Icetips Utilities

Classes and Templates

9/2/2008

Icetips Utilities

Published: September 2008

Publisher

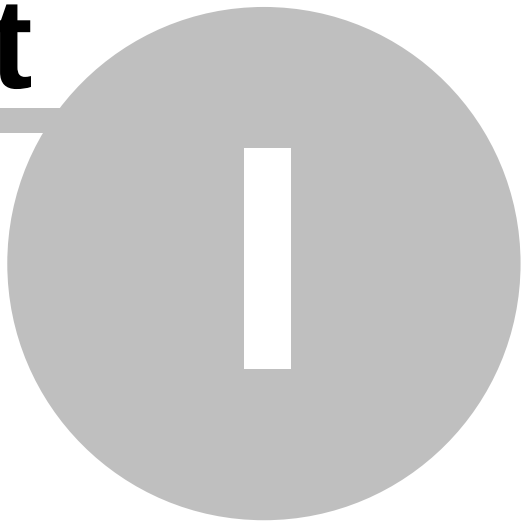
Icetips Creative, Inc.

Managing Editor

Arnor Baldvinsson

Table of Contents

Part



Chapter 1 - Icetips Utilities

1 Icetips Utilities

Welcome to the Icetips Utilities build 1.1.2319

This documentation is based on the classes in release dated August 21, 2008. Note that some documented classes have not had all the properties and methods documented yet.

Start Here 3

This release has a total of 327 methods and 138 properties in 29 classes (may be slightly different in the build you have received)

This help file details each property and method in each class separately with as much detail as we can. At this point only the Network class is documented. The documentation is installed into "%ROOT%\3rdParty\Docs\Icetips Utility Class\ITUtility.chm" where %ROOT% is the Clarion Root directory depending on which version of Clarion you choose to install the Classes for.

To add the Icetips Utility Class to your application, add the "Icetips Utility Classes Global" template to your applications. It must be applied to all applications that use the classes. The classes are ABC compliant and use a named "ITUTIL" group as definition:

```
!ABCIncludeFile(ITUTIL)
```

The classes are compatible with Clarion 6 and above and may be compatible with Clarion 5.5H but they are not compatible with Clarion 5.5G or earlier. Unless otherwise noted, classes should be instantiated at procedure level. There should be no threading considerations in Clarion 6 and above as far as we know since all instances are at procedure level. The only exception is the Global window threading class.

The classes should also be compatible with Clarion 7, but we have, as of build 0.95, not done any specific testing with C7 on the Icetips Utilities. We will, however, start doing that before next major release.

1.1 Start Here

Build 1.1.2319 - Tuesday, September 02, 2008

Please note that this product is still in active development and this documentation is very much under construction!

Due to various reasons, the releases of the Icetips Utilities betas have been delayed several times. This build includes additions to the documentation, although we still have quite a bit of work left on that front. It includes a new SetupBuilder Class that is specifically designed to work with SetupBuilder as well as Windows Vista in order to copy files to appropriate user locations after installation. This class also provides methods to compile and control SetupBuilder projects outside of the SetupBuilder IDE. Lindersoft has, as usual, be very responsive to our requests to add options to set certain compiler variables etc. and for that we are most grateful.

We have changed our classes to load all external apis dynamically at runtime. This means we are no longer dependent on .LIB files which caused duplicate problems with some other third party products. This also makes it easier for us to use various API calls that we may need in the furture as we do not need to include any additional files.

PLEASE NOTE:

Beta 3, Build 0.95.0000 also includes a template to add the classes to your application. This is a global template that MUST be applied to all templates where you are using the classes.

Build 0.93.0000 - July 26, 2007 Beta 1

Please remember that this is a beta release and not a final release.

There is quite a bit of work left on the documentation and we will also be creating both demo applications to demonstrate the use of the classes as well as training videos. Please check the Utilities webpage at http://www.icetips.com/utilities.php regularly for update information.

The classes do not yet have any implementation templates. I.e. you include them by using embedded code and instanciate them using embedded code.

How to add the classes to your application

To add the Icetips utilities to your application use the "Icetips Utility Classes Global" global template to your Global Extensions. This gives you access to all the classes in that file in your application.

How to use the classes in your procedures

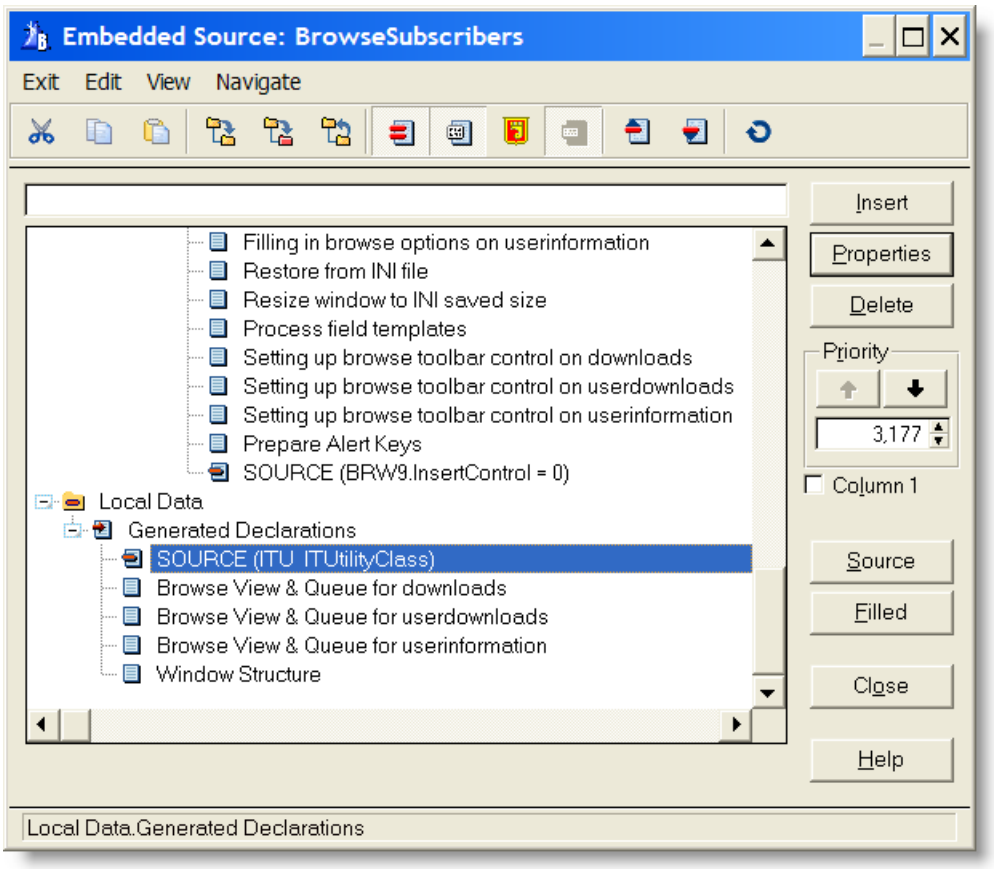
To instanciate a class in your procedure you simply declare an instance. Go to the Local Data embed and add the instanciation code, such as:

```
ITU  ITUtilityClass
```

or:

ITS ITStringClass

etc. You can derive the classes, but most of the methods are very specific so you generally don't need to override or derive. Some classes may need deriving to override virtual methods. The screenshot below shows how the ITUtiltyClass is instanciated in a procedure.



Once you have instanciated the class you can access the methods in it.

1.2 Welcome

Thank you for buying our Icetips Utility Class subscription. This is a set of Clarion classes that add various functionality to your programs.

This help file details each property and method in each class separately with as much detail as we can. Some of the classes may be undocumented when this document is released so please bear with us if something is missing. We have spent 4 years writing these classes as our needs demanded and the task of documenting is monumental since we did not do that as we wrote the classes.

Classes:

We will add functionality to these classes as we need to and if you have any suggestions for new classes, methods or properties, please let us know at http://www.icetips.com/suggestions.php

1.3 Compile issues in Clarion

Up to and including version 6.3 build 9056 there were no issues with compiles in Clarion. However in 9057 and 9058 certain problems were introduced.

Version 6.3, build 9057 introduced options to use OMIT() in ABC header files, but it was not implemented correctly and this caused problems with some of our classes. Build 9058 fixed this.

Version 6.3, build 9058 had a problem with the locally linked runtime library causing duplicate symbols in certain DLL apis. The fix is to use the local runtime library from 9057! Please refer to http://www.cwaddons.com/company/errata.html where you can download zip files with the appropriate lib files for the win32 library and winInet library.

1.4 Compile issues during BETA

While we try our best to make sure that the BETA releases are properly installed and functionaly, some things are just not there yet - which is why this is a beta release and not a final release:)

As of Beta 3, Build 0.95.500, there are no known compile issues.

1.5 Documentation Conventions

We have tried our best to produce a uniform and standardized documentation for our classes.

Please note that properties and methods are sorted alphabetically in this documentation index. However that is not always exactly the same as in the ITUtilities.inc file. In the source the Construct and Destruct method are always at the end of the method list. This may not always be in the documentation although we try to remember to put them at the end.

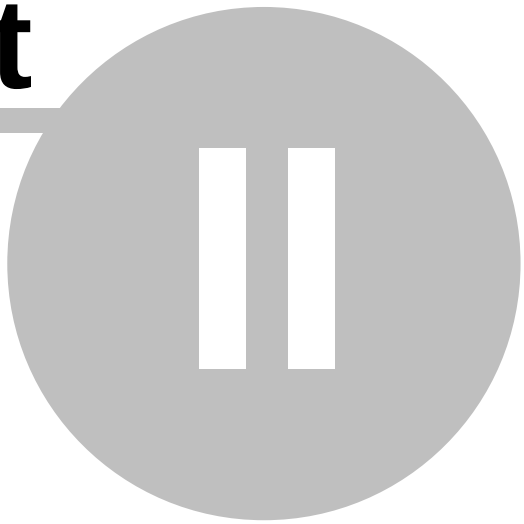
1.6 Coding conventions

We follow fairly strict coding convention that makes the code look good and hopefully very readable to our users.

The rules are simple:

1. 2 Character indents
2. Mixed case statements, i.e. KeyCode() instead of keycode() or KEYCODE()
3. Upper case logical keywords, i.e. NOT, AND, OR
4. Only full upper case keywords are SELF and PARENT
5. Never, ever use period instead of End, ever!
6. Space between statements and arithmetic characters, such as X + 1 instead of X+1 or S & X instead of S&X
7. Clarion properties are upper cased PROP the mixed case, i.e. PROP:LineHeight
8. Constructors and Destructors are declared after other methods.
9. Class properties are declared before methods.

Part



Chapter 2 - Version History

2 Version History

Enter topic text here.

2.1 2008

This chapter lists all releases in 2008 along with changes to classes, templates and documentation.

Version 1.1.2319 - Tuesday, September 02, 2008

Updates, features:
1. New method in CoreClass, RemoveForwardSlash

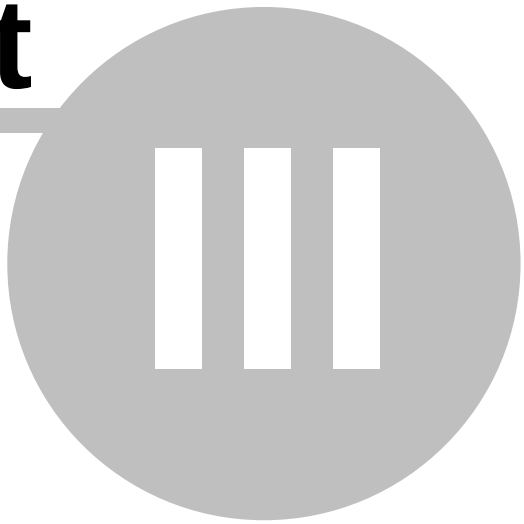
Fixes:
1. MS Header template had the wrong image size. Fixed.
2. tThemedControls was missing the TYPE attribute. Fixed.
3. "Add Header Sort to Queue" template made use of a class property (UsePictureForCase) that was not in Clarion 6.1 causing compile errors in Clarion versions prior to Clarion 6.2. Fixed.
4. SelectFile in the FileSelectClass 42 could clear variable when the FileDialog was canceled. Regression in build 2316. Fixed.
5. ITUtilityClass.inc file was not update with current version number. Fixed.
6. SearchReplace 31 method in the CoreClass 18 would fail when searching for '/' when the string to be searched contained '//' - only the first character would be replaced. Fixed.
7. SearchReplace 31 method in the CoreClass 18 would fail when searching for '/' and replacing with '//' - it would fill the entire string with '/' from the first occurance. This only happens if the replace string was longer and if both search and replace contained all the same character. Fixed.

Version 1.1.2316 - August 27, 2008

Updates, features:
2. Core Class documentation finished.
3. Demo apps will now be done for each individual class, rather than for the whole project as it get's way too big.
4. Core Class demo application finished.
5. LastApiError 21 and LastApiErrorCode 21 properties added to class. They are set in the GetLastAPIError 26 and GetLastAPIErrorCode 26
6. Install improved and tested on a clean Clarion 6.3 9059 platform.
7. Build Automator script created to automatically handle new builds.

Fixes:
8. GetFileAttrib 24(String...) was not passing the string on to the GetFileAttrib 24(*CString...) method, causing it to fail since no filename was passed to it. Fixed.
9. GetLastApiError 26 and GetLastAPIErrorCode 26 methods moved from WindowsClass 106 to CoreClass 18
10. ITWinWiz.tpl contained a reference to a template that was not included with the Utilities. Fixed.

Part



Chapter 3 - Classes

3 Classes

There are 30 classes in the Icetips Utility Classes right now. These are classes that we have been using in our own development work starting around 2003. The Icetips Utility Classes are standard ABC classes. In Beta 3 a Global extension template was added to implement the classes in the application. As documentation and development continues we will add class templates so the classes can be easily added to procedures as well as to create derived classes and enable overriding methods. Some of the classes have not been fully documented yet.

3.1 Armadillo Class

3.1.1 Overview Armadillo Class

```
ITArmadilloClass
Class(ITShellClass),TYPE,Module('ITArmadilloClass.clw'),Link('ITArmadilloClass',_
ITUtilLinkMode_),DLL(_ITUtilDllMode_)
HideDebugView              Byte
ShowEnterKeyDialog         Procedure(), BYTE ! Returns true/false if a key was
entered
InstallKey                 Procedure(STRING pName, STRING pCode), BYTE ! Returns
true/false if the key was valid
UpdateEnvironmentVars      Procedure
NotCompiledMessage         Procedure(String pS)
PTD                         Procedure(String pS, Byte pHideDebug=False),VIRTUAL
Construct                  Procedure
Destruct                   Procedure
                         End
```

3.1.2 Properties Armadillo Class

Enter topic text here.

3.1.3 Methods Armadillo Class

Enter topic text here.

3.2 Armadillo Code Generator Class

3.2.1 Overview Armadillo Code Generator Class

```
ITArmCodGenClass
Class(ITShellClass),TYPE,Module('ITArmCodGenClass.clw'),Link('ITArmCodGenClass',_
ITUtilLinkMode_),DLL(_ITUtilDllMode_)
Template                    CString(255)
ExpireInDays                Long
CreateCodeShort3Key         Procedure(Byte pLevel, String pName, String pTemplate,
Long pDays=0),STRING
                          End
```

3.2.2 Properties Armadillo Code Generator Class

Enter topic text here.

3.2.3 Methods Armadillo Code Generator Class

Enter topic text here.

3.3 Controls Class

3.3.1 Overview Controls Class

```
ITControlsClass
CLASS(ITStringClass),TYPE,Module('ITControlsClass.clw'),Link('ITControlsClass',_I
TUtilLinkMode_),DLL(_ITUtilDllMode_)
Controls                    &ITCtrlQ
RegisterWindow               Procedure(Byte pIncFrame=0)
GetControlText               Procedure(Long pFEQ),String
GetTypeText                  Procedure(Long pType),String
IsTranslatable               Procedure(Long pFEQ),Byte
GetControlByLabel            Procedure(String pLabel)
RemoveControlByLabel         Procedure(String pLabel,Byte pRemove=0)
RemoveControlList            Procedure(String pControlList,Byte pRemove=0)
CheckListbox                 Procedure(Long pFEQ),Byte
Construct                    Procedure
Destruct                     Procedure
                           END
```

3.3.2 Methods Controls Class

Enter topic text here.

3.3.3 Properties Controls Class

Enter topic text here.

