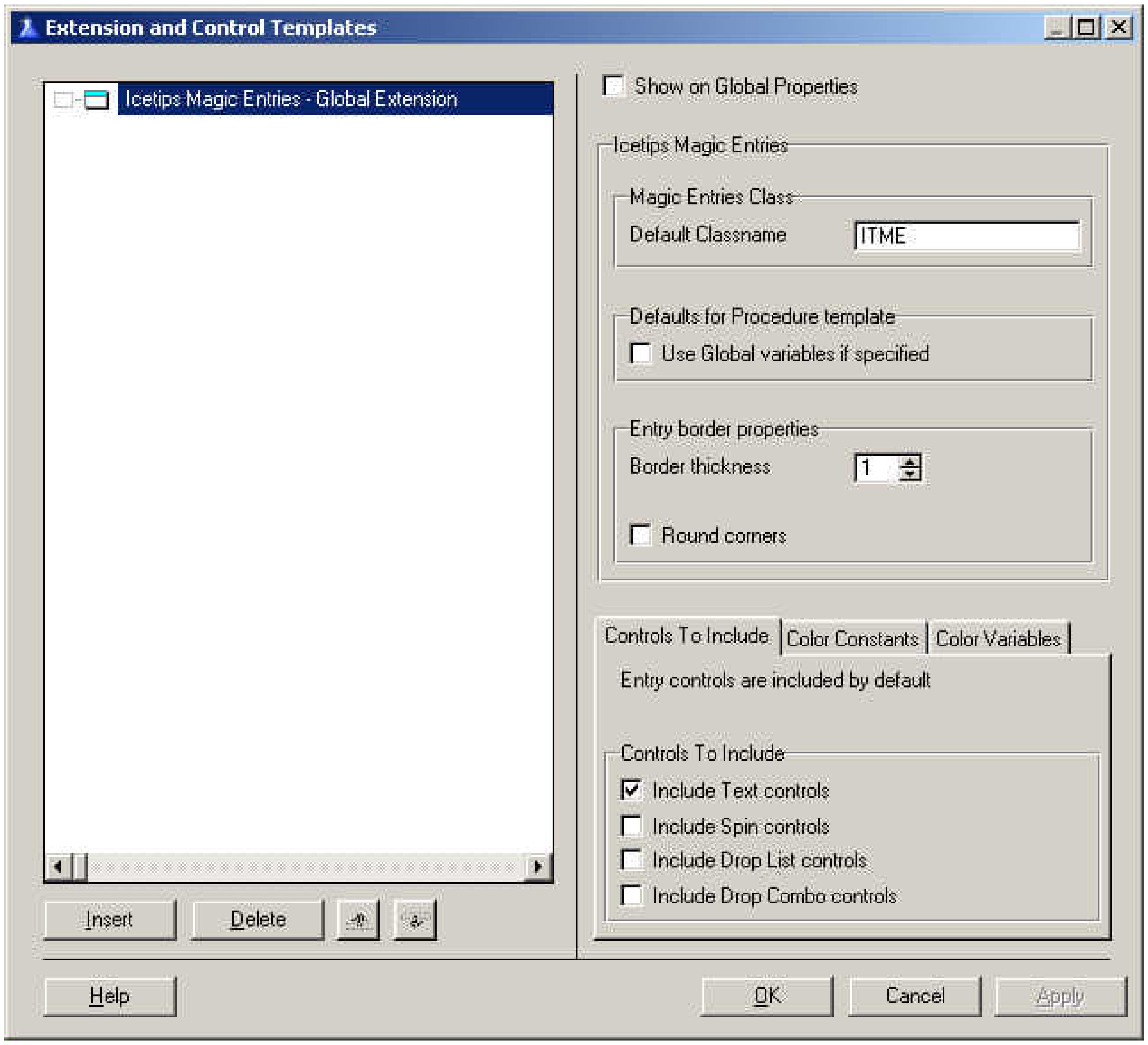
The build is returned in numbers like 4000 for Clarion 4.0 Gold, 4002 for Clarion 4B, 5500 for Clarion 5.5Gold and 5508 for Clarion 5.5H.

Clarion version related information

In all builds before Clarion 5.5A there is a bug in the Clarion Runtime Library which makes text boxes not update correctly when they have the transparent attribute set. Please refer to the Compatibility and Technical Issues chapter on page 34 for more information. Due to this problem, if you open the demo app in any release of Clarion prior to Clarion 5.5A, you will not be able to set the "Include Text Boxes" check on the global or procedure extension templates. If you open the demo in Clarion 5.5A or later, you must remember to check that on the ShowMagicEntries procedure so you can see how the textbox looks like with the Magic Entries applied to it.

Global Extension Template

In order for you to use the Magic Entries templates, first of all you need to populate the Icetips Magic Entries Global Extension template into your app. Click on the [Global] button in the Clarion IDE


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

Show on Global Properties

Icetips Magic Entries

Magic Entries Class

Default Classname: ITME

Defaults for Procedure template

Use Global variables if specified

Entry border properties

Border thickness: 1

Round corners

Controls To Include | Color Constants | Color Variables

Entry controls are included by default

Controls To Include

[✓] Include Text controls

Include Spin controls

Include Drop List controls

Include Drop Combo controls

Applying the Global Extension template

The prompts are documented in the following text. Most of these prompts are also present on the procedure extension template as well.

Default Classname

This is the default name used for the Icetips Magic Entries class as it is instanciated in each procedure. The classname defaults to ITME. You can use whatever you want, but a short name is easier to use if you write code to take advantage of the methods directly.

Use Global variables if specified

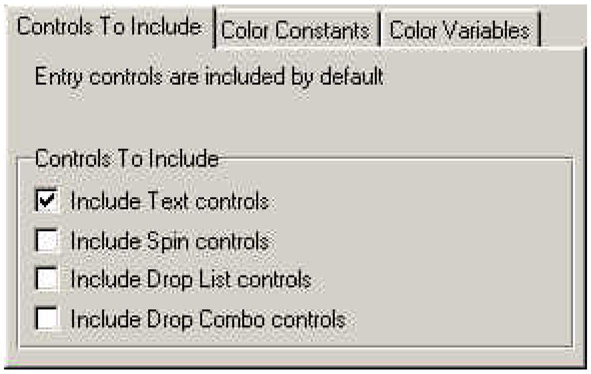
Checking this means that variables are used as defaults in the procedure extension template when it is populated to procedures. If this is unchecked the variables defined in the Global extension are not used. The default value is False, so make sure that if you want to use the variables at procedure level by default, that you check this. This only affects the procedure extension when it is applied. You can select the same variables in the Procedure extension if you want to, this simply selects them for you when you apply the Procedure extension template.

Border thickness

This is the border thickness that is used around each entry control. The default value is 1. This is used as a default value in the procedure template.

Round corners

This specifies if the boxes that are drawn behind each entry control should have round corners or not. The default is false. This is used as default value in the procedure template.