3.4 Core Class

3.4.1 Overview Core Class

The core class includes methods that can be used by other classes, basic methods that perform low level functions. It contains basic search method, file splitting functions, path functions and such. Any absolute core functions that we figure we may need will be added to the core class.

```
ITCoreClass
Class,TYPE,Module('ITCoreClass.clw'),Link('ITCoreClass',_ITUtilLinkMode_),DLL(_IT
UtilDllMode_)

DebugLevel                  Byte
EXEName                     CString(1025)
FileParts                   Group(FNS_Parts)
                          End
ProgPath                    CString(10241)
ProgramCommandLine          CString(10241)
ProgramDebugOn              Byte
ComputerName                CString(IT_MAX_COMPUTERNAME_LENGTH+1)
UserName                    CString(256)
XPThemesPresent             Byte
LastAPIError              String(255)
LastAPIErrorCode          Long

CreateGUID                  Procedure(Byte pAddBraces=1),STRING
FixPath                     Procedure(*CString pPath),String
FixPath                     Procedure(String pPath),String
GetComputerName             Procedure(),String,PROC         ! Returns the name
of the computer.  Puts it into the ComputerName property
GetFileAttrib               Procedure(*CString pFile, <*Byte pReadOnly>, <*Byte
pHidden>, <*Byte pSystem>),Long,PROC
GetFileAttrib               Procedure(String pFile, <*Byte pReadOnly>, <*Byte
pHidden>, <*Byte pSystem>),Long,PROC
GetFilePart                 Procedure(String pFilename,Byte pPart),String
GetLastAPIError             Procedure(<*Long pErrorCode>),String
GetLastAPIErrorCode         Procedure(),Long
GetTempFilename             Procedure(<String pPath>,<String pPrefix>),String
GetTempFolder               Procedure(),String
GetFilePart                 Procedure(String pFilename,Byte pPart),String
GetUserName                 Procedure(),String
IsFileInUse                 Procedure(*CString pFile),Byte
IsFileInUse                 Procedure(String pFile),Byte
IsFolder                    Procedure(*CString pPath),Byte
IsFolder                    Procedure(String pPath),Byte
ODS                         Procedure(String pS, Short pLevel=0),VIRTUAL
ODSD                        Procedure(String pS)
PTD                         Procedure(String pS, Byte pHideDebug=False),VIRTUAL
RemoveBackSlash             Procedure(String pPathOrFile, Byte pTrailing),String
RemoveForwardSlash          Procedure(String pPathOrFile, Byte
pTrailing)!!,String
SetFileAttrib               Procedure(*CString pFile, <Byte pReadOnly>, <Byte
pHidden>, <Byte pSystem>, <Long pAdditionalAttrib>)
SetFileAttrib               Procedure(String pFile, <Byte pReadOnly>, <Byte
pHidden>, <Byte pSystem>, <Long pAdditionalAttrib>)
SearchReplace               Procedure(String pFind, String pReplace, *CString
pSearchS),Long,PROC
SearchReplace               Procedure(String pFind, String pReplace, *String
pSearchS),Long,PROC
```

```
SplitFileParts 33          Procedure(String pFileName)
UnixToWindowsPath 33       Procedure(String pUnixPath),String
WindowsToUnixPath 33       Procedure(String pWindowsPath),String
Construct 34              Procedure
Destruct 34               Procedure
                        End
```

3.4.2 Data Types — Core Class

The core class only uses one data type, the FNS_Parts 19, which is used to split up filenames into it's basic parts of drive, path, filename and extension.

3.4.2.1 FNS_Parts — Core Class - Data Types

The FNS_Parts is a group that is declared in the ITEquates.inc as:

```
FNS_Parts               GROUP,TYPE
P_Drive                   CString(IT_MAX_Path)
P_Dir                     CString(IT_MAX_Path)
P_File                    CString(IT_MAX_Path)
P_Ext                     CString(IT_MAX_Path)
                        END
```

This is used by the CreateGUID method

See also:
CreateGUID 22

3.4.2.2 IT_GUID — Core Class - Data Types

The IT_GUID is a group that is declared in the ITWin32Structures.inc as:

```
IT_GUID                 GROUP,TYPE
Data1                     ULONG
Data2                     ULONG
Data3                     USHORT
Data4                     STRING(8)
                        END
```

The derived FileParts 21 property is used by the SplitFilePart 33 method to store the various file parts.

See also:
FileParts 21
GetFilePart 25
SplitFilePart 33

3.4.3 Properties — Core Class

There are currently 6 properties of the Icetips Utility Core class:

```
DebugLevel 20              Byte
EXEName 21                 CString(1025)
FileParts 21               Group(FNS_Parts 19)
```

```
                                End
ProgPath 21                       CString(10241)
ProgramCommandLine 21             CString(10241)
ProgramDebugOn 22                 Byte
XPThemesPresent 22                   Byte

ComputerName 20                   CString(IT_MAX_COMPUTERNAME_LENGTH+1)
UserName 20                       CString(256)
```

3.4.3.1 UserName

This property is set automatically by the Constructor and contains the active User Name. You can also set this property and retrieve the user name by using the GetUserName() 27 method.

Example:

```
ITC  ITCoreClass
 Code
 Message('User name: ' & ITC.UserName)
 ITC.GetUserName  !! Set the ITC.UserName property
```

See also:
Construct 34

3.4.3.2 ComputerName

This property is set automatically by the Constructor and contains the active Computer Name. You can also set this property and retrieve the user name by using the GetComputerName() 23 method.

Example:

```
ITC  ITCoreClass
 Code
 Message('Computer name: ' & ITC.ComputerName)
 ITC.GetComputerName  !! Set the ComputerName property
```

See also:
Construct 34

3.4.3.3 DebugLevel

DebugLevel is used in the ODS 29 method to determine if the passed string should be sent to OutputDebugString. By default DebugLevel is 0, so to send all strings to OutputDebugString you can use the ODS method with:

Example:

```
ITC  ITCoreClass
 Code
 ITC.ODS('This goes to DebugView',0)
```

or, since the parameter to ODS 29 defaults to 0, you can simply use:

```
ITC  ITCoreClass
 Code
 ITC.ODS('This goes to DebugView')
```

3.4.3.4 EXEName — Core Class - Properties

This is a 1K CString that contains the complete path and name of the exe.

This will contain both the path and the name of the exe.

Example:

```
'C:\Clarion\Apps\MyApp\MyApp.exe'
```

See also:
Construct 34

3.4.3.5 FileParts — Core Class - Properties

This is a GROUP structure derived from FNS_Parts 19. It is used by the SplitFileParts 33 method to store individual parts of a filename.

3.4.3.6 LastApiError — Core Class - Properties

This 256 byte string is set with the last API error message by the GetLastAPIError 26

3.4.3.7 LastApiErrorCode — Core Class - Properties

This LONG properties is set to the last API error code by GetLastAPIError 26 and GetLastAPIErrorCode 26 methods.

3.4.3.8 ProgPath — Core Class - Properties

This is a 10K CString that contains the path to the program.

```
ProgPath                    CString(10241)
```

For example if your program is located in 'C:\Clarion\Apps\MyApps\MyApp.exe', then ProgPath will contain "C:\Clarion\Apps\MyApps"

See also:
Construct 34

3.4.3.9 ProgramCommandLine — Core Class - Properties

This is a 10K CString variable that contains the entire command line for the program.

```
ProgramCommandLine          CString(10241)
```

See also:
Construct 34

3.4.3.10 ProgramDebugOn

This property is set either by passing DEBUG as a runtime parameter to the program, which is then picked up by the Construct method 34, or it can be set by the programmer to True or False. It is used in the ODSD 29 method to determine if the method calls ODS 29 which in turns uses OutputDebugString to print to a debugging viewer.

For the most popular debug viewer visit http://technet.microsoft.com/en-us/sysinternals/bb896647.aspx and download DebugView. It can be set up to run over your network so you can run it on computer A while sending information from an application running on computer B.

See also:
Construct 34

3.4.3.11 XPThemesPresent

This property indicates if the XPThemes template and classes are compiled in the project. This allows certain methods access to the XP Theme options.

See also:
Construct 34

3.4.4 Methods Core Class

There are currently 11 methods in the Icetips Utility Core class:

```
Construct 34                Procedure
Destruct 34                 Procedure
GetFilePart 25              Procedure(String pFilename, Byte pPart),String
ODS 29                      Procedure(String pS, Short pLevel=0),VIRTUAL
ODSD 29                     Procedure(String pS)
PTD 30                      Procedure(String pS, Byte pHideDebug=False),VIRTUAL
RemoveBackSlash 30          Procedure(String pPathOrFile, Byte pTrailing),String
SearchReplace 31            Procedure(String pFind, String pReplace, *String
pSearchS),Long,PROC
SearchReplace 31            Procedure(String pFind, String pReplace, *CString
pSearchS),Long,PROC
SplitFileParts 33           Procedure(String pFileName)
UnixToWindowsPath 33        Procedure(String pUnixPath),String  ! Changes / to \
WindowsToUnixPath 33        Procedure(String pWindowsPath),String  ! Changes \
to /
```

3.4.4.1 CreateGUID

Prototype: **(Byte pAddBraces=1),String**

pAddBraces Adds curly braces around the GUID. Example: {87C4B7EF-F219-4FD7-AB94-BEA4287C2E2F} If this parameter is false, no curly braces are added and this GUID would look like 87C4B7EF-F219-4FD7-AB94-BEA4287C2E2F

Returns The GUID formatted as specified by the pAddBraces parameter.

This method uses CoCreateGUID and StringFromGUID2 api calls to create and format a Global Unique Identifier which can be used for all kinds of things.

Example:

```
ITC  ITCoreClass
GUID String(40)
 Code
 GUID = ITC.CreateGUID()
```

See also:
IT_GUID 19

3.4.4.2 FixPath — Core Class - Methods

Prototype: **(String pPath),String**

pPath The path to fix

Returns The fixed path name

This method takes a path name and checks if it contains a trailing backslash or not and returns the path WITH a trailing backslash. It is overloaded with a method that takes a *CString parameter so it can be used with both string variables and CString variables.

Example:

```
ITC ITCoreClass
p  CString(2049)
 Code
 p = 'c:\temp'
 Message('Fixed path = ' & ITC.FixPath(p))
```

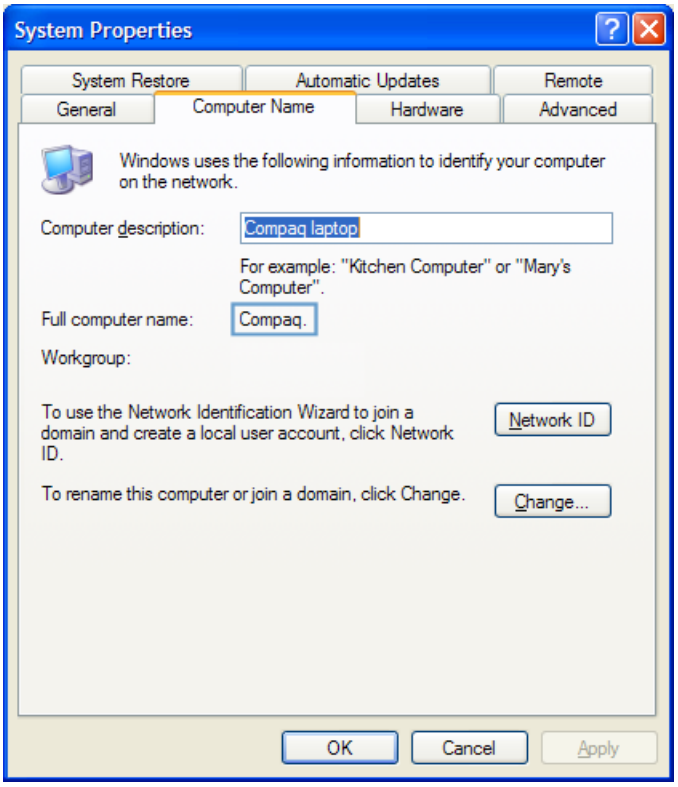
See also:
IsFolder 28
RemoveBackslash 30
CheckLeadingBackslash 52
CheckTrailingBackslash 53

3.4.4.3 GetComputerName — Core Class - Methods

Prototype: **(),String,PROC**

Returns Return the computer name

This method uses the GetComputerName api to retrieve the name of the computer as shown below.



On this computer the GetComputerName would return "Compaq" Note that the APIs normally return the computer name as all upper case.

Example:

```
ITC  ITCoreClass
 Code
 Message('Computer Name: ' & ITC.GetComputerName())
 !! Using the ITC.ComputerName property would also work.
```

See also:
ComputerName 20

3.4.4.4 GetFileAttrib

Core Class - Methods

Prototype:	**(String pFile, <*Byte pReadOnly>, <*Byte pHidden>, <*Byte pSystem>), Long,PROC**
pFile	Name of the file to check
pReadOnly	Optional parameter to receive the Read-Only bit
pHidden	Optional parameter to receive the Hidden bit
pSystem	Optional parameter to receive the System bit
Returns	Returns the file attribute as returned by the GetFileAttributes api.

This method retrieves the file attribute for the specified file, i.e. if it is a read-only, hidden or system file. You can pass in byte variables to receive the information. It is overloaded with a method that

takes a *CString parameter so it can be used with both String and CString variables.

Example:

```
ITC  ITCoreClass
f    CString(2049)
ro   Byte
hi   Byte
sy   Byte
 Code
 f = 'c:\temp\myfile.txt'
 ITC.GetFileAttrib(f,ro,hi,sy)
 ! ro, hi and sy will now either be true or false depending on if the individual attributes are set for the
```

See also:
SetFileAttrib 32

3.4.4.5 GetFilePart — Core Class - Methods

Prototype: **(String pFilename, Byte pPart),String**

pFileName The name, with or without path, of the file.

pPart Defines what part of the filename you want returned. The parts available are FNS_Drive, FNS_Path, FNS_File and FNS_Ext. You can add them up to match any parts you want.

Returns The part of the filename determined by the pPart.

This method returns the part of a full path filename that you request in the pPart parameter. This is a very useful function when dealing with filenames as it allows you to specify what parts you need. Also note the SplitFileParts 33 method which splits all parts of a filename into the FileParts 21 group.

Example:

```
F    CString(1025)
Fn   CString(1025)
ITC  ITCoreClass
 Code
 F = 'C:\Clarion\Apps\Test\GetFilePart\GetFilePart.app'
 Fn = ITC.GetFilePart(F,FNS_Drive+FNS_Path)
      ! Returns: 'C:\Clarion\Apps\Test\GetFilePart\'
 Fn = ITU.GetFilePart(F,FNS_Path)
      ! Returns: \Clarion\Apps\Test\GetFilePart\'
 Fn = ITU.GetFilePart(F,FNS_File)
      ! Returns: 'GetFilePart'
 Fn = ITU.GetFilePart(F,FNS_Ext)
      ! Returns: '.app'
 Fn = ITU.GetFilePart(F,FNS_File+FNS_Ext)
      ! Returns: 'GetFilePart.app'
 Fn = ITU.GetFilePart(F,FNS_Drive+FNS_Path+FNS_File)
      ! Returns: 'C:\Clarion\Apps\Test\GetFilePart\GetFilePart'
 Fn = ITU.GetFilePart(F,FNS_Drive+FNS_Path+FNS_File+FNS_Ext)
      ! Returns: 'C:\Clarion\Apps\Test\GetFilePart\GetFilePart.app'
```

See also:
SplitFileParts 33

3.4.4.6 GetLastAPIError

Prototype: **(<*Long pErrorCode>),String**

[pErrorCode] Optional Parameter that receives the API error code value.

Returns Returns a formatted error message from the operating system.

This method checks for an error after calling any Windows API function and returns a formatted error message and optionally an error code. This is very useful method to use when writing API functions to check for errors and get the error message text as the operating system formats it. The function first retrieves the last error from the operating system with the GetLastError api function. It then uses the FormatMessage api to format the error message text and return it to the calling code. The formatting is done with FORMAT_MESSAGE_FROM_SYSTEM + FORMAT_MESSAGE_MAX_WIDTH_MASK

Example:

```
(none)
```

See also:
GetLastApiErrorCode 26
APIErrorHandler 117
LastApiError 21
LastApiErrorCode 21

3.4.4.7 GetLastAPIErrorCode

Prototype: **(),Long**

Returns Returns the last error code from the system.

This method is very useful to check for errors after calling api functions. This method calls the GetLastError api function and returns the result. In fact this works exactly the same as calling GetLastError() api.

Example:

```
(none)
```

See also:
GetLastAPIError 26
APIErrorHandler 117
LastApiError 21
LastApiErrorCode 21

3.4.4.8 GetTempFilename

Prototype: **(<String pPath>,<String pPrefix>),String**

pPath Optional path. If omitted the default Temp path will be used
pPrefix Optional prefix for the temp filename.

Returns Returns a unique temporary filename.

This method creates a unique filename and returns the full path. If the pPath is specified that path is used, otherwise it will return the temporary path. Please note that this method CREATES the temporary file also. This is done by the GetTempFileName api call to prevent the filename from being allocated to some other program. If you do not intend to use the file, just need a filename, use Remove() to remove the file.