Entry controls are included by default

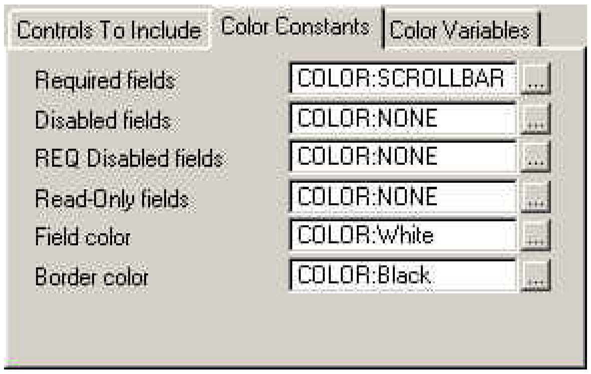
Controls To Include

- [✓] Include Text controls
- [ ] Include Spin controls
- [ ] Include Drop List controls
- [ ] Include Drop Combo controls

Controls to Include

Controls to Include

This is used to determine if certain control types should be included in the Magic Entry list of controls or not. Please note that DropLists, although fully supported, do not work correctly in Clarion because the transparent attribute changes the fontcolor as well. Also they can not be set transparent at runtime. Text boxes are included by default.

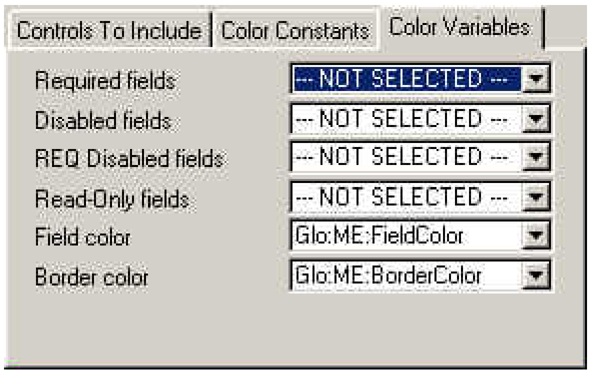


Required fields	COLOR:SCROLLBAR
Disabled fields	COLOR:NONE
REQ Disabled fields	COLOR:NONE
Read-Only fields	COLOR:NONE
Field color	COLOR:White
Border color	COLOR:Black

Setting Color Constants

Color Constants

You can select what colors to use by default for Required fields, Disabled Fields, Required Disabled fields and Read only fields, as well as normal fields and what the border color should be. Note that COLOR:None (or -1) are used to default to the normal color.



Required fields --- NOT SELECTED ---
Disabled fields --- NOT SELECTED ---
REQ Disabled fields --- NOT SELECTED ---
Read-Only fields --- NOT SELECTED ---
Field color Glo:ME:FieldColor
Border color Glo:ME:BorderColor

Selecting Color Variables

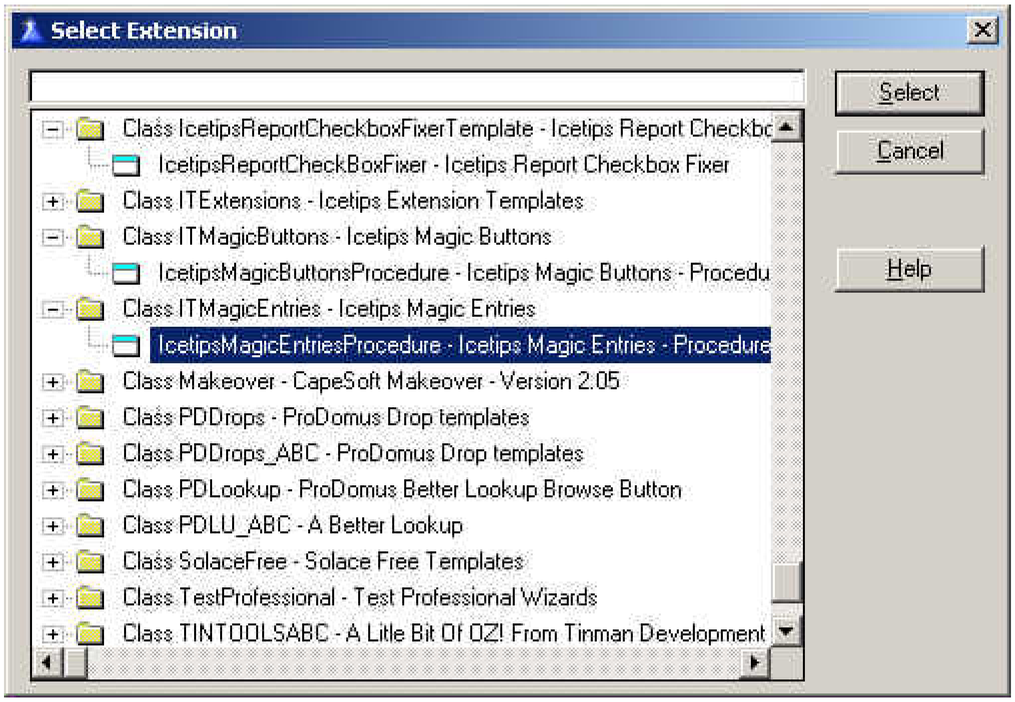
Color Variables

You can alternatively use variables to specify colors. They take priority over color constants, i.e. if you specify a variable for Requried fields, the procedure templates will default to use the variable when generating code. Please note the "Use Global variables if specified" setting above. If no variable has been selected the text --- NOT SELECTED --- will appear in the dropdown. Note that global variables and file fields can not be selected into global templates, so we have made a droplist of the global variables available. If you want to use variables for the colors, these variables must exist in the Global Data as you can not add them here.

Procedure Extension Template

The second step is to add the Icetips Magic Entries Procedure Extension template to any procedures in your app that you want to use the Icetips Magic Entries. Normally this would be forms and windows where you have entry fields for variables. You can apply this template anywhere where you need it. If you have handcoded procedures, you can simply add the method calls into your code and it will work fine. See the class reference at the end of this document for more information.

To apply the Procdure Extension template, open the procedure properties and click on the [Extensions] button. Click on the [Insert] button and then pick the IcetipsMagicEntriesProcedure template out of the list.



Class IcetipsReportCheckboxFixerTemplate - Icetips Report Checkbc
IcetipsReportCheckBoxFixer - Icetips Report Checkbox Fixer
Class ITExtensions - Icetips Extension Templates
Class ITMagicButtons - Icetips Magic Buttons
IcetipsMagicButtonsProcedure - Icetips Magic Buttons - Procedu
Class ITMagicEntries - Icetips Magic Entries
IcetipsMagicEntriesProcedure - Icetips Magic Entries - Procedure
Class Makeover - CapeSoft Makeover - Version 2.05
Class PDDrops - ProDomus Drop templates
Class PDDrops_ABC - ProDomus Drop templates
Class PDLookup - ProDomus Better Lookup Browse Button
Class PDLU_ABC - A Better Lookup
Class SolaceFree - Solace Free Templates
Class TestProfessional - Test Professional Wizards
Class TINTOOLSABC - A Litle Bit Of OZ! From Tinman Development

Select the Procedure Extension template

When the Procedure Extension template pops up, everything has been filled in and you do not need to do anything more if the default settings are exactly what you need. However, if you need to modify any of the settings, you can do that here as well as override, include and exclude entries from the list of entry controls that will be affected by the template. We will go through each setting on the next few pages and describe what they do.

Class name

By default the classname used for the Icetips Magic Entries is ITME. You can change this to anything you want as long as it is a valid Clarion label, i.e. no spaces are allowed etc.