The filename returned is always a short filename. Use LongPath() to get the Long filename for the returned filename. If you run the app from the Clarion IDE it will pick up a different TEMP folder than if you run it independent of the IDE. That is because the Clarion IDE is 16bit so it has a different enviornment setting than standard 32bit programs.

Example:

```
ITC  ITCoreClass
fn   CString(2049)
 Code
 fn = ITC.GetTempFileName()
 Remove(fn)  ! Remove the file.
 Message('Filename: ' & LongPath(fn))
```

See also:
GetTempFolder 27

3.4.4.9 GetTempFolder — Core Class - Methods

Prototype: **(),String**

Returns Returns the folder for temporary files

This method returns the currently set temp folder. This depends on the environment settings and the %TEMP% environment variable. If you run the app from the Clarion IDE it will pick up a different TEMP folder than if you run it independent of the IDE. That is because the Clarion IDE is 16bit so it has a different enviornment setting than standard 32bit programs.

Example:

```
ITC  ITCoreClass
 Code
 Message('Temp folder: ' & ITC.GetTempFolder())
```

See also:
GetTempFilename 26

3.4.4.10 GetUserName — Core Class - Methods

Prototype: **(),String**

Returns The user name for the currently logged in user.

This method retrieves the username for the currently logged in user and returns it and also sets the UserName 20 property.

Example:

```
ITC  ITCoreClass
 Code
 Message('Current user: ' & ITC.GetUserName())
```

See also:
GetComputerName 23

3.4.4.11 IsFileInUse

Prototype: **(String pFile),Byte**

pFile File name to check if it is locked and in use.

Returns Returns true or false depending on if the file is in use.

This method attempts to open the specified file in GENERIC_WRITE mode to see if it can be opened for writing. If not, it is deemed in use by some other program or process. If the file can be opened the method returns false. A file that is in use can generally be opened for reading. This method is very useful for operations such as deleting files or opening them with write access. If the file is in use it can't be written to. It is overloaded with a method that takes a *CString parameter so it can be used with both string variables and CString variables.

Example:

```
ITC ITCoreClass
Fn  CString(2049)
 Code
 Fn = ITC.EXEName
 ITC.ODS('ExeName = ' & Fn)
 If ITC.IsFileInUse(Fn) !! Should ALWAYS return true
   Message('This program is in use.')
 Else
   Message('This program is not in use (should never happen)')
 End

 Fn = ITC.GetFilePart(ITC.EXEName,FNS_FullPath+FNS_File) & '.app'
 ITC.ODS('AppName = ' & Fn)
 If ITC.IsFileInUse(Fn) !! Should not be in use.
   Message('This appfile is in use.')
 Else
   Message('The appfile is not in use.')
 End
```

See also:
IsFolder 28

3.4.4.12 IsFolder

Prototype: **(String pPath),String**

pPath The path to check if it's a folder or a filename

Returns Return information

Method information

It is overloaded with a method that takes a *CString parameter so it can be used with both string variables and CString variables.

Example:

```
ITC ITCoreClass
Fn  CString(2049)
S   CString(1024)
 Code
 Fn = ITC.EXEName
 S = '"' & Fn & '"' & Choose(ITC.IsFolder(Fn)=True,' IS ',' is NOT ') & 'a folder.'
 Fn = LongPath()
 S = S & '|' & '"' & Fn & '"' & Choose(ITC.IsFolder(Fn)=True,' IS ',' is NOT ') & 'a folder.'
 Message(S,'IsFolder')
```

See also:
IsFileInUse 28
GetFileAttrib 24

3.4.4.13 ODS — Core Class - Methods

Prototype: **(String pS, Short pLevel=0), VIRTUAL**

pS String to send to OutputDebugString
pLevel Indicates DebugLevel 20 to compare with. This parameter defaults to 0.

This method sends the string directly to OutputDebugString after converting it to CString if the pLevel is equal or more than the DebugLevel 20 property value.

Example:

```
ITC  ITCoreClass
 Code
 ITC.ODS('Check if this shows up in DebugView')
 ITC.DebugLevel=2
 ITC.ODS('This should NOT show up',1)
```

See also:
DebugLevel 20

3.4.4.14 ODSD — Core Class - Methods

Prototype: **(String pS)**

pS String to send to OutputDebugString

This method sends the string to ODS 29 if the ProgramDebugOn 22 property is set to True.

Example:

```
ITC  ITCoreClass
 Code
 ITC.ProgramDebugOn = True
```

```
ITC.ODSD('This will show up in DebugView')
 ITC.ProgramDebugOn = False
 ITC.ODSD('This will NOT show up in DebugView')
```

See also:
ProgramDebugOn 22
ODS 29

3.4.4.15 PTD — Core Class - Methods

Prototype: **(String pS, Byte pHideDebug=False), VIRTUAL**

pS String to send to OutputDebugString

pHideDebug Flag that can be used in derived methods to prevent the method to send the output to OutputDebugString, but rather redirect it to some other output device, such as a file.

This virtual method is used to **P**rint **T**o **D**ebug and send the output to tools such as DebugView from www.systeminternals.com. Microsoft acquired SystemInternals in July 2006, but the utilities are still free and available for download at http://technet.microsoft.com/en-us/sysinternals/bb896647.aspx This method calls the ODS 29 method passing the pS parameter to it if the pHideDebug is false.

Example:

```
ITC  ITCoreClass
 Code
 ITC.PTD('Check if this shows up in DebugView')
 ITC.PTD('This should not show up in DebugView',True)
```

See also:
ODS 29

3.4.4.16 RemoveBackSlash — Core Class - Methods

Prototype: **(String pPathOrFile, Byte pTrailing),String**

pPathOrFile Path or filename to check.

pTrailing If true, the method strips trailing backslashes, if false it strips leading backslashes.

Returns The stripped path or filename.

This function will remove either leading or trailing backslashes from file/path names.

Example:

```
Fn   CString(1025)
ITC  ITCoreClass
 Code
 Fn = 'C:\Clarion\'
 Fn = ITC.RemoveBackSlash(Fn,True)   ! Fn is now 'C:\Clarion'
 Fn = '\Clarion\'
 Fn = ITC.RemoveBackSlash(Fn,False)  ! Fn is now 'Clarion\'
 Fn = ITC.RemoveBackSlash(Fn,True)   ! Fn is now 'Clarion'
```

See also:
RemoveForwardSlash 31

3.4.4.17 RemoveForwardSlash — Core Class - Methods

Prototype: **(String pPathOrFile, Byte pTrailing),String**

pPathOrFile URL, path or filename to check.

pTrailing If true, the method strips trailing forwardslashes, if false it strips leading forwardslashes.

Returns The stripped path or filename.

This function will remove either leading or trailing forwardslashes from file/path names.

Example:

```
URL  CString(1025)
ITC  ITCoreClass
 Code
 Fn = 'http://www.icetips.com/'
 Fn = ITC.RemoveBackSlash(Fn,True)   ! Fn is now 'http://www.icetips.com'
 Fn = '//www.icetips.com/'
 Fn = ITC.RemoveBackSlash(Fn,False)  ! Fn is now 'www.icetips.com/'
 Fn = ITC.RemoveBackSlash(Fn,True)   ! Fn is now //www.icetips.com'
```

See also:
RemoveBackSlash 30

3.4.4.18 SearchReplace — Core Class - Methods

Prototype: **(String pFind, String pReplace, *String pSearchS),Long,PROC**

pFind The string to search for.

pReplace The string to replace with.

pSearchS The string to search and replace in. Note that this is passed by address so you must pass a variable to this method.

Returns The replaced string.

This is a simple but powerful search and replace method. The method is overloaded with a method that takes a *CString parameter so it can be used with both string variables and CString variables. The search is NOT case sensitive.

Example:

```
S    String(255)
ITC  ITCoreClass
 Code
 S = 'Check this out now'
 ITC.SearchReplace('now','NOW',S) ! S is now:  'Check this out NOW'
```

3.4.4.19 SetFileAttrib

Prototype: **(String pFile, <Byte pReadOnly>, <Byte pHidden>, <Byte pSystem>, <Long pAdditionalAttrib>),IT_DWORD**

pFile	The filename to change attributes on.
pReadOnly	Set the Read-Only attribute
pHidden	Set the Hidden attribute
pSystem	Set the System attribute
pAdditionalAttrib	Set additional attributes.

Returns Returns 0 if the function failed and non-zero value if it succeeded.

This method changes the attributes of the specified file. If only the filename is specified or the pReadOnly, pHidden and pSystem are all set to zero the attribute is set to FILE_ATTRIBUTE_ARCHIVE. The example below is from the CoreClassDemo.app and demonstrates the use of variables to set the attributes. The demo app also shows how to retrieve the attributes when a file is selected and set the variables. We suggest you study the code in the demo application.

Example:

```
ITC ITCoreClass
S   CString(256)
 Code
 If Not Loc:AttribFile
   Post(EVENT:Accepted, ?LookupFile)
   Exit
 End
 If Loc:ReadOnly+Loc:Hidden+Loc:System > 0
   If Loc:ReadOnly
     S = ' +ReadOnly'
   End
   If Loc:Hidden
     S = S & ' +Hidden'
   End
   If Loc:System
     S = S & ' +System'
   End
   S = 'to' & S
 Else
   S = ' back to Archive Only'
 End

 If Message('Are you sure that you want to change the attributes for "' & Clip(Loc:AttribFile) &|
          '" ' & S &|
          '?||Note that you may need to change your Windows Explorer settings to see Hidden and System file
          'SetFileAttrib',|
          ICON:Question,BUTTON:Yes+BUTTON:No,BUTTON:No) = BUTTON:Yes
   If Not ITC.SetFileAttrib(Loc:AttribFile,Loc:ReadOnly,Loc:Hidden,Loc:System)
     Message('Could not set the attributes on the file:||' & ITW.GetLastAPIError())
   End
 End
```

See also:

3.4.4.20 SplitFileParts — Core Class - Methods

Prototype: **(String pFileName)**

pFileName Name of the file to split up

This method splits up a filename that is passed to it into drive, directory, filename and extension. These parts are stored in the FileParts group derived from FNS_Parts[19]. The method does not return any data, instead access the group components directly, see below. The FileParts group is cleared on each call to SplitFileParts so you can not rely on information from previous call to be available after a second call to SplitFileParts.

Example:

```
ITC  ITCoreClass
S  String(1024)
 Code
 S = 'C:\Clarion\Apps\MyApp\MyApp.exe'
 ITC.SplitFileParts(S)
 Message('File parts: ' &|
         'Drive: ' & ITC.FileParts.P_Drive &|
         '|Dir:  ' & ITC.FileParts.P_Dir &|
         '|File: ' & ITC.FileParts.P_File &|
         '|Ext:  ' & ITC.FileParts.P_Ext)
```

See also:
GetFilePart[25]

3.4.4.21 UnixToWindowsPath — Core Class - Methods

Prototype: **(String pUnixPath),String**

pUnixPath Path with / separated directory or folder names.

Returns Path with \ separated directory or folder names.

This method is useful when duplicating paths on a local machine and on a server.

Example:

```
ITC  ITCoreClass
ServerPath  CString(256)
LocalPath   CString(256)
 Code
 ! Path() returns 'C:\Clarion\Apps\MyApp'
 ServerPath = '/images/thisimage.png'
 LocalPath  = Path() & ITC.UnixToWindowsPath(ServerPath)
 ! LocalPath is now: 'C:\Clarion\Apps\MyApp\images\thisimage.png'
```

See also:
WindowsToUnixPath[33]

3.4.4.22 WindowsToUnixPath — Core Class - Methods

Prototype: **(String pWindowsPath),String**

pWindowsPath Path with \ separated directory or folder names.

Returns Path with / separated directory or folder names.

This method is useful when duplicating paths on a local machine and on a server.

Example:

```
ITC  ITCoreClass
ServerPath  CString(256)
LocalPath   CString(256)
 Code
 LocalPath  = '\images\thisimage.png'
 ServerPath  = ITC.WindowsToUnixPath(LocalPath)
 ! ServerPath is now: '/images/thisimage.png'
```

See also:
UnixToWindowsPath 33

3.4.4.23 Construct

Prototype: **None**

The Core class Construct method retrieves information about the running program and stores them in the class properties:

EXEName 21	Set to COMMAND('0') which contains the full path to the executable name that is running.
ProgPath 21	This stores the drive + path of the EXEName
ProgramCommandLine 21	This stores the whole command line of the program as returned by COMMAND('')
ProgramDebugOn 22	This stores the value of a runtime parameter as returned by COMMAND('DEBUG')
XPThemesPresent 22	This indicates if the XP Themes are present or not.
ComputerName 20	This stores the computer name.
UserName 20	This stores the user name of the currently logged in user.

See also:
EXEName 21
ProgPath 21
ProgramCommandLine 21
ProgramDebugOn 22

3.4.4.24 Destruct

Prototype: **None**

The Core class Destruct method currently has no code in it.

3.5 Date Class

3.5.1 Overview Date Class

```
ITDateClass
Class(ITWindowsClass),TYPE,Module('ITDateClass.clw'),Link('ITDateClass',_ITUtilLi
nkMode_),DLL(_ITUtilDllMode_)

Months                      &IT_MonthQueue
Days                        &IT_WeekDayQueue
InitMonthNames              Procedure
InitDayNames                Procedure
SetMonthName                Procedure(Byte pMonth, String pMonthName, <String
pShortName>)
GetMonthName                Procedure(Byte pMonth, Byte pLongName=True),String
SetDayName                  Procedure(Byte pDay, String pDayName, <String
pShortName>)
GetDayName                  Procedure(Byte pDay, Byte pLongName=True),String
GetPreviousWeekDay          Procedure(Long pBaseDate, Byte pWeekDay, Byte
pDayIfSame=True),LONG
GetNextWeekDay              Procedure(Long pBaseDate, Byte pWeekDay, Byte
pDayIfSame=True),LONG
Construct                   Procedure
Destruct                    Procedure
                          End
```

3.5.2 Properties Date Class

Enter topic text here.

3.5.3 Methods Date Class

Enter topic text here.

3.6 Debug Class

3.6.1 Overview Debug Class

```
ITDebugClass
CLASS(ITUtilityClass),TYPE,Module('ITDebugClass.clw'),Link('ITDebugClass',_ITUtil
LinkMode_),DLL(_ITUtilDllMode_)

UseFile                     Byte
UseODS                      Byte
OutputFile                  CString(1025)
Output                      &ITDebugQueue
ODS                         Procedure(String pS),VIRTUAL
Init                        Procedure(Byte pUseFile=False, Byte pAlsoODS=True,
<String pFileName>)
WriteToFile                 Procedure(Byte pAppend=True)
Construct                   Procedure
Destruct                    Procedure
                          END
```

3.6.2 Properties Debug Class

Enter topic text here.

3.6.3 Methods Debug Class

Enter topic text here.

3.7 Directory Class

3.7.1 Overview Directory Class

```
ITDirectoryClass
CLASS(ITUtilityClass),TYPE,Module('ITDirectoryClass.clw'),Link('ITDirectoryClass'
,_ITUtilLinkMode_),DLL(_ITUtilDllMode_)

FileQueue                  &ITFileQueue
ReadDir                    CString(2049) ! Path part of the pPath passed to
ReadDirectory
ReadFiles                  CString(129)  ! Filename + Extension part of pPath
passed to ReadDirectory
NumberOfFiles              Long
ReadDirectory              Procedure(String pPath, Long pAttrib=ff_:normal, Byte
pFree=True),Long,Proc
LowercaseExtension         Procedure
DumpInQueue                Procedure(Queue pQ, <*? pFName>,<*? pFSize>,<*?
pFDate>,<*? pFTime>,<*? pFAttrib>,<*? pFullName>)
Construct                    Procedure
Destruct                     Procedure
                           END
```

3.7.2 Properties Directory Class

Enter topic text here.

3.7.3 Methods Directory Class

Enter topic text here.

3.8 EXIF Class

3.8.1 Overview EXIF Class

This class uses an activeX from http://www.watermarker.com/ Unfortunately this Active-X is slow and we are going to rewrite this class to use the FreeImage library. If you have the Active-X you can use this class to extract EXIF information from image files.

```
ITExifClass
CLASS(ITStringClass),TYPE,Module('ITExifClass.clw'),Link('ITExifClass',_ITUtilLin
kMode_),DLL(_ITUtilDllMode_)

! This uses the aisExif.dll from http://www.watermarker.com/
Exif                        &ITExifQ
ExifList                    &String
LoadParameters              Procedure,VIRTUAL
ReadExifInfo                Procedure(String pImageName)
GetParam                    Procedure(String pParam),String
SetParam                    Procedure(String pImageName, String pParam, String
pValue)
DumpInQueue                 Procedure(Queue pQ, *? pParam, *? pValue, Byte
pFilledOnly=True, |
                                      Byte pUseDescription=True, Byte
pConvertDates=True),VIRTUAL
AddQueueEntry               Procedure(String pParameter, Queue pQ, *? pParam, *?
pValue, |
                                      Byte pUseDescription=True, Byte
pConvertDates=True),VIRTUAL
GetDateTimeValue            Procedure(String pDateTime),String,VIRTUAL
GetDateValue                Procedure(String pDateTime),Long
GetTimeValue                Procedure(String pDateTime),Long
Construct                   Procedure
Destruct                    Procedure
                          END
```

3.8.2 Properties EXIF Class

Enter topic text here.

3.8.3 Methods EXIF Class

Enter topic text here.

3.9 Export Class

3.9.1 Overview Export Class

```
ITExportClass
Class(ITUtilityClass),TYPE,Module('ITExportClass.clw'),Link('ITExportClass',_ITUt
ilLinkMode_),DLL(_ITUtilDllMode_)

FileRecord                  &GROUP
ExportedFile                &File
ExportedView                &View
ExportFile                  String(IT_MAX_Path)
AllFields                   &FieldQueueType
ExpFields                   &FieldQueueType
NumberOfFields              Long
Initialized                 Byte
ExportReady                 Byte
RecordsExported             Long
QuoteCharacter              String(1),PRIVATE
DelimiterCharacter          String(1),PRIVATE

Init                        Procedure(FILE pFile)
Init                        Procedure(VIEW pView)
Init                        Procedure,PRIVATE
GetNumberOfFields           Procedure(FILE pFile),LONG
LoadFileFields              Procedure(FILE pFile),LONG,PROC
LoadViewFields              Procedure(),LONG,PROC
AddExportField              Procedure(STRING pFieldName, <STRING pHeaderName>,
<BYTE pQuote>, <BYTE pIsVar>, <BYTE pVarIsNum>),BYTE,PROC
SetQuoteCharacter           Procedure(STRING pQuoteChar)
SetDelimiter                Procedure(STRING pDelimiter)
GetQuoteCharacter           Procedure(),STRING
GetDelimiter                Procedure(),STRING
StartExport                 Procedure(STRING pExportFile, Byte pWriteHeaders=True,
Byte pQuoteHeaders=True)
WriteHeaders                Procedure(BYTE pQuote=True, <STRING pDelimiter>,
<STRING pQuoteWith>),BYTE,PROC
                                      ! Returns true if successful, false if it
failed.
ExportRecord                Procedure(),BYTE,PROC  ! Returns true if successful,
false if it failed.
WriteLine                   Procedure(String pLine),BYTE,PROC ! Writes an
unformatted line to the file
EndExport                   Procedure(),LONG,PROC  ! Returns number of records
exported successfully.
ParseHeaderNameFromField    Procedure(STRING pFieldName),STRING
Kill                        Procedure()
                          End
```

3.10 File Class

3.10.1 Overview File Class

```
ITFileClass
Class(ITNetworkClass),TYPE,Module('ITFileClass.clw'),Link('ITFileClass',_ITUtilLi
nkMode_),DLL(_ITUtilDllMode_)

LocalDrives                &IT_LocalDrives
EnumLocalDrives            Procedure(),Long  ! Returns the number of enumerated
drives
GetDriveType               Procedure(String pDrive),Long
GetDriveTypeString         Procedure(String pDrive),String
IsLocalDrive               Procedure(String pPath),Byte  ! Takes drive, path or
path+file and returns true/false
Construct                  Procedure
Destruct                   Procedure
                         End
```

3.11 File Search Class

3.11.1 Overview File Search Class

```
ITFileSearchClass
Class(ITFileClass),TYPE,Module('ITFileSearchClass.clw'),Link('ITFileSearchClass',
_ITUtilLinkMode_),DLL(_ITUtilDllMode_)

Directories                &ITDirQueue
Files                      &ITFileQueue
WildCards                  &ITWildcards
StartDirectory             CString(2049)
TotalDirectories           Long
TotalFiles                 Long
FindHandle                 IT_HANDLE
FileSort                   Byte
FileFilter                 CString(256)

SetStartDir                Procedure(String pSD)
SetFileFilter              Procedure(String pFF)
ScanDirectories            Procedure(<String pSD>, Byte
pShowWindow=False),Long,Proc  ! Returns number of directories
ReadDirectories            Procedure(String pDir, Byte pShowWindow=False),Private
   ! Recursive read of all directories in directory
CountFilesInDirectories    Procedure(String pFF,<String pDirectory>),Long
ResetFileCounters          Procedure(<String pDirectory>)
GetLevel                   Procedure(String pDir),Byte
SetFileSort                Procedure(Byte pSort)
SetNoFileSort              Procedure
GetFileSort                Procedure(), Byte
ScanFiles                  Procedure(<String pDir>,<String pWC>, Long
pAttrib=FF_:NORMAL, BYTE pCountOnly=False),Long,Proc
                           ! Returns number of files
GetWildcardList            Procedure(String pWildcards),Long,Proc
!GetDirectories             Procedure(*FILE:Queue pQ,String pFn),Long,Proc
!GetFiles                   Procedure(*FILE:Queue pQ,String pFn, Long
pAttrib),Long,Proc
Construct                  Procedure
Destruct                   Procedure
                         End
```

3.11.2 Properties File Search Class

Enter topic text here.

3.11.3 Methods File Search Class

Enter topic text here.

3.12 File Select Class

3.12.1 Overview File Select Class

Enter topic text here.

3.13 Files Class

3.13.1 Overview Files Class

This is a tiny class with just one method in it. Pass a file lable to the GetFilePrefix and you will get back the prefix string for the file. Very useful when using PROP:Alias to set the prefix for SQL statements.

```
ITFilesClass
CLASS(ITStringClass),TYPE,Module('ITFilesClass.clw'),Link('ITFilesClass',_ITUtilL
inkMode_),DLL(_ITUtilDllMode_)

GetFilePrefix              Procedure(FILE pF),String
                         END
```

3.13.2 Methods Files Class

Enter topic text here.

3.13.3 Properties Files Class

Enter topic text here.

3.14 Global Thread Class

3.14.1 Overview Global Thread Class

```
ITGlobalThreadClass
CLASS(ITUtilityClass),TYPE,Module('ITGlobalThreadClass.clw'),Link('ITGlobalThread
Class',_ITUtilLinkMode_),DLL(_ITUtilDllMode_)
CriticalSection            &ICriticalSection,PRIVATE
WindowThreads              &ITGlobalThreadQ
FrameWindow                &Window
FrameProcedure             CString(256)
FrameThread                Long
AddWindow                  Procedure(WINDOW pWin, String pProcedureName, Byte
pIsFrame=0)
GetInsertLevel             Procedure(Long pThread),Long
CloseWindow                Procedure(Long pThread, Long pHandle)
CloseAllWindows            Procedure
RemoveWindow               Procedure
Construct                  Procedure
Destruct                   Procedure
                         END
```

3.14.2 Properties Global Thread Class

Enter topic text here.

3.14.3 Methods Global Thread Class

Enter topic text here.

3.15 Hyperlink Class

3.15.1 Overview Hyperlink Class

```
ITHyperLinkClass
Class(ITFileClass),TYPE,Module('ITHyperLinkClass.clw'),Link('ITHyperLinkClass',_I
TUtilLinkMode_),DLL(_ITUtilDllMode_)
HyperLinks                  &ITControlQ
Accepted                    Long  ! Ctrl accepted
CursorFile                  CString(256)
DefaultTip                  CString(256)
UseDefaultTip               Byte  ! Set before calling RegisterControl
Initialized                 Byte

Init                         Procedure(<String pCursor>,<String pTip>)
Kill                        Procedure
RegisterControl             Procedure(Long pFEQ, Long pContentsFEQ=0, Byte
pChangeToString=True, <String pCursor>,<String pURL>)
UnRegisterControl           Procedure(Long pFEQ)
HyperLinkAccepted           Procedure(Long pFEQ),Byte ! Returns true if pFEQ is
found in HyperLinks
TakeAccepted                Procedure(),BYTE,VIRTUAL  ! Event handler for regions
TakeResize                  Procedure(),BYTE,PROC!,PRIVATE  ! Event handler for
resizing
SetControlPositions         Procedure(Long pCtrl, Long pRgn)
SetControlFonts             Procedure(Long pCtrl, Long pString)
SetHyperLink                Procedure(Long pFEQ)
SetLinkTip                  Procedure(Long pFEQ, String pTip, Byte
pAppendContents=TRUE)
HyperLinkExists             Procedure(Long pFEQ),Byte
Construct                   Procedure
Destruct                    Procedure
                          End
```

3.15.2 Properties Hyperlink Class

Enter topic text here.

3.15.3 Methods Hyperlink Class

Enter topic text here.

3.16 Image Class

3.16.1 Overview Image Class

```
ITImageClass
Class(ITFileClass),TYPE,Module('ITImageClass.clw'),Link('ITImageClass',_ITUtilLin
kMode_),DLL(_ITUtilDllMode_)
BufferValue                Short,Private
UseBufferValue             Byte
BufferSet                  Byte

SetBufferValue             Procedure(Short pBuffer)
GetBufferValue             Procedure(),Short
UseBufferValue             Procedure(Byte pUseBuffer)
CheckBuffer                Procedure
ResizeImage                Procedure(Long pFEQ, String pImage,<Long pW>, <Long
pH>)
GetImageSize               Procedure(Long pImageFEQ, *Long pW, *Long pH, Byte
pPixels=False)
SetImageSize               Procedure(Long pImageFEQ, Long pW, Long pH, Byte
pPixels=False)
GetSetImageSize            Procedure(Long pImageFEQ, Byte pPixels=False)
SetRelativePosition        Procedure(Long pImageFEQ, Long pRelativeFEQ, Long pX,
Long pY, Byte pJust)
Construct                  Procedure
Destruct                   Procedure
                         End
```

3.17 Locale Class

3.17.1 Overview Locale Class

```
ITLocaleClass
CLASS(ITUtilityClass),TYPE,Module('ITLocaleClass.clw'),Link('ITLocaleClass',_ITUt
ilLinkMode_),DLL(_ITUtilDllMode_)
MonthNames                CString(41),Dim(12)
MonthsFilled              Byte
GetMonths                 Procedure
GetMonthName              Procedure(Byte pMonth),String
Construct                 Procedure
Destruct                  Procedure
                        END
```

3.17.2 Properties Locale Class

Enter topic text here.

3.17.3 Methods Locale Class

Enter topic text here.

3.18 Macro Class

3.18.1 Overview Macro Class

```
ITMacroClass
Class(ITUtilityClass),TYPE,Module('ITMacroClass.clw'),Link('ITMacroClass',_ITUtil
LinkMode_),DLL(_ITUtilDllMode_)

Macros 48                    &IT_Macros
MacroCounter 48              Long
AddMacro 49                  Procedure(String pMacro, String pValue)
ExpandMacro 49               Procedure(String pMacro),String
ExpandReplace 49             Procedure(String pStr),String  ! Searches and
replaces all macros in string
Construct 49                 Procedure
Destruct 48                  Procedure
                         End
```

3.18.2 Properties Macro Class

```
Macros 48                    &IT_Macros
MacroCounter 48              Long
```

3.18.2.1 MacroCounter Macro Class - Properties

Enter topic text here.

3.18.2.2 Macros Macro Class - Properties

Enter topic text here.

3.18.3 Methods Macro Class

```
AddMacro 49                  Procedure(String pMacro, String pValue)
ExpandMacro 49               Procedure(String pMacro),String
ExpandReplace 49             Procedure(String pStr),String  ! Searches and
replaces all macros in string
Construct 34                 Procedure
Destruct 34                  Procedure
```

3.18.3.1 Destruct Macro Class - Methods

Enter topic text here.

3.18.3.2 Construct — Macro Class - Methods

Enter topic text here.

3.18.3.3 ExpandReplace — Macro Class - Methods

Enter topic text here.

3.18.3.4 ExpandMacro — Macro Class - Methods

Enter topic text here.

3.18.3.5 AddMacro — Macro Class - Methods

Enter topic text here.

3.18.4 Equates — Macro Class

Enter topic text here.

3.19 Network Class

3.19.1 Overview Network Class

```
ITNetworkClass
Class(ITShellClass),TYPE,Module('ITNetworkClass.clw'),Link('ITNetworkClass',_ITUt
ilLinkMode_),DLL(_ITUtilDllMode_)

Is98NetCompatible          Byte
NetEnumOpen                Byte,PRIVATE
NetResources               &IT_NETRESOURCEs
LocalResources             &IT_NetworkShares
ComputerName               CString(IT_MAX_COMPUTERNAME_LENGTH+1)
HideDebugView              Byte

GetComputerName            Procedure(),String,PROC         ! Returns the name of
the computer.  Puts it into the ComputerName property
GetNetworkDriveName        Procedure(String pDrive),String ! Returns the remote
drive name
GetNetworkFileName         Procedure(String pFile),String  ! Returns the remote
filename (UNC)
GetLocalNetworkFileName    Procedure(String pFile),String  ! Returns the local or
remote filename (UNC)
EnumNetworkDrives          Procedure(),LONG,PROC           ! Returns number of
enumerated resources
EnumNetworkPrinters        Procedure()
EnumLocalShares            Procedure(),Long                ! Returnes the number
of enumerated resources
EnumLocalSharesWin32       Procedure(),Long,PRIVATE        ! Returnes the number
of enumerated resources
EnumLocalSharesWinNT       Procedure(),Long,PRIVATE        ! Returnes the number
of enumerated resources
IsLocalShare               Procedure(String pPath), Byte   ! Returns true if the
path is on a local share
IsUNC                      Procedure(String pPath), Byte   ! Returns true if the
path starts with \\
ParseKeyData               Procedure(String pData),PRIVATE
CheckTrailingBackSlash     Procedure(STRING pPath), String
CheckLeadingBackSlash      Procedure(STRING pPath), String
ShowLocalShares            Procedure,PRIVATE
PTD                        Procedure(String pS, Byte pHideDebug=False),VIRTUAL
Construct                  Procedure
Destruct                   Procedure
                         End
```

3.19.2 Debugging Network Class

The methods in the ITNetwork class have debug code in them that uses OutputDebugString api call to send the information to an external tool. In order to view the debug information you need a tool such as DebugView from www.sysinternals.com. **DebugView is a free tool** and can be used on multiple computers so you can have the debug information show up on a second computer and view it as your program runs on another computer.

In order to turn the debug logging on in your application, simply run your program with /ITNETWORK command line parameter:

MyProgram.exe /ITNetwork

The command line parameter is not case sensitive. /ITNetwork, /ITNETWORK or /itnetwork will all give the same effect.

Below is an example output from a call to GetLocalNetworkFileName 54 on a computer called COMPAQ3200 with a local share of C:\ called PRES_200. Notice that all lines coming from the ITNetworkClass are prefixed with "ITNetwork:" You can use this as a filter in

```
 1   3.12749410   [2908] ITNetwork: Construct, Hide DebugView = False
 2   12.32415581  [2908] ITNetwork: GetLocalNetworkFileName Begins
 3   12.32423476  [2908] ITNetwork: GetNetworkFilename, WNetGetUniversalName =
2250
 4   12.32429489  [2908] ITNetwork: pFile   = C:\Installations\aawsepersonal.exe

 5   12.32433433  [2908] ITNetwork: UncName = C:\Installations\aawsepersonal.exe

 6   12.32437811  [2908] ITNetwork: Filename: C:\Installations\aawsepersonal.exe

 7   12.32441549  [2908] ITNetwork: Need to find local share name
 8   12.32457083  [2908] ITNetwork: Before GetComputerName
 9   12.32460993  [2908] ITNetwork: GetComputerName = ABCOMPAQ3200
10   12.32464809  [2908] ITNetwork: After GetComputerName
11   12.32465735  [2908] ITNetwork: Computer Name: ABCOMPAQ3200
12   12.32479729  [2908] ITNetwork:
13   12.32483673  [2908] ITNetwork:
-------------------------------------------------
14   12.32490315  [2908] ITNetwork:  Enumerated Local Shares
15   12.32494094  [2908] ITNetwork: CCSFlags:    538976288
16   12.32497734  [2908] ITNetwork: MaxUses:     -1
17   12.32501403  [2908] ITNetwork: Name:        PRES_C200
18   12.32505090  [2908] ITNetwork: Path:        C:\
19   12.32508669  [2908] ITNetwork: Permissions: 0
20   12.32512336  [2908] ITNetwork: Type:        0
21   12.32516002  [2908] ITNetwork: --------------------
22   12.32519856  [2908] ITNetwork:
-------------------------------------------------
23   12.32524488  [2908] ITNetwork: EnumLocalShares, Before Return, KeyIndex = 1

24   12.32528265  [2908] ITNetwork: After EnumLocalShares
25   12.32532021  [2908] ITNetwork: 1 local shares found
26   12.32535735  [2908] ITNetwork: Path = C:\
27   12.32540102  [2908] ITNetwork: Name = PRES_C200
28   12.32544396  [2908] ITNetwork: Share found: PRES_C200
29   12.32545328  [2908] ITNetwork: Share found:
\\ABCOMPAQ3200\PRES_C200\Installations\aawsepersonal.exe
30   12.32549089  [2908] ITNetwork: UNCName =
\\ABCOMPAQ3200\PRES_C200\Installations\aawsepersonal.exe
31   12.32552804  [2908] ITNetwork: GetLocalNetworkFileName Ends
32   12.44182026  [2908] ITNetwork: Construct, Hide DebugView = False
```

See also:
PTD 56

3.19.3 Properties — Network Class

3.19.3.1 Is98NetCompatible — Network Class - Properties

This property is true if the Operating System is a Windows 98/95/ME. It is false if the Operating System is Windows NT/2000/XP.