Please note that these classes are NOT ABC compatible classes and will therefor not show up in the ABC class viewer.

Border thickness

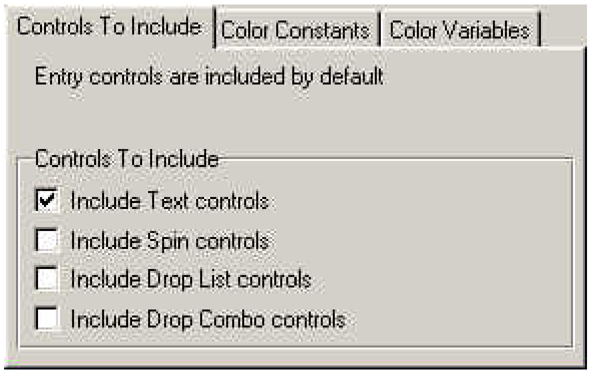
This is the border thickness that is used around each entry control. The default value is 1.

Round corners

This specifies if the boxes that are drawn behind each entry control should have round corners or not. The default is false.

Controls to Include

This is used to determine if certain control types should be included in the Magic Entry list of controls or not. Please note that DropLists, although fully supported, do not work correctly in Clarion because the transparent attribute changes the fontcolor as well. Also they can not be set transparent at runtime. Text boxes are included by default.



Entry controls are included by default

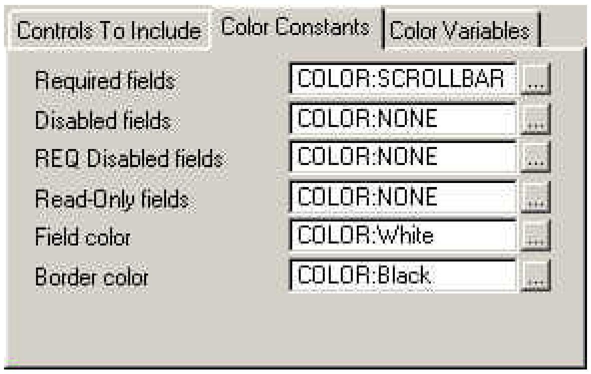
Controls To Include

- [✓] Include Text controls
- [ ] Include Spin controls
- [ ] Include Drop List controls
- [ ] Include Drop Combo controls

Controls to Include

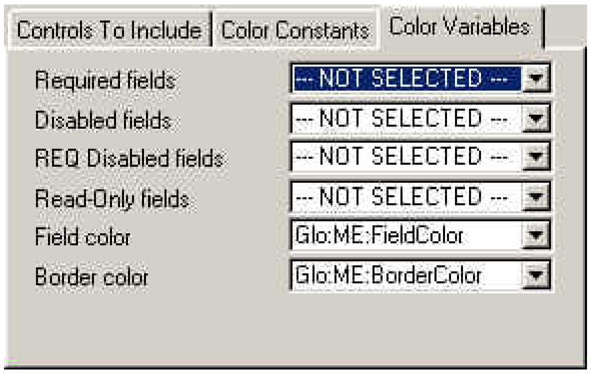
Color Constants

You can select what colors to use by default for Required fields, Disabled Fields, Required Disabled fields and Read only fields, as well as normal fields and what the border color should be. Note that COLOR:None (or -1) are used to default to the normal color.



Required fields	COLOR:SCROLLBAR
Disabled fields	COLOR:NONE
REQ Disabled fields	COLOR:NONE
Read-Only fields	COLOR:NONE
Field color	COLOR:White
Border color	COLOR:Black

Setting Color Constants



Required fields	--- NOT SELECTED ---
Disabled fields	--- NOT SELECTED ---
REQ Disabled fields	--- NOT SELECTED ---
Read-Only fields	--- NOT SELECTED ---
Field color	Glo:ME:FieldColor
Border color	Glo:ME:BorderColor

Selecting Color Variables

Color Variables

You can alternatively use variables to specify colors. They take priority over color constants, i.e. if you specify a variable for Requried fields, the procedure templates will default to use the variable when generating code. Please note the "Use Global variables if specified" setting above. If no variable has been selected the text --- NOT SELECTED --- will appear in the dropdown. Note that global variables and file fields can not be selected into global templates, so we have made a droplist of the global variables available. If you want to use variables for the colors, these variables must exist in the Global Data as you can not add them here.

Required Fields

Select the color that is used for Required fields. This is used on all fields that have the Required attribute when the window is opened. COLOR:None means that it will use the Field color.

Disabled Fields

Select the color that is used for Disabled fields. This is used on all fields that are disabled when the window is opened. COLOR:None means that it will use the Field color.

REQ Disabled Fields

Select the color that is used for Required and Disabled fields. This is used on fields that have both the Required attribute and are disabled when the window is opened. COLOR:None means that it will use the Field color.

Read-Only Fields

Select the color that is used for Read-Only fields. This is used on fields that have the Read-Only attribute set when the window is opened.

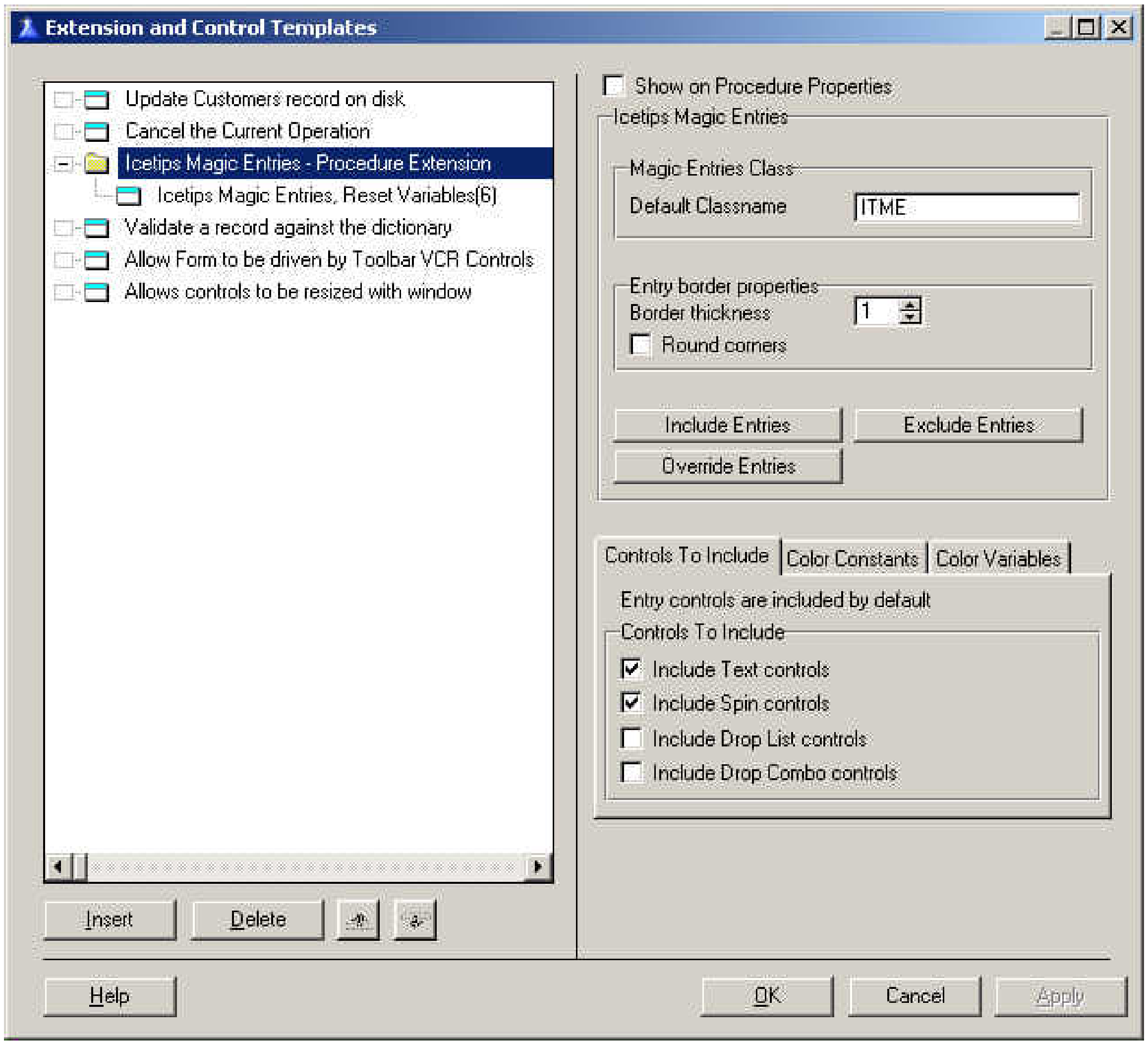
When these attributes are applied in the Magic Entries class, the Read-only takes priority over Required. I.e. if Read-Only fields are yellow and Required fields are red, a field that is both read-only and required is going to show up as yellow. It does not make much sense to have a field show up as required when it is read-only!

Field color

Select the field color that is used for all fields except those that are disabled, required or read-only and the color is set for these attributes.

Border color

Select the color for the border of the entry field. This default to black.



Extension and Control Templates

Update Customers record on disk
Cancel the Current Operation
Icetips Magic Entries - Procedure Extension
Icetips Magic Entries, Reset Variables(6)
Validate a record against the dictionary
Allow Form to be driven by Toolbar VCR Controls
Allows controls to be resized with window

☐ Show on Procedure Properties

Icetips Magic Entries

Magic Entries Class

Default Classname: ITME

Entry border properties

Border thickness: 1

☐ Round corners

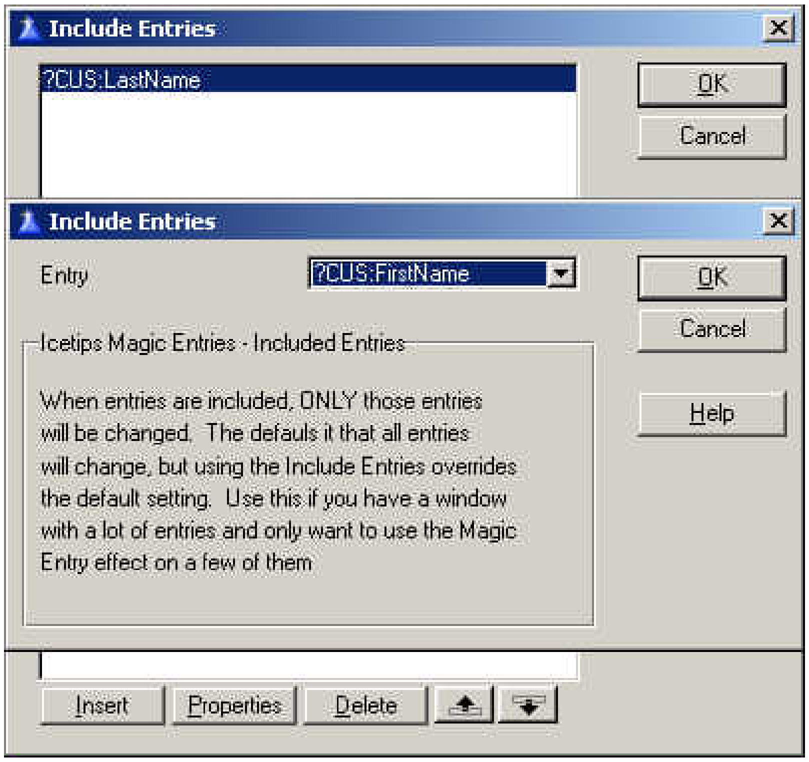
Entry controls are included by default

Controls To Include

☑ Include Text controls
☑ Include Spin controls
☐ Include Drop List controls
☐ Include Drop Combo controls

Procedure Extension Template

Include Entries



Include Entries

?CUS:LastName

Include Entries

Entry ?CUS:FirstName

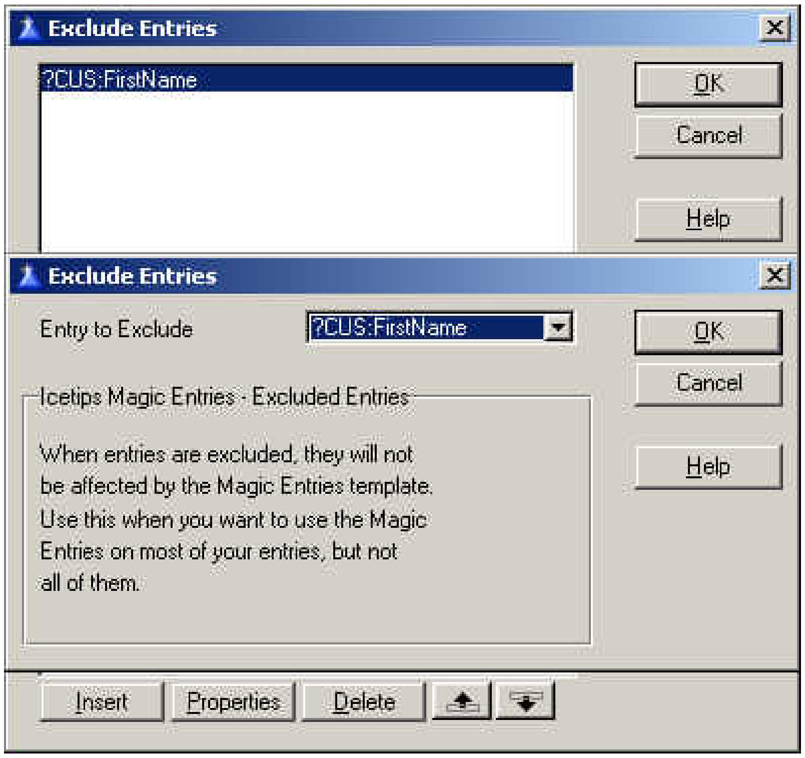
Icetips Magic Entries - Included Entries

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

Including Entries

By default, all entry controls on the window are included in the list of controls to apply the Magic Entries to. If you select to include text controls, all text controls are also included. The Include Entries can be used to override this. If you use this option, only the controls you add here are added to the list of controls. You would normally use this if you only want to apply the Magic Entries to a few controls out of many on your window.

Note that if you use the Override Entries option, it will add the controls that you override to the list of controls that are changed by the template, even if they are not in the list of Included Entries.