3.19.3.2 LocalResources

Queue of IT_NetworkShares type.

```
IT_NetworkShares          QUEUE,TYPE
CCSFlags                    LONG
MaxUses                     UNSIGNED
Name                        CString(IT_MAX_PATH)
Path                        CString(IT_MAX_PATH)
Permissions                 LONG
Type                        LONG
                          END
```

3.19.3.3 NetEnumOpen

Private property. Used internally to determine if the network enumeration process was opened successfully or not.

3.19.3.4 NetResources

Queue of IT_NETRESOURCEs type.

```
IT_NETRESOURCEs          QUEUE,TYPE
Scope                       ULONG
Type                        ULONG
DisplayType                 ULONG
Usage                       ULONG
LocalName                   Cstring(256)
RemoteName                  Cstring(256)
Comment                     Cstring(256)
Provider                    Cstring(256)
                          END
```

3.19.4 Methods

3.19.4.1 CheckLeadingBackSlash

Prototype: (STRING pPath), String

Takes a path as parameter. Checks if the path begins with backslash and if it does not, it adds a leading backslash.

Example:

```
MyProc   Proc
P    String(255)
ITN  ITNetworkClass
 Code
 P = 'somepath\myfile.txt'
 P = ITN.CheckLeadingBackSlash(P)
 Message('P is now = ' & P)
```

P would now be '\somepath\myfile.txt'

This is mostly for internal use in the class, but is made public in case someone needs it.

See also:
CheckTrailingBackSlash 53

3.19.4.2 CheckTrailingBackSlash — Network Class - Methods

Prototype: (STRING pPath), String

Takes a path as parameter. Checks if the path ends with backslash and if it does not, it adds a leading backslash.

Example:

```
MyProc   Proc
P    String(255)
ITN  ITNetworkClass
 Code
 P = 'somepath\myfile.txt'
 P = ITN.CheckLeadingBackSlash(P)
 Message('P is now = ' & P)
```

P would now be '\somepath\myfile.txt'

This is mostly for internal use in the class, but is made public in case someone needs it.

See also:
CheckLeadingBackSlash 52

3.19.4.3 ConvertToAscii — Network Class - Methods

Not implemented

3.19.4.4 EnumLocalShares — Network Class - Methods

Prototype: ()

Primary method to enumerate local shares. This method checks for the Computer name 23 and if no computer name is detected, it indicates that there are no local shares so it skips the process. Since the storage information for local shares is different on 95/98/ME than it is on NT/2000/XP, this method determines which operating system is in use and then calls the EnumLocalSharesWin32 53 or EnumLocalSharesWinNT 54 depending on the result of the call to GetWindowVersion()

For example code, see the source for the GetLocalNetworkFileName 54 method.

See also:
EnumLocalSharesWin32 53
EnumLocalSharesWinNT 54

3.19.4.5 EnumLocalSharesWin32 — Network Class - Methods

Prototype: (), Long

Returns the number of enumerated local shares. This method enumerates information that is stored in the "LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Network\Lanman" registry key.

See also:
EnumLocalShares 53
EnumLocalSharesWinNT 54

3.19.4.6 EnumLocalSharesWinNT

Prototype: (), Long

Returns the number of enumerated local shares. This method enumerates information that is stored in the "LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\lanmanserver\Shares" registry key.

See also:
EnumLocalShares 53
EnumLocalSharesWin32 53

3.19.4.7 EnumNetworkDrives

Prototype: (), LONG, PROC

This method is not functional yet.

This method enumerates remote shared network resources, i.e. it does not enumerate local shared resources. This can be used to find all available network resources.

3.19.4.8 GetLocalNetworkFileName

Prototype: (String pFile),String

This method takes a filename as parameter and will return the appropriate UNC filename for it, no matter if it is a local file or a remote file (on another computer), using the enumeration functions to find the appropritate name. Note that if no UNC name can be established it will return the original name back, so this method should be 100% safe on standalone computers where there are no local or remote shares to connect to. In that case it will simply return the filename/path passed to it.

Example:

```
MyProc   Proc
P    String(255)
ITN  ITNetworkClass
 Code
 P = 'c:\somepath\myfile.txt'
 P = ITN.GetLocalNetworkFileName(P)
 Message('P is now = ' & P)
```

P would now be '\\COMPAQ\TheCDrive\somepath\myfile.txt' if the computer name was "COMPAQ" and the local share name for the C:\ was "TheCDrive". If there was no local share, this would return "c:\somepath\myfile.txt" even if the computer name was still "COMPAQ" If you want to provide an option later on to convert to UNC, it would be advisable to store the computer name with the filename, that way the filename(s) could be fairly easily converted to UNC.

See also:
CheckLeadingBackSlash 52
EnumLocalShares 53
GetNetworkFileName 55
LocalResources 52

3.19.4.9 GetNetworkDriveName

Prototype: (String pDrive), String

This method returns the UNC drive name for a mapped drive.

Example:

```
MyProc   Proc
P    String(255)
ITN  ITNetworkClass
 Code
 P = 'Z:'
 P = ITN.GetNetworkDriveName(P)
 Message('P is now = ' & P)
```

If the Z: drive is mapped to "RemoteC" drive share on "RemotePC" computer, this would return "\\RemotePC\RemoteC"

See also:
GetComputerName 23
GetLocalNetworkFileName 54
GetNetworkFileName 55

3.19.4.10 GetNetworkFileName

Prototype: (String pFile), String

This method returns the UNC name for a remote file. It does NOT return the UNC filename for a local file. Use GetLocalNetworkFileName 54 for that purpose.

Example:

```
MyProc   Proc
P    String(255)
ITN  ITNetworkClass
 Code
 P = 'Z:\somepath\somefile.txt'
 P = ITN.GetNetworkFileName(P)
 Message('P is now = ' & P)
```

If the Z: drive is mapped to "RemoteC" drive share on "RemotePC" computer, this would return "\\RemotePC\RemoteC\somepath\somefile.txt" If the Z: drive is a local drive, the GetNetworkFileName would return "Z:\somepath\somefile.txt"

See also:

GetComputerName 23
GetLocalNetworkFileName 54
GetNetworkDriveName 55

3.19.4.11 ParseKeyData

Network Class - Methods

Prototype: (String pData)

This method is a private method that is used to parse data for the LocalResource queue.

3.19.4.12 PTD

Network Class - Methods

Prototype: (String pS, Byte pHideDebug=False),VIRTUAL

This method is a virtual method that uses the PTD in the parent class to PrintToDebug, i.e. call the OutputDebugString api call to send debugging information to tools such as DebugView.

See also:
Debugging 50

3.19.4.13 Construct

Network Class - Methods

The constructor initializes the NetResources 52 and LocalResources 52 queues. It also gets the window version and sets the Is98NetCompatible 51 property to true if the VersionPlatformID is set to IT_VER_PLATFORM_WIN32_WINDOWS.

3.19.4.14 Destruct

Network Class - Methods

The Destructor frees the NetResource 52 and LocalResoucers 52 and disposes of them.

3.20 Page Of Pages Class

3.20.1 Overview Page Of Pages Class

```
ITPageOfPagesClass
CLASS(ITUtilityClass),TYPE,Module('ITPageOfPagesClass.clw'),Link('ITPageOfPagesCl
ass',_ITUtilLinkMode_),DLL(_ITUtilDllMode_)

ReportPrevieweQueue       &PrintPreviewFileQueue
TotalPages                Long  ! This is the total counter from the
ReportPreviewQueue
ThisReport                &REPORT
PageBuffer                &String
SearchString              CString(101),Private
PageOf                    Long ! = R - SELF.LastPointer
StartPointer              Long
LastPointer               Long
TimeTaken                 Long

Init                       Procedure(REPORT pReport, PrintPreviewFileQueue pQ,
<String pSearchString>)
SetPageOfPages            Procedure
ReadTheFile               Procedure(String pFileName, Long pFileSize),Long,PROC !
Returns bytes read
WriteTheFile              Procedure(String pFileName, Long pFileSize),Long,PROC !
Returns bytes written
SetPageofText             Procedure(),Byte ! Returns false if nothing was
replace, true if needs to be written back
UpdatePageFile            Procedure(String pFile, Long pFileSize),Byte
SetSearchString           Procedure(String pSearchString)
GetSearchString           Procedure(),String
DisposePageBuffer         Procedure
AllocatePageBuffer        Procedure(Long pBytesToAllocate)
ProfileToODS              Procedure
Construct                 Procedure
Destruct                  Procedure
                        END
```

3.20.2 Properties Page Of Pages Class

Enter topic text here.

3.20.3 Methods Page Of Pages Class

Enter topic text here.

3.21 Periods Class

3.21.1 Overview Periods Class

```
ITPeriodClass
CLASS(ITStringClass),TYPE,Module('ITPeriodClass.clw'),Link('ITPeriodClass',_ITUti
lLinkMode_),DLL(_ITUtilDllMode_)

Years                       &ITPerQ
Months                      &ITPerQ
Days                        &ITPerQ
YearsFEQ                    Long
MonthsFEQ                   Long
DaysFEQ                     Long
SelectedYear                Long
SelectedMonth               Long
SelectedDay                 Long
SelectedDate                Long
SelectedDateFEQ             Long
FirstYear                   Long
LastYear                    Long
YearVar                     ANY
MonthVar                    ANY
DayVar                      ANY
LastFEQ                     Long,Private
MonthNames                  CString(1025)
PeriodType                  Byte,Private

Init                        Procedure(Long pYearList, Long pMonthList, Long
pDaysList, <Long pSelectedDate>, *? pYear, *? pMonth, *? pDay)
Kill                        Procedure
LoadQueues                  Procedure
LoadYears                   Procedure
LoadMonths                  Procedure
LoadDays                    Procedure(Long pYear=0, Long pMonth=0)
AddToQueue                  Procedure(ITPerQ pQ, String pStrVal, Long
pValue),Private
AddYearRange                Procedure(Long pFirstYear, Long pLastYear)
SetMonthNames               Procedure(<String pMonthNames>)
AdjustDropDowns             Procedure
GetMonthDays                Procedure(Long pYear, Long pMonth),Byte
GetCalculatedDate           Procedure(Long pYear=0, Long pMonth=0, Long
pDay=0),Long
GetCalculatedDateStr        Procedure(Long pYear=0, Long pMonth=0, Long
pDay=0),String
GetSelectedDate             Procedure(),Long
TakeEvent                   Procedure,Byte
ResetDropDowns              Procedure
Construct                   Procedure
Destruct                    Procedure
                          End
```

3.21.2 Properties Periods Class

Enter topic text here.

3.21.3 Methods Periods Class

Enter topic text here.

3.22 Popup Class

3.22.1 Overview Popup Class

Enter topic text here.

3.22.2 Properties Popup Class

Enter topic text here.

3.22.3 Methods Popup Class

Enter topic text here.

3.23 Progress Class

3.23.1 Overview — Progress Class

```
ITProgressClass
Class(ITUtilityClass),TYPE,Module('ITProgressClass.clw'),Link('ITProgressClass',_
ITUtilLinkMode_),DLL(_ITUtilDllMode_)

CurrentValue               Long,Private
DisplayControls            &ITDisplayQueue
HideUnhide                 Byte
PercentValue               Byte,Private
ProgressControl            Long,Private
TotalValue                 Long,Private

AddDisplayControl          Procedure(LONG pControLToDisplay,Byte pUpdateOnShow=1)
AddToCurrentValue          Procedure(Long pValueToAdd),LONG,PROC ! Adds value and
returns new current value
Calculate                  Procedure                        ! Calculates percent
GetCurrentPercent          Procedure(),BYTE                 ! Returns 0 - 100
GetCurrentValue            Procedure(),LONG                 ! Returns CurrentValue
GetProgressControl         Procedure(),LONG                 ! Returns
ProgressControl
GetTotalValue              Procedure(),LONG                 ! Returns TotalValue
HideControls               Procedure(BYTE pHide)            ! Hide/unhide display
controls
Init                        Procedure(LONG pProgressControl, Long pTotalValue,
Byte pHideUnhide=1, Byte pCanBeZeroOrOne=True)
Kill                        Procedure
SetCurrentValue             Procedure(Long pCurrentValue)
SetTotalValue               Procedure(LONG pTotalValue)
ShowProgress                Procedure
ShowUpdateProgress          Procedure(Long pValueToAdd)     ! Update progressbar
after adding pValueToAdd to the value.
Update                      Procedure ! Calls ShowUpdateProgress(1)
                          End
```

3.23.2 Properties — Progress Class

Enter topic text here.

3.23.2.1 CurrentValue — Progress Class - Properties

Enter topic text here.

3.23.2.2 DisplayControls — Progress Class - Properties

Enter topic text here.

3.23.2.3 Initialized — Progress Class - Properties

This property is set to true in the Init method and is checked in order to update the progress bar. Prevents problems if the init is not called.

3.23.2.4 HideUnhide

Progress Class - Properties

Enter topic text here.

3.23.2.5 PercentValue

Progress Class - Properties

Enter topic text here.

3.23.2.6 ProgressControl

Progress Class - Properties

Enter topic text here.

3.23.2.7 TotalValue

Progress Class - Properties

Enter topic text here.

3.23.3 Methods

Progress Class

3.23.3.1 AddDisplayControl

Progress Class - Methods

Enter topic text here.

3.23.3.2 AddToCurrentValue

Progress Class - Methods

Enter topic text here.

3.23.3.3 Calculate

Progress Class - Methods

Enter topic text here.

3.23.3.4 GetCurrentPercent

Progress Class - Methods

Enter topic text here.

3.23.3.5 GetProgressControl

Progress Class - Methods

Enter topic text here.

3.23.3.6 GetTotalValue

Progress Class - Methods

Enter topic text here.

3.23.3.7 HideControls

Progress Class - Methods

Enter topic text here.

3.23.3.8 Init

Enter topic text here.

3.23.3.9 Kill

Enter topic text here.

3.23.3.10 SetCurrentValue

Enter topic text here.

3.23.3.11 SetTotalValue

Enter topic text here.

3.23.3.12 ShowProgress

Enter topic text here.

3.23.3.13 ShowUpdateProgress

Enter topic text here.

3.23.3.14 Update

Enter topic text here.

3.24 Record Class

3.24.1 Overview Record Class

```
ITRecordClass
CLASS(ITStringClass),TYPE,Module('ITRecordClass.clw'),Link('ITRecordClass',_ITUti
lLinkMode_),DLL(_ITUtilDllMode_)

ITRecord                    &GROUP
ITFile                      &FILE
RecSize                     Long
FieldNum                    Long
FieldNameLen                Byte
TempString                  &CString,PRIVATE
FieldFormats                &ITFieldFormats
Init                        Procedure(FILE pFile) !!, *GROUP pRecord)
AddFieldFormat              Procedure(String pFieldName, String pFormat)
GetRecordStringSize         Procedure(String pDelimeter),Long
SetFieldNameLen             Procedure(Byte pLen)
GetRecordWithData           Procedure(String pDelimeter, Byte pPad=False),String
WriteRecToFile              Procedure(String pFile, Byte pAppend)
WriteLineToFile             Procedure(String pFile, Byte pAppend, String pLine)
DisposeTempStr              Procedure
Construct                   Procedure
Destruct                    Procedure
                          End
```

3.24.2 Properties Record Class

Enter topic text here.

3.24.3 Methods Record Class

Enter topic text here.

3.25 RTF Text Class

3.25.1 Overview RTF Text Class

Enter topic text here.

3.25.2 Methods RTF Text Class

Enter topic text here.

3.25.3 Properties RTF Text Class

Enter topic text here.

3.26 Select List Class

3.26.1 Overview Select List Class

This class handles selections in a listbox.

```
ITSelectListClass
CLASS(ITUtilityClass),TYPE,Module('ITSelectListClass.clw'),Link('ITSelectListClas
s',_ITUtilLinkMode_),DLL(_ITUtilDllMode_)

ListFEQ                    Long
CurrentChoice              Long
HighPtr                    Long
LowPtr                     Long
LastChoice                 Long
MarkField                  ANY
MarkQueue                  &QUEUE
Init                       Procedure(Long pListFeq, *? pMarkField, QUEUE
pMarkQueue)
Kill                       Procedure
TakeNewSelection           Procedure(),BYTE
                         END
```

3.26.2 Properties Select List Class

Enter topic text here.