Exclude Entries

?CUS:FirstName

Exclude Entries

Entry to Exclude ?CUS:FirstName

Icetips Magic Entries - Excluded Entries

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

Excluding Entries

Exclude Entries

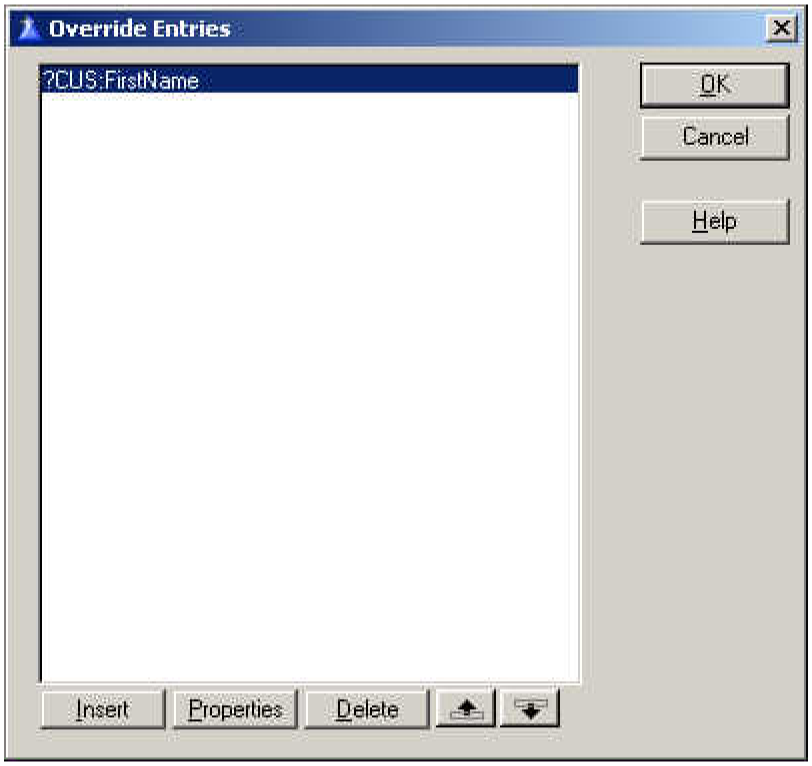
Entries can be excluded from the list of controls that the Magic Entries are applied to.

This would be most useful when you want to include most of your entry controls, but you want to exclude some. Obviously if the excluded entries were not included in the first place, noting is going to happen to them.

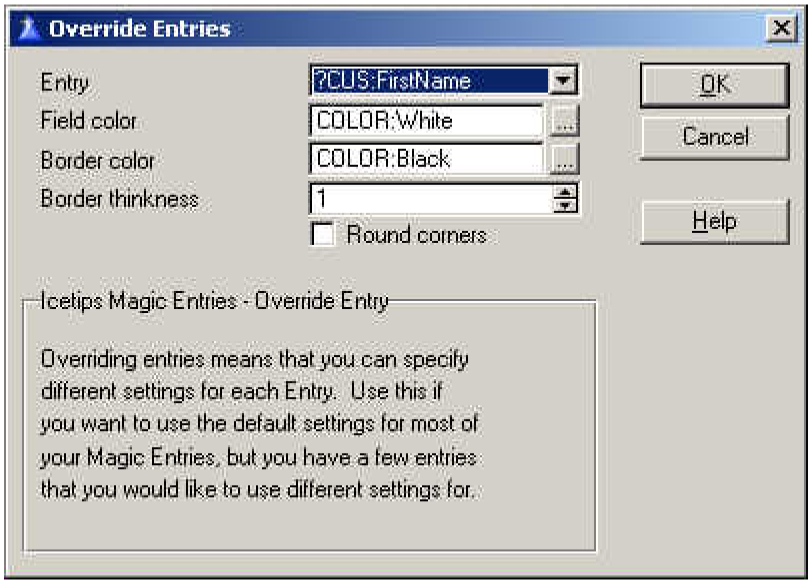
Override Entries

This option allows you to override the color settings for individual entry controls, allowing you to customize your window the way you want it to look like. All entry controls are available in this list and if they are not included, either implicitly or explicitly, they will be included if they are overridden.

For each entry you can override the field color, border color, border thickness and set it to use round corners or not.



Override entries



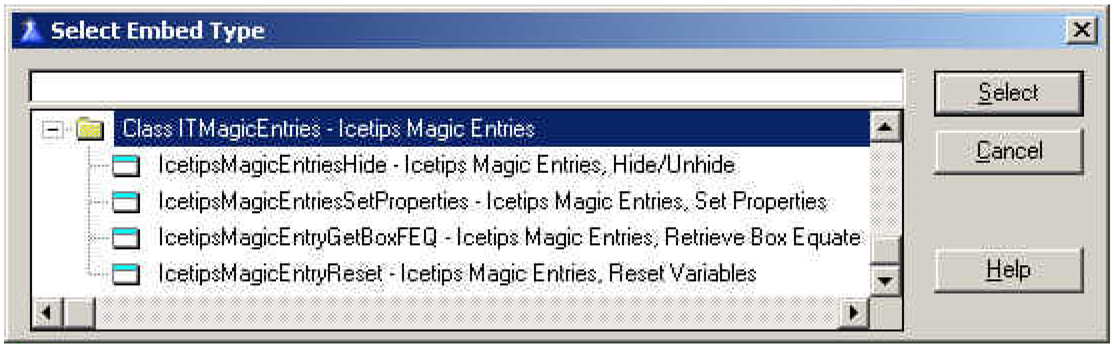
Entry: ?CUS:FirstName
Field color: COLOR:White
Border color: COLOR:Black
Border thinkness: 1

Overriding entries means that you can specify different settings for each Entry. Use this if you want to use the default settings for most of your Magic Entries, but you have a few entries that you would like to use different settings for.

Overriding an individual entry

Code templates

The Icetips Magic Entries includes 4 code templates to make life easier if you need to manipulate the image control for a particular entry. These code templates allow you to hide/unhide the image or both the image and the entry. This section will show how each code template operates. Also please refer to the class reference at the end of this


Select Embed Type

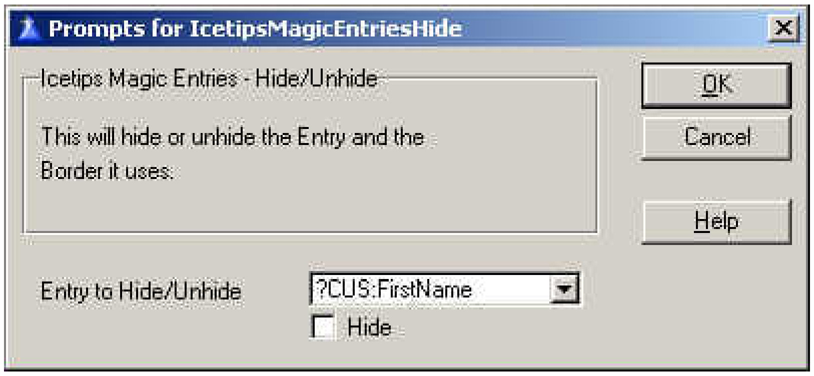
Class ITMagicEntries - Icetips Magic Entries
IcetipsMagicEntriesHide - Icetips Magic Entries, Hide/Unhide
IcetipsMagicEntriesSetProperties - Icetips Magic Entries, Set Properties
IcetipsMagicEntryGetBoxFEQ - Icetips Magic Entries, Retrieve Box Equate
IcetipsMagicEntryReset - Icetips Magic Entries, Reset Variables

Selecting any of the Icetips Magic Entries code templates

document for comprehensive documentation about each method in the classes and how to use them.