3.26.3 Methods Select List Class

Enter topic text here.

3.27 SetupBuilder Class

3.27.1 Overview SetupBuilder Class

The SetupBuilder Class contains various methods to make working with SetupBuilder installs easier. Several methods are not really SetupBuilder specific and could be used with other installation software as well. This particularly applies to methods that copy files from a "global" location to a "local" location. A Global location is a path that is accecssible by all users under all operating systems. A Local location is a path that is accessible only by the current user.

The SetupBuilder class has two main type of methods. The first one serves in your application to copy files from a globally accessible location where SetupBuilder installs the files, to a locally accessible location where your application uses them. This is particularly useful under **Windows Vista** where the installer can only install files into folders that are open to all users. That location may not be appropriate, or even visible, for the users of your application, i.e. if there is user specific data. These methods copy the data from the global folder to a local folder and update a registry key so you can check if the data has been copied.

The second type are methods that you can use to call SetupBuilder to compile your projects, update compiler variables and determine destination folder for the compiled install and build report. Please note that you must have SetupBuilder version 6.6 build 2000 or later for this to work properly. Previous versions will be able to compile the project, but can not change the output folder or the name of the compiled install executable.

Please check the **TestSetupBuilderClass** procedure in the **UtilDemo.app** that is located in your "Clarion\3rdParty\Examples\ITUtilities" folder. It allows you to pick a source and destination CSIDLs for the copying part of the class, as well as pick a project and destination for compiling your project.

```
ITSetupBuilderClass
CLASS(ITVersionClass),TYPE,Module('ITSetupBuilderClass.clw'),|

Link('ITSetupBuilderClass',_ITUtilLinkMode_),DLL(_ITUtilDllMode_)
CompilerVariables 69          &SBCompileVars
DestinationFolder 69          CString(2049)
FilesCopied 70                Byte
GlobalCSIDL 70                Long
LocalCSIDL 70                 Long
PathString 70                 CString(1025)  ! Path for CSIDL_COMMON, CSIDL_LOCAL,
HKLM and HKCU
SBBuildNumber 71              Long
SBCommandLine 71              &CString
SBErrorLogFile 72             CString(2049)  ! Error log file if an error occurs
                                            ! during compile. If no error occurs,
                                            ! this file does not exist.
SBExecutable 72               CString(2049)  ! Setupbuilder.exe path and name for
                                            ! calling compiler
SBGlobalInstallPath 72        CString(2049)  ! CSIDL_COMMON
SBGlobalRegistryKey 73        CString(2049)  ! HKLM
SBHtmlLogFile 73              CString(2049)  ! The HTML file created after a
                                            ! compile
SBLocalInstallPath 73         CString(2049)  ! CSIDL_LOCAL
SBLocalRegistryKey 73         CString(2049)  ! HKCU
SBMajorVersion 74             Byte
SBMinorVersion 74             Byte
SBProjectToCompile 74         CString(2049)  ! Project name and path.
```

```
AddCompilerVariable 75        Procedure(String pVariable, |
                                      String pValue, |
                                      Byte pQuoteValue=0)
BuildCommandLine 75           Procedure(String pBaseCommandLine),String
CompileSBProject 76           Procedure(String pProjectToCompile),Long,PROC
CopyTheFiles 77               Procedure(),Long,PROC   ! Returns number of files
copied
CreateDestinationFolder 78    Procedure(),Long,PROC   ! Creates the Destination
Folder tree
FinishInstall 78              Procedure(String pPath, Long pLocal, Long
pGlobal),Long,PROC
GetDestinationFolder 79       Procedure(),String,PROC ! Fills DestinationFolder and
returns it.
GetGlobalKey 79               Procedure(),String
GetGlobalPath 79              Procedure(),String
GetLocalKey 80                Procedure(),String
GetLocalPath 80               Procedure(),String
GetSBExecutable 80            Procedure(),BYTE        ! Returns true if executable
                                                    ! is found
GetSBVersionInformation 81    Procedurex
SetDestinationFolder 81       Procedure(String pDestination)
SetGlobalCSIDL 81             Procedure(Long pCSIDL)
SetLocalCSIDL 82              Procedure(Long pCSIDL)
SetPathString 83              Procedure(String pPathString)
ShowHTMLLogFile 83            Procedure
ShowLogFile 84                Procedure(Byte pOpenInWindow=True)
ShowLogFile 84                Procedure(INIClass pINIMgr)
Construct 86                  Procedure
Destruct 86                   Procedure
                      END
```

3.27.2 Data Types — SetupBuilder Class

The SetupBuilder class uses one special data type for compiler variables.

SBCompileVars 68

See also:
CompilerVariables 69
AddCompilerVariable 75

3.27.2.1 SBCompileVars — SetupBuilder Class - Data Types

The **SBCompileVars** is a queue type that is used in the SetupBuilder class to store compiler variables and new values to update those variables with during the compiling process.

```
SBCompileVars          QUEUE,TYPE
VariableName             CString(101)
VariableValue            CString(2049)
QuoteValue               Byte
                       END
```

See also:
CompilerVariables 69

3.27.3 Properties SetupBuilder Class

The SetupBuilder class has several properties that store various information during the compile process. Also when the class is used to finish an install and copy files.

```
CompilerVariables 69        &SBCompileVars 68
DestinationFolder 69        CString(2049)
FilesCopied 70              Byte
GlobalCSIDL 70              Long
LocalCSIDL 70               Long
PathString 70               CString(1025)  ! Path for CSIDL_COMMON, CSIDL_LOCAL,
HKLM and HKCU
SBBuildNumber 71            Long
SBCommandLine 71            &CString
SBErrorLogFile 72           CString(2049)  ! Error log file if an error occurs
during compile
SBExecutable 72             CString(2049)  ! Setupbuilder.exe path and name for
calling compiler
SBGlobalInstallPath 72      CString(2049)  ! CSIDL_COMMON
SBGlobalRegistryKey 73      CString(2049)  ! HKLM
SBHtmlLogFile 73            CString(2049)  ! The HTML file created after a
compile
SBLocalInstallPath 73       CString(2049)  ! CSIDL_LOCAL
SBLocalRegistryKey 73       CString(2049)  ! HKCU
SBMajorVersion 74           Byte
SBMinorVersion 74           Byte
SBProjectToCompile 74       CString(2049)  ! Project name and path.
```

3.27.3.1 CompilerVariables SetupBuilder Class - Properties

Queue used to store Compiler variables and new values to update the SetupBuilder project script with. It is declared as:

```
CompilerVariables        &SBCompileVars 68
```

See also:
SBCompileVars 68
AddCompilerVariable 75

3.27.3.2 DestinationFolder SetupBuilder Class - Properties

The DestinationFolder property is a Cstring that contains the destination folder for the compiled install and Build Report when the SetupBuilder compiler is called. By default SetupBuilder uses the path where the project file (*.sb6) is and creates a subfolder with the same name as the project file. For example, if the project file is located in "C:\Products\MyProduct\MyProduct.sb6" the SetupBuilder will compile the install and build report into "C:\Products\MyProduct\MyProduct" which is the default destination folder. The DestinationFolder can be changed to any accessible folder location. If the DestinationFolder does not exist, the class will create it inside the CompileSBProject 76 method before the compiler is called to create the install.

```
DestinationFolder        CString(2049)
```

See also:
CompileSBProject 76

GetDestinationFolder 79
SetDestinationFolder 81

3.27.3.3 FilesCopied

This property is a Byte variable that indicates if the copy process from the Global folder to the Local folder has been done. This property is set in SetPathString 83 and FinishInstall 78 and checked and updated in the CopyTheFiles 77 method.

```
FilesCopied                Byte
```

3.27.3.4 GlobalCSIDL

This Long property contains the CSIDL value for the Global folder that files should be copied FROM. This could for example be IT_CSIDL_COMMON_APPDATA.

```
GlobalCSIDL                Long
```

See also:
LocalCSIDL 70
SetGlobalCSIDL 81
FinishInstall 78

3.27.3.5 LocalCSIDL

This Long property contains the CSIDL value for the Local folder that files should be copied TO. This could for example be IT_CSIDL_PERSONAL or IT_CSIDL_LOCAL_APPDATA depending on the nature of the files.

If this is application data, that the user should not mess with, IT_CSIDL_LOCAL_APPDATA might be more appropriate. Note that on Windows Vista, IT_CSIDL_LOCAL_APPDATA is hidden so it does not show up in the Explorer window.

If this is data that the user may need to copy, backup, etc. then IT_CSIDL_PERSONAL would be more appropriate. IT_CSIDL_PERSONAL generally refers to the "My Documents" folder. Please check out our FREE SpecialFolder *(http://www.icetips.com/downloadfile.php?FileID=71)* program that is invaluable in determining the actual locations on the various operating systems.

```
LocalCSIDL                 Long
```

See also:
SetLocalCSIDL 82
FinishInstall 78

3.27.3.6 PathString

This Cstring property is used to determine the Global and Local locations of the files for copying as well as updating to the registry keys. This property contains a partial path that is appended to the CSIDL paths pointed to by GlobalCSIDL 70 and LocalCSIDL 70. It is also used as key for the HKLM and HKCU registry keys to store the value of the FilesCopied property.

For example if the GlobalCSIDL is IT_CSIDL_COMMON_APPDATA resulting in "C:\Documents and Settings\All Users\Application Data" and the LocalCSIDL is IT_CSIDL_PERSONAL resulting in "C:\Documents and Settings\Arnor\My Documents" and the PathString is "Icetips Creative\Utilities", then this would translate to:

Folders:
SBGlobalInstallPath = "C:\Documents and Settings\All Users\Application Data\Icetips Creative\Utilities"
SBLocalInstallPath = "C:\Documents and Settings\Arnor\My Documents\Icetips Creative\Utilities"

Registry keys:
HKLM\SOFTWARE\Icetips Creative\Utilities
HKCU\SOFTWARE\Icetips Creative\Utilities

The HKCU (Current User) registry keys would contain a single value, FilesCopied, which would be either true or false depending on if the files have been copied or not. The HKLM (Local Machine) registry key is not used at this point.

PathString CString(1025)

See also:
SetPathString 83
SBGlobalInstallPath 72
SBGlobalRegistryKey 73
SBLocalInstallPath 73
SBLocalRegistryKey 73
GetGlobalPath 79
GetGlobalKey 79
GetLocalPath 80
GetLocalKey 80
SetGlobalCSIDL 81
SetLocalCSIDL 82

3.27.3.7 SBBuildNumber

SetupBuilder Class - Properties

This Long property contains the build number of the SetupBuilder executable. For example in version 6.6.2000, the SBBuildNumber would be 2000. This is retrieved from the version information in the SetupBuilder executable, SBExecutable 72.

SBBuildNumber Long

See also:
GetSBExecutable 80
GetSBVersionInformation 81
SBExecutable 72

3.27.3.8 SBCommandLine

SetupBuilder Class - Properties

This dynamic CString contains the commandline used to run the SetupBuilder compiler. It is constructed at runtime based on the executable found for the project file, compiler variables 69 added to the project and if destination is specified or not. As different versions of Windows have different

maximum sizes for the command line, the class checks to see if the commandline being constructed is larger than the maximum allowed by the operating system. If it is too big, the CompileSBProject 76 method will return -2 and the SBCommandLine 71 will be empty. The SBCommandLine 71 is disposed in the Destruct 86 method.

SBCommandLine &CString

See also:
AddCompilerVariable 75
CompileSBProject 76
CompilerVariables 69
Destruct 86

3.27.3.9 SBErrorLogFile

SetupBuilder Class - Properties

If an error occurs during the SetupBuilder compilation, an error log file is created and it's location is placed in this CString property. The location depends on what version of SetupBuilder is being used. If no error occurs during compiling, this file does not exist. I.e. it only exists after a failed compile. There are two methods that can display the error log file. The demo program shows how to determine if an error has occurred and open the log file in a window.

SBErrorLogFile CString(2049)

See also:
ShowLogFile - Window 84
ShowLogFile - ShellExecute 85

3.27.3.10 SBExecutable

SetupBuilder Class - Properties

This CString property contains the full path name to the SetupBuilder executable file. The file location is found by using file association or registry settings.

SBExecutable CString(2049)

See also:
GetSBExecutable 80

3.27.3.11 SBGlobalInstallPath

SetupBuilder Class - Properties

This CString property contains the Global path constructed in the SetGlobalCSIDL 82 and is used as the SOURCE folder for the CopyTheFiles 77 method.

SBGlobalInstallPath CString(2049)

See also:
CopyTheFiles 77
GetGlobalPath 79
SBLocalInstallPath 73
SetGlobalCSIDL 81

3.27.3.12 SBGlobalRegistryKey

This CString property contains the Global registry key. This key is always under the HKLM\SOFTWARE node in the registry. In the current release (Beta 3) this key is not used by the class, except it is constructed properly in the SetPathString 83 method and returned by the GetGlobalKey 79 method.

SBGlobalRegistryKey CString(2049)

See also:
GetGlobalKey 79
SetPathString 83

3.27.3.13 SBHtmlLogFile

This CString property contains the name of the HTML Build Report that SetupBuilder creates when the compiling is done. It is always stored with the compiled install executable with the same name, but a ".htm" extension.

SBHtmlLogFile CString(2049)

See also:
ShowHTMLLogFile 83

3.27.3.14 SBLocalInstallPath

This CString property contains the Local path constructed in the SetLocalCSIDL 82 and is used as the DESTINATION folder for the CopyTheFiles 77 method.

SBLocalInstallPath CString(2049)

See also:
CopyTheFiles 77
GetLocalPath 80
SBGlobalInstallPath 72
SetLocalCSIDL 82

3.27.3.15 SBLocalRegistryKey

This CString property contains the Local registry key that will be updated with the FilesCopied value. This key is always under the HKCU\SOFTWARE node in the registry. The value of this property is constructed in the SetPathString 83.

SBLocalRegistryKey CString(2049)

See also:
GetLocalKey 80
SetPathString 83

3.27.3.16 SBMajorVersion

This Byte property contains the Major version. For example in version 6.6.2000, the SBMajorVersion would be 6. In 6.5.1711 it would also be 6.

```
SBMajorVersion          Byte
```

See also:
GetSBVersionInformation 81
SBBuildNumber 71
SBMinorVersion 74

3.27.3.17 SBMinorVersion

This Byte property contains the Minor version. For example in version 6.6.2000, the SBMinorVersion would be 6. In 6.5.1711 it would be 5.

```
SBMinorVersion          Byte
```

See also:
GetSBVersionInformation 81
SBBuildNumber 71
SBMajorVersion 74

3.27.3.18 SBProjectToCompile

This CString property contains the name of the project to compile. This must be fully qualified path and filename and must have a .SB5 or .SB6 extension.

```
SBProjectToCompile      CString(2049)
```

See also:
CompileSBProject 76

3.27.4 Methods SetupBuilder Class

The SetupBuilder class has several methods for copying file and compiling SetupBuilder projects.

```
AddCompilerVariable 75      Procedure(String pVariable, |
                                    String pValue, |
                                    Byte pQuoteValue=0)
BuildCommandLine 75         Procedure(String pBaseCommandLine),String
CompileSBProject 76         Procedure(String pProjectToCompile),Long,PROC
CopyTheFiles 77             Procedure(),Long,PROC
CreateDestinationFolder 78  Procedure(),Long,PROC
FinishInstall 78            Procedure(String pPath,
                                    Long pLocal,
                                    Long pGlobal),Long,PROC
GetDestinationFolder 79     Procedure(),String,PROC
```

```
GetGlobalKey 79              Procedure(),String
GetGlobalPath 79             Procedure(),String
GetLocalKey 80               Procedure(),String
GetLocalPath 80              Procedure(),String
GetSBExecutable 80           Procedure(),BYTE
GetSBVersionInformation 81   Procedure
SetDestinationFolder 81      Procedure(String pDestination)
SetGlobalCSIDL 81            Procedure(Long pCSIDL)
SetLocalCSIDL 82             Procedure(Long pCSIDL)
SetPathString 83             Procedure(String pPathString)
ShowHTMLLogFile 83           Procedure
ShowLogFile 84               Procedure(Byte pOpenInWindow=True)
ShowLogFile 84               Procedure(INIClass pINIMgr)
Construct 86                 Procedure
Destruct 86                  Procedure
```

3.27.4.1 AddCompilerVariable

Prototype: **(String pVariable, String pValue, Byte pQuoteValue=0)**

pVariable The name of the Compiler variable to update during compile.

pValue The value to pass to SetupBuilder for this variable.

pQuoteValue Determines if the pValue should be quoted when added to the command line.