Hide/UnHide

When you hide an entry that uses the Icetips Magic Entries, you need to hide the image that is used to create the entry's background. This code template allows you to do that very easily. Select the entry control you want to hide or unhide. If you want to hide it, check the Hide checkbox and if you want to unhide the entry, leave the Hide checkbox unchecked. Alternatively you can use the HideEntry method passing the entry label to it, like ITME.HideEntry(?CUS:FirstName) or ITME.UnhideEntry(?CUS:FirstName) to unhide it.


Prompts for IcetipsMagicEntriesHide

Icetips Magic Entries - Hide/Unhide

This will hide or unhide the Entry and the Border it uses.

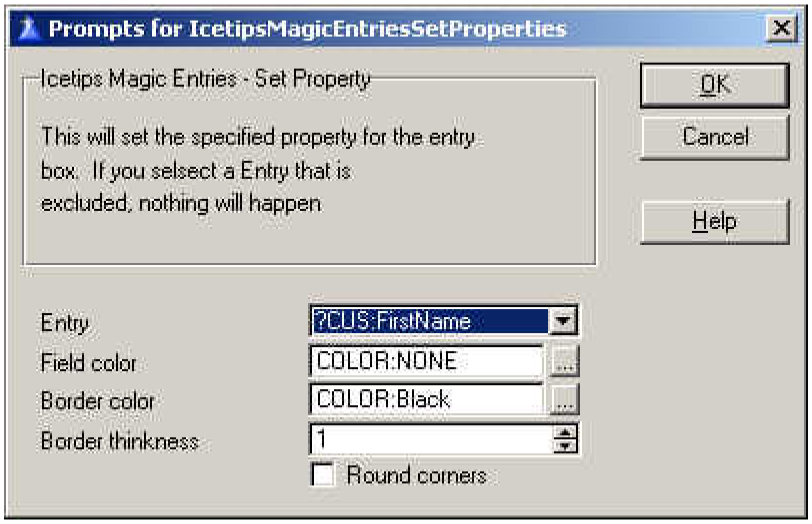
Entry to Hide/Unhide: ?CUS:FirstName

Hide [ ]

Selecting a entry control to hide/unhide with the image

SetProperties

This template allows you to set whatever property you want for the entry and/or the image behind it. Select the entry which image you want to set properties for, select the field and border colors, the border thickness and if the border should have round or sharp corners.


Prompts for IcetipsMagicEntriesSetProperties

Icetips Magic Entries - Set Property

This will set the specified property for the entry box. If you selsect a Entry that is excluded, nothing will happen

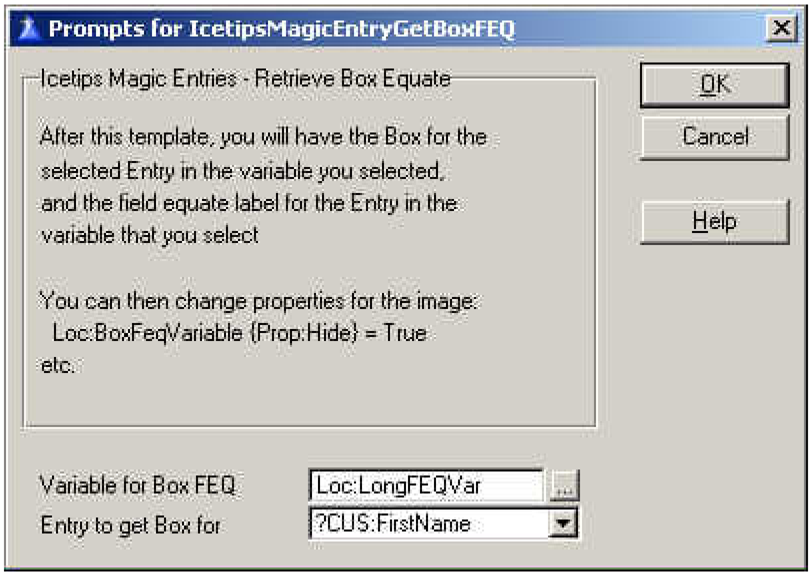
Entry: ?CUS:FirstName
Field color: COLOR:NONE
Border color: COLOR:Black
Border thinkness: 1
☐ Round corners

Using the SetProperties code template

This setting will generate the following code:

```
ITME.SetColor(?CUS:FirstName, 0, -1,1, 0)
```

GetImageFEQ



Prompts for IcetipsMagicEntryGetBoxFEQ

Icetips Magic Entries - Retrieve Box Equate

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

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

Variable for Box FEQ: Loc:LongFEQVar

Entry to get Box for: ?CUS:FirstName

Retrieving the Field Equate label for the Entry box image

This code template will retrieve the Field EQuate (FEQ) label of the entry box image, so you can modify it seperately by code. The Entry FEQ is stored in the variable you select, which must be a LONG variable to hold the value for the equate.

This will generate the following code:

```
If Loc:LongFEQVar.RetrieveEntry(?CUS:FirstName)
  Loc:LongFEQVar = ITME.Entries.BoxFEQ
End
```

You can now apply whatever properties you want to the image by using:

```
Loc:LongFEQVar {Prop:FillColor} = COLOR:Red
```

As an example to change this particular entry fields border to Red.

Entry Reset

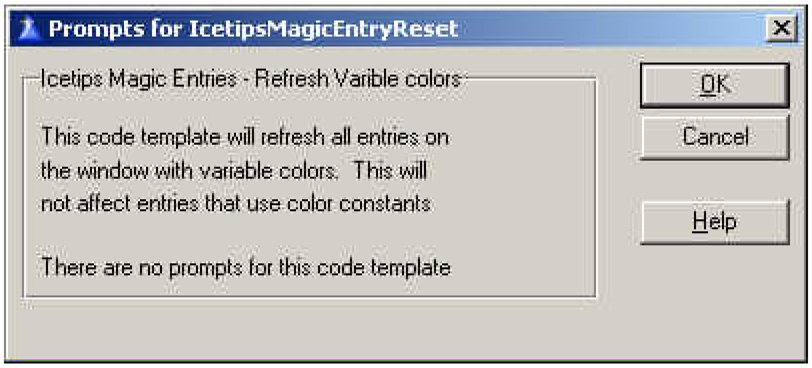
This code template resets all entry fields, except overridden fields to use the correct values for colors etc. This is required after values of variables that define colors has changed. For example:

```
Glo:RequiredColor = COLOR:Red
```

Then it would be appropriate to use this code template right after the variable has changed. This code template simply reassigns all color variables and calls the RefreshAll method. The same effects can be accomplished by using as an example:

```
ITME.DisabledFill    = Glo:DisabledColor
ITME.BoxBorder       = Glo:BorderColor
ITME.BoxFill         = Glo:FieldColor
ITME.RefreshAll
```

If you don't use variables for colors, you do not need to use this code template.