Returns The method does not return a value

This method is used before compiling a SetupBuilder project. It allows you to pass data to the SetupBuilder compiler that will update the compiler variables in the project with the data in the pValue. This is placed into the CompilerVariables 69 queue and then used in the BuildCommandLine 75 to construct the correct command line to pass to the SetupBuilder compiler.

Please note that the command line length is limited and varies depending on what operating system is in use. There is a basic length checking mechanism built into the BuildCommandLine 75 method and if the command line exceeds the maximum allowed by the operating system, the CompileSBProject 76 terminates and returns -2 to the calling code.

Example:

```
ITS.AddCompilerVariable('PRODUCTVER','0.95.000',True)
ITS.AddCompilerVariable('EXENAME','TestBuild_0.95.000.EXE',True)
```

See also:
BuildCommandLine 75
CompileSBProject 76
CompilerVariables 69

3.27.4.2 BuildCommandLine

Prototype: **(String pBaseCommandLine),String**

pBaseCommandLine Contains the path and name of the SetupBuilder executable and the path and the name of the SetupBuilder project to compile.

Returns Returns the full command line to execute

This method builds up a command line to compile the project. The basic command line that is passed to it contains the SetupBuilder executable and the project to compile, like:

ShortPath(SELF.SBExecutable) & ' /C "' & ShortPath(SELF.SBProjectToCompile) & '"'

Optionally compiler variables can be added to the commandline by using the AddCompilerVariable 75 method. A destination folder can also be specified, but it is optional. If no destination folder is specified, it is constructed by using the project file path and name in the same manner as SetupBuilder does it. This method is called automatically by the CompileSBProject 76.

Please note that the command line length is limited and varies depending on what operating system is in use. There is a basic length checking mechanism built into this method and if the command line exceeds the maximum allowed by the operating system, the CompileSBProject 76 terminates and returns -2 to the calling code.

See also:
AddCompilerVariable 75
CompileSBProject 76

3.27.4.3 CompileSBProject

Prototype: **(String pProjectToCompile),Long,PROC**

pProjectToCompile Path and name of the SetupBuilder project (*.sb5 or *.sb6 files) to compile.

Returns True if the project compiled.
False if the project failed.
-1 if the SetupBuilder executable could not be found.
-2 if the command line is too big.

This method takes the project file passed to it and executes the SetupBuilder compiler to compile it. The example below shows the code that is in the TestSetupBuilderClass proceudure in the UtilDemo.app example application in your "Clarion\3rdParty\Examples\ITUtilities" folder.

Example:

```
CompileTheProject           ROUTINE
 Data
R  Long
 Code
 If Loc:SBProjectDestFolder
   ! Set the destination folder if it is specified.  Comment this out if you
   ! want to use the SetupBuilder default destination.
   ITS.SetDestinationFolder(Loc:SBProjectDestFolder)
 End

 ! First parameter is the SB Compiler Variable.
 ! Second parameter is the value to place into the compiler variable
 ! Third parameter determines if the VALUE is enclosed in double quotes or not.
 ITS.AddCompilerVariable('PRODUCTVER','0.95.000',True)
  ITS.AddCompilerVariable('EXENAME','TestBuild_0.95.000.EXE',True)

 ! NOTE:  This information is sent via the command line to SetupBuilder.
 !        The length of the command line is limited, depending on what
 !        Operating System you are using.  The class will not run the
```

```
!        compiler if the command line exeeds the OS limit, avoiding
!        making a mess of things.  In order to keep the command line
!        as short as possible, make sure that you use ShortPath()
!        on any paths that you need to pass to the SB compiler.

! Compile the project
R =  ITS.CompileSBProject(Loc:SBProjectToCompile)
ITS.ODS('Return value: ' & R)
Case R
Of 1
  If Message('The project was compiled successfully.  ' &|
             'Do you want to view the Build HTML file?',|
             'Project compiled successfully',ICON:Exclamation,|
              BUTTON:No+BUTTON:Yes,BUTTON:Yes) = BUTTON:Yes
    ITS.ShowHTMLLogFile
  End
Of 0
  If Message('Error occured while compiling the project.  ' &|
             'Do you want to view the compile log?',|
             'Errors occurred',ICON:Hand,|
             BUTTON:Yes+BUTTON:No,BUTTON:Yes) = BUTTON:Yes
    ITS.ShowLogFile(IniMGR)
  End
Of -1
  Message('Both SetupBuilder 5 and SetupBuilder 6 are installed ' &|
          'on this machine.  In that case only SetupBuilder 6.x is ' &|
          'supported and a SetupBuilder 5.x project can not be compiled.',|
          'SetupBuilder 5 and 6 detected',ICON:Hand)
Of -2
  Message('The Command line was too long.',|
          'Command Line is too long',ICON:Hand)

End
```

See also:
AddCompilerVariable 75
BuildCommandLine 75

3.27.4.4 CopyTheFiles

Prototype: **(),Long,PROC**

Returns Number of files copied

This method creates a directory structure specified in the PathString property in the folder specified by the LocalCSIDL property. It uses the CreateDirectories method from the UtilityClass and the CopyFiles method from the ShellClass to copy the folder structure from the Global folder in the SBGlobalInstallPath property to the Local folder in the SBLocalInstallPath property. This method is called by the FinishInstall method. When the method is finished copying, it updates the FilesCopied value of the HKCU\SOFTWARE registry key along with the partial path that is specfied in the PathString 70 property.

Example:

```
CopyTheFiles            ROUTINE

 If Loc:LocalCSIDLNumber And Loc:GlobalCSIDLNumber
   ITS.FinishInstall(Loc:PartialPath,  Loc:LocalCSIDLNumber,
Loc:GlobalCSIDLNumber)
 End
```

See also:
GlobalCSIDL 70
LocalCSIDL 70
PathString 70
SBGlobalInstallPath 72
SBLocalInstallPath 73

CreateDirectories 94
UtilityClass 91
ShellClass 87

3.27.4.5 CreateDestinationFolder

SetupBuilder Class - Methods

Prototype: **(),Long,PROC**

Returns The number of directories created

This method uses the CreateDirectories 94 method from the Utilities class 91 to create the destination folder tree specified in the DestinationFolder 69 property. This method is used by the CompileSBProject 76 method.

Example:

```
ITS.CreateDestinationFolder
```

See also:
CompileSBProject 76
CreateDirectories 94
DestinationFolder 69
Utilities class 91

3.27.4.6 FinishInstall

SetupBuilder Class - Methods

Prototype: **(String pPath, Long pLocal, Long pGlobal),Long,PROC**

pPath Partial path

pLocal Local CSIDL value, such as IT_CSIDL_PERSONAL

pGlobal Global CSIDL value, such as IT_CSIDL_COMMON_APPDATA

Returns Returns number of files copied

This mehod is really the only method that you need to use to set up copying files from one place to the other. See the CopyTheFiles routine in the TestSetupBuilderClass procedure in the UtilDemo.app in your "Clarion\3rdParty\Examples\ITUtilities" folder.

Example:

```
CopyTheFiles              ROUTINE

 If Loc:LocalCSIDLNumber And Loc:GlobalCSIDLNumber
   ITS.FinishInstall(Loc:PartialPath,Loc:LocalCSIDLNumber,Loc:GlobalCSIDLNumber)
 End
```

See also:
CopyTheFiles 77
SetGlobalCSIDL 81
SetLocalCSIDL 82
SetPathString 83

3.27.4.7 GetDestinationFolder

Prototype: **(),String,PROC**

Returns Returns the contents of the DestinationFolder property

If the DestinationFolder 69 property is not set, this method will construct it based on the path and name of the Project file that is being compiled, stored in the SBProjectToCompile 74 property

Example:

```
S = ITS.GetDestinationFolder()
```

See also:
DestinationFolder 69
SBProjectToCompile 74

3.27.4.8 GetGlobalKey

Prototype: **(),String**

Returns Returns the contents of the SBGlobalRegistryKey property.

This method returns the contents of the SBGlobalRegistryKey 73 property.

Example:

```
K = ITS.GetGlobalKey()
```

See also:
SBGlobalRegistryKey 73

3.27.4.9 GetGlobalPath

Prototype: **(),String**

Returns Returns the contents of the SBGlobalInstallPath property.

This method returns the contents of the SBGlobalInstallPath 72 property.

Example:

```
P = ITS.GetGlobalPath()
```

See also:
SBGlobalInstallPath 72

3.27.4.10 GetLocalKey

Prototype: **(),String**

Returns Returns the contents of the SBLocalRegistryKey property

This method returns the contents of the SBLocalRegistryKey 73 property

Example:

```
K = ITS.GetLocalKey()
```

See also:
SBLocalRegistryKey 73

3.27.4.11 GetLocalPath

Prototype: **(),String**

Returns Returns the contents of the SBLocalInstallPath property.

This method returns the contents of the SBLocalInstallPath 73 property.

Example:

```
P = ITS.GetLocalPath()
```

See also:
SBLocalInstallPath 73

3.27.4.12 GetSBExecutable

Prototype: **(),BYTE**

Returns Returns true if the SetupBuilder executable was found

This method finds the executable for the appropriate version of SetupBuilder. This is determined by the extension of the Project to compile stored in the SBProjectToCompile 74 property. This method also constructs the path and filename of the error log, stored in the SBErrorLogFile 72 property. This method is used by the CompileSBProject 76 method. The path and name of the SetupBuilder executable is stored in the SBExecutable 72 property.

Example:

```
If ITS.GetSBExecutable()
  Run(ITS.SBExecutable)
End
```

See also:
CompileSBProject 76
SBExecutable 72
SBProjectToCompile 74

3.27.4.13 GetSBVersionInformation

Prototype: **None**

Returns The method does not return a value

This method retrieves the version information from the SetupBuiler executable stored in the SBExecutable[72] property. It then splits up the first 3 components into SBMajorVersion[74], SBMinorVersion[74] and SBBuildNumber[71] properties. The method uses the GetVersionInfo method of the Version Class[104] to retrieve the version information.

Example:

```
 ITS.GetSBVersionInformation
 Message('SetupBuilder version: ' & ITS.SBMajorVersion & '.' & ITS.SBMinorVersion
& '.' & ITS.SBBuildNumber)
```

See also:
SBBuildNumber[71]
SBExecutable[72]
SBMajorVersion[74]
SBMinorVersion[74]
Version Class[104]

3.27.4.14 SetDestinationFolder

Prototype: **(String pDestination)**

pDestination The Destination folder

Returns The method does not return a value

The method assigs the pDestination parameter to the DestinationFolder[69] property. This property is used to determine the folder where the compiled installation executable and build report html files are put after the project is compiled with the CompileSBProject[76] method. To specify a non-default destination, call this method before calling the CompileSBProject[76] method.

Example:

```
If Loc:SBProjectDestFolder
  ! Set the destination folder if it is specified.
  ITS.SetDestinationFolder(Loc:SBProjectDestFolder)
End
! Compile the project
R =  ITS.CompileSBProject(Loc:SBProjectToCompile)
```

See also:
DestinationFolder[69]
CompileSBProject[76]

3.27.4.15 SetGlobalCSIDL

Prototype: **(Long pCSIDL)**

pCSIDL The CSIDL value to use

Returns The method does not return a value

This method constructs a path name based on the CSIDL passed to it and the PathString 70 property using the GetSpecialFolder method of the Shell Class 87. The CSIDL value is stored in the GlobalCSIDL 70 property.

Example:

```
GetGlobalPath           ROUTINE
 Get(SFQ,Choice(?Loc:GlobalCSIDL))
 ITS.SetPathString(Loc:PartialPath)
 ITS.SetGlobalCSIDL(SFQ.FolderIDValue)

 Loc:GlobalCSIDLNumber = SFQ.FolderIDValue
 Loc:GlobalPath        = ITS.GetGlobalPath()
 Loc:GlobalKey         = 'HKLM:\' & ITS.GetGlobalKey()
 Display()
```

See also:
GlobalCSIDL 70
PathString 70
Shell Class 87

3.27.4.16 SetLocalCSIDL

Prototype: **(Long pCSIDL)**

pCSIDL The CSIDL value to use

Returns The method does not return a value

This method constructs a path name based on the CSIDL passed to it and the PathString 70 property using the GetSpecialFolder method of the Shell Class 87. The CSIDL value is stored in the LocalCSIDL 70 property.

Example:

```
GetLocalPath            ROUTINE
 Get(SFQ,Choice(?Loc:LocalCSIDL))
 ITS.SetPathString(Loc:PartialPath)
 ITS.SetLocalCSIDL(SFQ.FolderIDValue)

 Loc:LocalPath        = ITS.GetLocalPath()
 Loc:LocalKey         = 'HKCU:\' & ITS.GetLocalKey()
 Loc:LocalCSIDLNumber = SFQ.FolderIDValue
 Display()
```

See also:
LocalCSIDL 70
PathString 70
Shell Class 87

3.27.4.17 SetPathString

Prototype: **(String pPathString)**

pPathString Partial path to add to Global and Local folders as well as registry key

Returns The method does not return a value.

This method assigns the pPathString parameter value to the PathString 70 property. It also constructs the SBGlobalRegistryKey 73 and SBLocalRegistryKey 73 properties from the PathString. Finally it assigns the appropriate HKCU registry key to the FilesCopied 70 property indicating if the files have been copied to the local folder for the current user or not.

Example:

```
GetLocalPath            ROUTINE
 Get(SFQ,Choice(?Loc:LocalCSIDL))
 ITS.SetPathString(Loc:PartialPath)
 ITS.SetLocalCSIDL(SFQ.FolderIDValue)

 Loc:LocalPath        = ITS.GetLocalPath()
 Loc:LocalKey         = 'HKCU:\' & ITS.GetLocalKey()
 Loc:LocalCSIDLNumber = SFQ.FolderIDValue
 Display()
```

See also:
FilesCopied 70
PathString 70
SBGlobalRegistryKey 73
SBLocalRegistryKey 73

3.27.4.18 ShowHTMLLogFile

Prototype: **None**

Returns The method does not return a value.

This method uses the OpenURL method of the Shell Class 87 to open the Build Report html file. The location and name of the Build Report is stored in the SBHtmlLogFile 73 property. The location is retrieved in the CompileSBProject 76 method and also in the BuildCommandLine 75 method.

Example:

```
R =  ITS.CompileSBProject(Loc:SBProjectToCompile)
ITS.ODS('Return value: ' & R)
Case R
Of 1
  If Message('The project was compiled successfully.  ' &|
             'Do you want to view the Build HTML file?',|
             'Project compiled successfully',ICON:Exclamation,|
              BUTTON:No+BUTTON:Yes,BUTTON:Yes) = BUTTON:Yes
    ITS.ShowHTMLLogFile
  End
Of 0
  If Message('Error occured while compiling the project.  ' &|
             'Do you want to view the compile log?',|
             'Errors occurred',ICON:Hand,|
             BUTTON:Yes+BUTTON:No,BUTTON:Yes) = BUTTON:Yes
    ITS.ShowLogFile(IniMGR)
```

```
  End
Of -1
  Message('Both SetupBuilder 5 and SetupBuilder 6 are installed ' &|
          'on this machine.  In that case only SetupBuilder 6.x is ' &|
          'supported and a SetupBuilder 5.x project can not be compiled.',|
          'SetupBuilder 5 and 6 detected',ICON:Hand)
Of -2
  Message('The Command line was too long.',|
          'Command Line is too long',ICON:Hand)

End
```

See also:
BuildCommandLine 75
CompileSBProject 76
SBHtmlLogFile 73
Shell Class 87

3.27.4.19 ShowLogFile - Window

SetupBuilder Class - Methods

Prototype: **(INIClass pINIMgr)**

pINIMgr Global INI Manager class that is used to store window size and position.

Returns The method does not return a value

This method can be called to open and view the error log file generated if the compile process failed. That happens only if the CompileSBProject 76 method returns false. This method shows the file in a Clarion window that will save it's size and position using the standard INI Manager class being passed in the pINIMgr parameter. There is another ShowLogFile 84 method that can show the error log in either a window (which does not store it's size and position) or using the associated program by using ShellExecute. The filename for the error log file is stored in the SBErrorLogFile 72 property.