Prompts for IcetipsMagicEntryReset

Icetips Magic Entries - Refresh Varible colors

This code template will refresh all entries on the window with variable colors. This will not affect entries that use color constants

There are no prompts for this code template

Entry Reset code template

There are no prompts for this code template, just apply it. It does provide automatic access to all colors that use variables in the template, so it is handy to use because it knows what color properties to set and which properties it should leave alone.

Class Reference

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

Class Equates

FEQType ITEMIZE

```
FEQType      ITEMIZE(1),Pre()
Normal         EQUATE
Required       EQUATE
Read_Only      EQUATE
Disabled       EQUATE
ReqDisabled    EQUATE
             END
```

These itemized Equates are used to determine what type of entry field is being registered. The type is stored in the Entries queue.

Class Properties

Entries &MEFeqQ

The Entries property is a reference to a queue:

```
MEFeqQ       Queue,Type
EntryFEQ       Long  ! Label of the Entry control
BoxFEQ         Long  ! Label of the Box control
BoxBorder      Long  ! Border color
BoxBorderLine  Byte  ! Border thickness
BoxFill        Long  ! Fill color
BoxRound       Byte  ! Round corners (True/False)
EntryType      Byte  ! Type - FEQType equate value
ResetStatus    Byte  ! Reset possible (True/False)
             End
```

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

Initialized Byte

Set to True in the Init method. Used to determine if the class is initialized and to determine if images that have been created should be destroyed in the Kill method.

BoxBorderLine — Byte

The thickness in dialog units of the border line.

BoxRound — Byte

If True the box is created with round corners. If False, the corners are sharp.

BoxFill — Long

The fillcolor or background color on the entry box.

RequiredFill — Long

Fill or background color for entry fields that are required - have the REQ property set. Use -1 to use the default background color, BoxFill.

DisabledFill — Long

Fill or background color for entry fields that are disabled. Use -1 to use the default background color, BoxFill.

ReqDisabledFill — Long

Fill or background color for entry fields that are required but are also disabled. Use -1 to use the default background color, BoxFill.

ReadOnlyFill — Long

Fill or background color for entry fields that are Read-Only. Use -1 to use the default background color, BoxFill. Note that if the field is disabled or required, those property take priority over the Read-Only color.

BoxBorder — Long

The color of the box border line.

RequiredBorder — Long

Border for requires entries, -1 for BoxBorder

DisabledBorder — Long

Border for disabled entries, -1 for BoxBorder

ReqDisabledBorder — Long

Border for Requred, Disabled entries, -1 for BoxBorder

ReadOnlyBorder **Long**

Border for Read-Only entries, -1 for BoxBorder

Xoffset **Long**

Indicates the X offset of the box in relationship to the entry field. Negative values move the box to the left of the entry, positive to the right of the entry. This value is in pixels.

Yoffset **Long**

Indicates the Y offset of the box in relationship to the entry field. Negative values move the box up above the entry, positive move the box down below the entry. This value is in pixels.

Woffset **Long**

Indicates the width offset of the box in relationship to the entry field. Negative values make the box narrower than the entry field, positive values make it wider. This value is in pixels.

Hoffset **Long**

Indicates the height offset of the box in relationship to the entry field. Negative values make the box shorter than the entry field, positive values make it taller. This value is in pixels.

IncludeText **Byte**

Include Text controls

IncludeSpin **Byte**

Include Spin controls

IncludeDropList **Byte**

Include DropList controls

IncludeDropCombo **Byte**

Include DropCombo controls

ClarionBuild **Long**

Clarion release build. Contains for example 5500 for C5.5 Gold, 5501 for C5.5A etc.

Class Methods

Init **Procedure(Long pBoxBorder, Long pBoxFill, |**
Byte pBorderLine, Byte pBoxRound)

The Init method instanciates the Entries queue and sets several internal properties for use in other methods. It also sets the default offsets for the boxes, default colors for borders and default fills for all entry types.

SetOffsets **Procedure(Long pXOff, Long pYOff, Long pWOff, Long pHOff)**

This method sets the internal offsets with the values passed to it. If you want to change the offsets in the template generated code, the appropriate method would be to call SetOffsets after the ApplyWindowMagic method is called, followed by the SetAllBoxPos to reposition and resize the entry boxes. In handcoded projcect you can call this method right after the Init method to set the offsets.

RegisterWindow **Procedure(Byte pIncludeText)**

The RegisterWindow method registers all entry controls on the window and adds them to the Self.Entries queue. The pIncludeText parameter can be true or false and indicates if text boxes should be included in the control list or not.

RegisterEntry **Procedure(Long pFEQ,<Long pEntryFill>)**

The RegisterEntry registers a single entry control, optionally overriding the fill color. This method is called from the RegisterWindow to register each control.

UnregisterEntry **Procedure(Long pFEQ)**

The UnregisterEntry method deletes the specified entry control from the control queue if it has been registered. If it has not been registered, no action is taken.

SetEntryColor **Procedure(Long pColor, <Long pFEQ>)**

The SetEntryColor method can be used to set the color for one entry or all entries. If the pFEQ parameter is omitted, all controls are set to use the passed color, but if a control label is passed in the pFEQ parameter only that control is affected.

SetBorderColor **Procedure(Long pFEQ, Long pColor)**

The SetBorderColor method sets the border color of the pFEQ entry to pColor.

SetBorderRound **Procedure(Long pFEQ, Byte pRound)**

The SetBorderRound method sets the border for the pFEQ entry either round or not depending what the value of pRound is - either true or false.

SetBorderLine Procedure(Long pFEQ, Byte pLineWeight)

The SetBorderLine method sets the weight (thickness) of the border for the specified control (pFEQ)

SetResetStatus Procedure(Long pFEQ, Byte pResetStatus)

This method changes the Reset status of the specified entry. When an entry is overridden in the templates this is set to False (i.e. NO change). That means that when calling RefreshAll or RefreshEntry, overridden entries are skipped and not changed.

SetColor Procedure(Long pFEQ, Long pBoxBorder, | Long pBoxFill, Byte pBorderLine, Byte pBoxRound)

The SetColor method changes the settings for the specified entry control (pFEQ). The method can change the border color (pBoxBorder), the fill color (pBoxFill), the border line weight (pBorderLine) and if the entry box should have round or sharp corners (pBoxRound) This method can be used at any time after the ApplyWindowMagic method has been called.

ReSetProperty Procedure(Long pFEQ, | <Long pProperty>, <String pValue>)

The ReSetProperty is a very powerful method. It can be called at any time after the ApplyWindowMagic method.

First of all the method resets the color of the entry based on the current status of Prop:Req, Prop:Disable and Prop:Readonly. This is therefor appropriate to call after making changes to any of these properties for one or more entry fields.

Second this method can be used to set any properties for the entry box. This does not make any changes to the actual entry control, only to the box. Appropriate use for this would be:

```
ITME.ResetProperty(?CUS:Name,Prop:Hide,True)
```

SetAttributes Procedure

This method is only used internally by other methods and should not be used outside the class. It assigns the entry color settings to the correct values depending on if they use default values or not and what type of entry it is (normal, required etc.)

ApplyWindowMagic Procedure

The ApplyWindowMagic method calls the ApplyEntryMagic for all registered entries. This is the last method called by the template generated code when preparing the window.

ApplyEntryMagic — Procedure(Long pFEQ)

The ApplyEntryMagci method creates the box underneath the entry field, sets the position to match the entry field and sets all the color poperties for the box, so it appears according to how the colors have been set up and what properties are applicable to the entry field at that point.

RefreshEntry — Procedure(Long pFEQ)

This method updates the specified entry and redraws it in whatever color is being used for it. This is necessary after a variable color has changed. This method should be called after changing a color value (i.e. variable color) to refresh the specified entry control. This method is call for each control from the RefreshAll method which refreshes all controls.

RefreshAll — Procedure

This method updates all entry controls on the window. This is needed after values of colors have changed, for example a new color has been selected into a variable, so that the changes are reflected on the window.

RetrieveEntry — Procedure(Long pFEQ),Long

The RetrieveEntry method attempts to retrieves the record for the specified Entry control (pFEQ) and returns true or false depending on if the Entry control was found or not. An appropriate usage for this method is to check if the entry exists in the Class before applying any other methods to it. This method is used extensively in the other methods.

HideEntry — Procedure(Long pFEQ)

The HideEntry method hides the specified Entry field (pFEQ) and the associated box control.

UnhideEntry — Procedure(Long pFEQ)

The UnhideEntry method Unhides the specified Entry field (pFEQ) and the associated box control.

SetBoxPos — Procedure(Long pFEQ)

The SetBoxPos method retrieves the position of the specified Entry field (pFEQ) and then positions the associated box control so that it is correctly positioned under the entry control. This method uses the Offset values set by the SetOffset methods.

SetAllBoxPos — Procedure

The SetAllBoxPos resets the position of all box controls so they are positioned under the associated entry controls. The SetBoxPos is called from this method for every entry control that is registered. This method is appropriate to use after a window resize event has fired.

Kill Procedure

The Kill method destroys the box controls, frees the Entries queue and then disposes of it ensuring that nothing is left in memory. This method should be called before the window is closed.

Color Properties

Sometimes the color properties can be quite cumbersome and unintuitive. For example for the box control that we use in the Magic Entries, Prop:Fill and Prop:FillColor might seem to have the same meaning. Not so! Prop:Fill is the actuall fill color of the box control, while Prop:FillColor is the border color of the box. So if you wanted to change all the border colors to COLOR:Red, you could do it this way:

```
Loop I = 1 To Records(ITME.Entries)
  Get(ITME.Entries)
  ITME.Entries.BoxFEQ {Prop:FillColor} = COLOR:Red
End
```

Or you could do it using the methods:

```
Loop I = 1 To Records(ITME.Entries)
  Get(ITME.Entries)
  ITME.SetBorderColor(ITME.Entries.EntryFEQ,COLOR:Red)
End
```

Which makes this easier to remember!

Handcoding

Because of the classes, it is easy to use the Magic Entries in handcoded projects. It must be noted that the Magic Entries 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('itmeclass.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
```

Note that the templates also prompt you to add _ICETIPS_=>1 to the project. That is simply done so the compiler can check if the defines have been added or not. When handcoding you don't need that, but it will not harm in anyway.

Adding to the procedure

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

```
ITME   ITMagicEntryClass
```

This creates an object names ITME 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
ITME ITMagicEntryClass

 Code
 Open(W)
 Display
 ITME.Init(0,16777215,1,0)
 ITME.RequiredFill = -1
 ITME.DisabledFill = -1
 ITME.ReqDisabledFill = -1
 ITME.ReadOnlyFill = -1
 ITME.RegisterWindow(0)
 ITME.ApplyWindowMagic

 Accept
   ...
 End

 ITME.Kill
 Close(W)
```

The Magic Entry 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 entries are compatible with both the ABC and Clarion template chains, in Clarion versions 4, 5 and 5.5. Please note that there are some version dependent issues. Clarion 4 and Clarion 5 as well as Clarion 5.5Gold can not have transparent textboxes without problems. We decided not to support text boxes for the Magic Entries in these versions. There is also an issue with dropdown listboxes. They can not be set transparent at runtime and when they are set at design time, the font color of the non-drop part is set to the Selected font color. Normally that changes the font from black to white. Since we can not turn the dropdowns transparent, the box behind the control doesn't show through so the dropdowns don't ever get the Magic Entry look. All the code is there to support dropdown listboxes when it get's fixed. This bug does not affect the dropdown combos, which work fine.

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 while completing the project. Also note that the file dates are in dd.mm.yyyy format and file times are in hh:mm format

Files in: 3rdParty\Docs\ITMagicEntries

```
Date        Time               Size Filename
13.08.2002  12:00         1.807.9972 ITMagicEntries.pdf
```
(Note that this is not correct date/time/size)
```
               1 File(s)      1.666.934 bytes
```

Files in: 3rdParty\Examples\ITMagicEntries

```
Date        Time               Size Filename
13.08.2002  11:02           125.440 magicentries.app
12.08.2002  21:03               326 pointer.cur
12.08.2002  21:05               766 dropper.cur
13.08.2002  09:59             2.048 magicentries.dct
18.10.2001  17:36             5.269 itlogo.gif
12.04.2002  14:25            34.144 b_Frame.gif
15.02.2002  23:01             3.020 b_Toolbar.gif
23.04.2002  15:46             2.887 shop.gif
15.02.2002  21:35             2.238 b_IThelporig.ico
15.02.2002  19:45             2.238 b_ITinsert.ico
17.01.2002  01:07             2.238 b_ITlast.ico
15.02.2002  19:40             2.238 b_ITmark.ico
15.02.2002  19:05             2.238 b_ITNext.ico
17.01.2002  01:07             2.238 b_ITnextpage.ico
16.02.2002  11:14             2.238 b_IThelp.ico
16.02.2002  10:43             2.238 b_ITok.ico
17.01.2002  01:08             2.238 b_ITprevpage.ico
15.02.2002  18:41             2.238 b_ITPrior.ico
18.01.2002  13:14             2.238 b_ITsearch.ico
17.01.2002  01:05             2.238 b_ITfirst.ico
16.02.2002  16:06             2.238 refresh.ico
15.02.2002  19:38             2.238 b_ITedit.ico
20.08.1998  15:11               766 fontcol.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
12.08.2002  16:05             2.998 mail.ico
16.02.2002  10:50             2.238 b_ITcancel.ico
18.01.2002  13:21             2.238 b_ITnoprint.ico
13.08.2002  10:37           298.240 customers.tps
              30 File(s)        518.426 bytes
```

Files in: 3rdParty\LibSrc

```
Date        Time               Size Filename
12.08.2002  18:25            13.848 ITMEClass.clw
12.08.2002  18:24             3.809 ITMEClass.inc
               2 File(s)         17.657 bytes
```

Files in: 3rdParty\Template

```
Date        Time                Size Filename
12.08.2002  18:10             27.289 ITMagicEntriesC.tpl
12.08.2002  18:11             26.829 ITMagicEntries.tpl
               2 File(s)         54.118 bytes
```

Files in: 3rdParty\Tools\ITInstall

```
Date        Time                Size Filename
02.08.2002  12:52            790.528 itutil.exe
08.08.2002  17:36                294 magicentries.itc
               2 File(s)        790.822 bytes
```