Example:

```
R =  ITS.CompileSBProject(Loc:SBProjectToCompile)
Case R
Of 1
  If Message('The project was compiled successfully.  ' &|
             'Do you want to view the Build HTML file?',|
             'Project compiled successfully',ICON:Exclamation,|
              BUTTON:No+BUTTON:Yes,BUTTON:Yes) = BUTTON:Yes
    ITS.ShowHTMLLogFile
  End
Of 0
  If Message('Error occured while compiling the project.  ' &|
             'Do you want to view the compile log?',|
             'Errors occurred',ICON:Hand,|
             BUTTON:Yes+BUTTON:No,BUTTON:Yes) = BUTTON:Yes
    ITS.ShowLogFile(IniMGR)
  End
Of -1
  Message('Both SetupBuilder 5 and SetupBuilder 6 are installed ' &|
          'on this machine.  In that case only SetupBuilder 6.x is ' &|
          'supported and a SetupBuilder 5.x project can not be compiled.',|
          'SetupBuilder 5 and 6 detected',ICON:Hand)
Of -2
  Message('The Command line was too long.',|
          'Command Line is too long',ICON:Hand)

End
```

See also:
CompileSBProject 76
SBErrorLogFile 72
ShowLogFile - ShellExecute 84

3.27.4.20 ShowLogFile - ShellExecute

Prototype: **(Byte pOpenInWindow=True)**

pOpenInWindow Indicates if the logfile should be opened in a Clarion window or if it should be opened with associated program using ShellExecute.

Returns The method does not return a value

This method can be called to open and view the error log file generated if the compile process failed. That happens only if the CompileSBProject 76 method returns false. There is another ShowLogFile 84 method that will always show the error log in a Clarion window which is resizable and stores the window size and location in the standard INIManager class, which is passed to it. The filename for the error log file is stored in the SBErrorLogFile 72 property.

Example:

```
R =  ITS.CompileSBProject(Loc:SBProjectToCompile)
Case R
Of 1
  If Message('The project was compiled successfully.  ' &|
             'Do you want to view the Build HTML file?',|
             'Project compiled successfully',ICON:Exclamation,|
              BUTTON:No+BUTTON:Yes,BUTTON:Yes) = BUTTON:Yes
    ITS.ShowHTMLLogFile
  End
Of 0
  If Message('Error occured while compiling the project.  ' &|
             'Do you want to view the compile log?',|
             'Errors occurred',ICON:Hand,|
             BUTTON:Yes+BUTTON:No,BUTTON:Yes) = BUTTON:Yes
    ITS.ShowLogFile(True)    ! Shows in Clarion window
    !ITS.ShowLogFile(False)  ! Would use associated program
  End
Of -1
  Message('Both SetupBuilder 5 and SetupBuilder 6 are installed ' &|
          'on this machine.  In that case only SetupBuilder 6.x is ' &|
          'supported and a SetupBuilder 5.x project can not be compiled.',|
          'SetupBuilder 5 and 6 detected',ICON:Hand)
Of -2
  Message('The Command line was too long.',|
          'Command Line is too long',ICON:Hand)

End
```

See also:
CompileSBProject 76
SBErrorLogFile 72
ShowLogFile - Window 84

3.27.4.21 Construct

Prototype: **None**

Returns The method does not return value.

The Construct method creates a new CompilerVariables 69 property based on the SBCompileVars 68 data type.

Example:

```
None
```

See also:
Destruct 86

3.27.4.22 Destruct

Prototype: **None**

Returns The method does not return value.

The Destruct method frees and disposes of the CompilerVariables property as well as the SBCommandLine property.

Example:

```
None
```

See also:
Construct 86

3.28 Shell Class

3.28.1 Overview Shell Class

```
ITShellClass
Class(ITUtilityClass),TYPE,Module('ITShellClass.clw'),Link('ITShellClass',_ITUtil
LinkMode_),DLL(_ITUtilDllMode_)

GetSpecialFolder           Procedure(Long pFolderID),String
ShellExec                  Procedure(Long pW, String pOp, String pFile, <String
pParam>, |
                                     <String pDir>, Long
pShow=IT_SW_SHOWNORMAL),Long,PROC
ITShellExec                Procedure(String pFile, <String pOp>, <String pParam>,
<String pDir>, Long pShow=IT_SW_SHOWNORMAL),Long,PROC
ShellExecEx                Procedure(*IT_SHELLEXECUTEINFO pShellExecInfo),IT_BOOL
OpenURL                    Procedure(String pURL),VIRTUAL             !! AB
2006-03-06
ShowFilePropertyWindow     Procedure(String pFileName)
AboutShell                 Procedure(String pApp, Long pW=IT_NULL, <String
pOther>, Long pIcon=IT_NULL)
GetEnvVar                  Procedure(String pEnvVar), String
GetAssociatedProg          Procedure(String pFileName),String
APIErrorHandler            Procedure(String pCaption),VIRTUAL
                         End
```

3.28.2 Overview Shell Class

3.29 String Class

3.29.1 Overview String Class

The string class has some very powerful string methods, including methods to read an entire file into a string buffer and write a string buffer to a file. It can also parse a string into lines or into words.

```
ITStringClass
Class(ITUtilityClass),TYPE,Module('ITStringClass.clw'),Link('ITStringClass',_ITUt
ilLinkMode_),DLL(_ITUtilDllMode_)
Lines                       &ITLinesQ
Words                       &ITWordQ
TempS                       &String,PRIVATE
ResStr                      &String,PRIVATE
FileString                  &String,PRIVATE
FileBuffer                  &String,PRIVATE

AddIntoParentheses          Procedure(String pOriginal, String pAddition,
<*CString pSeparator>),String
AllocateFileString          Procedure(Long pBytesToAllocate)
CombineFieldName            Procedure(String pFieldName, String pPrefix, <String
pTableName>),String
CompactString               Procedure(String pOriginal, Byte
pUpperCase=False),String
CompareAndExtract           Procedure(String pOriginal, String pSearchFor),String
DebugLines                  Procedure
DepunctuateString           Procedure(*String pS, Byte pAllowDigits=False)
DisposeFileString           Procedure
FileToString                Procedure(String pFileName),String
FreeString                  Procedure(Byte pWords=0)
GetFieldPrefix              Procedure(String pFieldName),String
GetLine                     Procedure(Long pIndex),String
MatchParenthesis            Procedure(String pS),Short
PadString                   Procedure(String pStr, String pPad, Short pLen, Byte
pStart=0),String
ReadFileToString            Procedure(String pFileName),Long,PROC
SplitFieldName              Procedure(String pFieldName, <*String pPrefix>),String
SplitString                 Procedure(String pStr, String pDelimiter),Long,PROC
StringToLines               Procedure(String pS),Long,PROC
StringToWords               Procedure(String pS, Byte pCount=True, Byte
pCaseSensitive=False),Long,PROC
StripParenthesis            Procedure(String pTxt, <String pParLeft>, <String
pParRight>),String
UseEither                   Procedure(String pS1, String pS2, Byte
pFavourite=1),String
WriteStringToFile           Procedure(String pFileName, String pContent),Long,PROC
Construct                   Procedure
Destruct                    Procedure
                          End
```

3.29.2 Data Types String Class

The String class uses two datatypes, both queues, that are used to store parsed lines and words.

ITLinesQ 89
ITWordQ 89

3.29.2.1 ITWordQ

The ITWordQ is used in the

```
ITWordQ                 QUEUE,TYPE
Word                      CString(61)
Len                       Byte
Counter                   Long
                        END
```

3.29.2.2 ITLinesQ


```
ITLinesQ                QUEUE,TYPE
OL                        CString(1025)
Len                       Long
                        END
```

3.29.3 Properties

The String Class has two public properties and 4 private properties.

Lines 89	&ITLinesQ
Words 89	&ITWordQ
TempS 90	&String,PRIVATE
ResStr 89	&String,PRIVATE
FileString 89	&String,PRIVATE
FileBuffer 89	&String,PRIVATE

3.29.3.1 Lines

Enter topic text here.

3.29.3.2 Words

Enter topic text here.

3.29.3.3 Private Properties

3.29.3.3.1 FileBuffer

Enter topic text here.

3.29.3.3.2 FileString

Enter topic text here.

3.29.3.3.3 ResStr

Enter topic text here.

3.29.3.3.4 TempS

Enter topic text here.

3.29.4 Methods String Class

The String Class has 22 methods plus constructor and destructor.

```
Construct                  Procedure
Destruct                   Procedure

AddIntoParentheses         Procedure(String pOriginal, String pAddition,
<*CString pSeparator>),String
AllocateFileString         Procedure(Long pBytesToAllocate)
CombineFieldName           Procedure(String pFieldName, String pPrefix, <String
pTableName>),String
CompactString              Procedure(String pOriginal, Byte
pUpperCase=False),String
CompareAndExtract          Procedure(String pOriginal, String pSearchFor),String
DebugLines                 Procedure
DepunctuateString          Procedure(*String pS, Byte pAllowDigits=False)
DisposeFileString          Procedure
FileToString               Procedure(String pFileName),String
FreeString                 Procedure(Byte pWords=0)
GetFieldPrefix             Procedure(String pFieldName),String
GetLine                    Procedure(Long pIndex),String
MatchParenthesis           Procedure(String pS),Short
PadString                  Procedure(String pStr, String pPad, Short pLen, Byte
pStart=0),String
ReadFileToString           Procedure(String pFileName),Long,PROC
SplitFieldName             Procedure(String pFieldName, <*String pPrefix>),String
SplitString                Procedure(String pStr, String pDelimiter),Long,PROC
StringToLines              Procedure(String pS),Long,PROC
StringToWords              Procedure(String pS, Byte pCount=True, Byte
pCaseSensitive=False),Long,PROC
StripParenthesis           Procedure(String pTxt, <String pParLeft>, <String
pParRight>),String
UseEither                  Procedure(String pS1, String pS2, Byte
pFavourite=1),String
WriteStringToFile          Procedure(String pFileName, String pContent),Long,PROC
```

3.30 Utility Class

3.30.1 Overview — Utility Class

The Utility Class derives from the Windows Class[106]. It adds several lower level functions.

The Utility class has several very useful functions to do various things, such as create nested directories, format error message strings with or without file errors and many more. The Utility Class has it's own Construct[103] and Destruct[117] methods that set up queues that are used by the class.

```
ITUtilityClass
Class(ITWindowsClass),TYPE,Module('ITUtilityClass.clw'),Link('ITUtilityClass',_IT
UtilLinkMode_),DLL(_ITUtilDllMode_)


MSQ[92]                          &IT_MS_Q
MultiFileSelPath[93]             CString(1025)

ColorToHtml[93]                  Procedure(Long pColorValue),String
CreateDirectories[94]            Procedure(String pDirectories, String
pStartDir),Long
DirectoryExists[95]              Procedure(String pDirectory),Byte
ErrorMsg[95]                     Procedure(Byte pStdErr=True, Byte pFilErr=False,
<String pSeparator>),String
FirstNonSpace[96]                Procedure(String pS),Long
GetClockFromString[97]           Procedure(String pClock),Long
GetClockValue[97]                Procedure(Long pClock, Byte pIntervalMin, Byte
pRoundUp),Long
GetFormatted100sec[99]           Procedure(Long pTime,<String pDelimiter>,<String
pTimeFormat>),String
GetCommandLineParam[98]          Procedure(String pFlag),String
GetExcelDate[99]                 Procedure(Long pClarionDate),Long
GetFileInfo[99]                  Procedure(String pFileName, Long pAtt=0, <*ANY
pDate>, <*ANY pTime>, <*Long pSize>, <*Long pAttrib>)
GetHour[100]                     Procedure(Long pClock),Long
GetMinute[101]                   Procedure(Long pClock),Long
GetUnixDateTime[101]             Procedure(*DECIMAL pUnixTime, <*Long pTime>),Long
HtmlToColor[101]                 Procedure(String pHtmlColor),Long
LongToHex[102]                   Procedure(Long pLong),String
MultiFileSelect[102]             Procedure(String pMfS),Long,PROC
GetCRC32[98]                     Procedure(String pBuffer),Ulong
CompareCRC32[94]                 Procedure(String pBuffer, Ulong pCRC),Byte
Construct[117]                   Procedure
Destruct[117]                    Procedure
                         End
```

3.30.2 Equates — Utility Class

The Utility class uses the following Equates in splitting file and path information:

```
FNS_Drive                EQUATE(01h)
FNS_Path                 EQUATE(02h)
FNS_File                 EQUATE(04h)
FNS_Ext                  EQUATE(08h)
FNS_FullPath             EQUATE(FNS_Drive+FNS_Path)
FNS_FileName             EQUATE(FNS_File+FNS_Ext)
```

When using the GetFilePart 25 method these equates are used to determine what parts of the filename are returned.

Example

```
ITU  ITUtilityClass

FN   CString(1025)
FP   CString(1025)
 Code
 FN = 'C:\Clarion6\LibSrc\ABFILE.CLW'
 FP = ITU.GetFilePart(FNS_Drive)          ! Returns 'C:'
 FP = ITU.GetFilePart(FNS_Path)           ! Returns '\Clarion6\LibSrc\'
 FP = ITU.GetFilePart(FNS_File)           ! Returns 'ABFILE'
 FP = ITU.GetFilePart(FNS_Ext)            ! Returns '.CLW'

 FP = ITU.GetFilePart(FNS_File+FNS_Ext)   ! Returns 'ABFILE.CLW'
 FP = ITU.GetFilePart(FNS_Drive+FNS_Path) ! Returns 'C:\Clarion6\LibSrc\'
```

3.30.3 Data Types — Utility Class

The Utility class uses one special data type for multi file selections.

IT_MS_Q 92

See also:
MultiFileSelect 102

3.30.3.1 IT_MS_Q — Utility Class - Data Types

The following queue type is used in Multi File Selection.

```
IT_MS_Q                 QUEUE,TYPE
MSFileN                   CString(1025)
                        END
```

See also:
MultiFileSelect 102

3.30.4 Properties — Utility Class

The Utility class has two public properties.

MSQ 92	&IT_MS_Q 92
MultiFileSelPath 93	CString(1025)

See also:
MultiFileSelect 102

3.30.4.1 MSQ — Utility Class - Properties

Queue used in Multi file Select. It is declared as:

```
MSQ                     &IT_MS_Q 92
```

See also:
MultiFileSelect 102

3.30.4.2 MultiFileSelPath — Utility Class - Properties

This is a CString used in the Multi File Select methods to hold the path for the files.

```
MultiFileSelPath          CString(1025)
```

See also:
MultiFileSelect 102

3.30.5 Methods — Utility Class

The Utility class has 34 methods, including the Constructor and Destructor.

```
Construct 103                  Procedure
Destruct 103                   Procedure

ColorToHtml 93                 Procedure(Long pColorValue),String
CreateDirectories 94           Procedure(String pDirectories, String
pStartDir),Long
DirectoryExists 95             Procedure(String pDirectory),Byte
ErrorMsg 95                    Procedure(Byte pStdErr=True, Byte pFilErr=False,
<String pSeparator>),String
FirstNonSpace 96               Procedure(String pS),Long
GetClockFromString 97          Procedure(String pClock),Long
GetClockValue 97               Procedure(Long pClock, Byte pIntervalMin, Byte
pRoundUp),Long
GetFormatted100sec 99          Procedure(Long pTime,<String pDelimiter>,<String
pTimeFormat>),String
GetCommandLineParam 98         Procedure(String pFlag),String
GetExcelDate 99                Procedure(Long pClarionDate),Long
GetFileInfo 99                 Procedure(String pFileName, Long pAtt=0, <*ANY
pDate>, <*ANY pTime>, <*Long pSize>, <*Long pAttrib>)
GetHour 100                    Procedure(Long pClock),Long
GetMinute 101                  Procedure(Long pClock),Long
GetUnixDateTime 101            Procedure(*DECIMAL pUnixTime, <*Long pTime>),Long
HtmlToColor 101                Procedure(String pHtmlColor),Long
LongToHex 102                  Procedure(Long pLong),String
MultiFileSelect 102            Procedure(String pMfS),Long,PROC
GetCRC32 98                    Procedure(String pBuffer),Ulong
CompareCRC32 94                Procedure(String pBuffer, Ulong pCRC),Byte
```

3.30.5.1 ColorToHTML — Utility Class - Methods

Prototype: **(Long pColorValue),String**

pColorValue Clarion color value

Returns String containing the correct html color value as #RRGGBB

This method takes an ordinary Clarion 24bit color value and turns it into a standard HTML color string in the format '#RRGGBB' See the TestUtilityClass procedure in the Example program 164

Example:

```
Col        Long
HTMLColor  String(7)
 Code
 If ColorDialog('Select color',Col)
   HTMLColor = ITU.ColorToHTML(Col)
   ?HTMLColor {Prop:FontColor} = Col
   Display
 End
```

See also:
HTMLToColor 101

3.30.5.2 CompareCRC32 — Utility Class - Methods

Prototype: **(String pBuffer, Ulong pCRC),Byte**

pBuffer String to compare CRC value for

pCRC The CRC value to compare the CRC from the buffer to.

Returns True or false depending on if the CRC values match.

This methods takes a string buffer, calculates the CRC value for it and then compares it to the passed CRC value. If the calculated and passed CRC match the method returns True. If they do not match, the method returns false.

Example:

```
Loc:CRC        ULong
Loc:TestString String(20)
Loc:CRCString  String(20)
 Code
 Loc:TestString = 'Icetips Creative'
 Loc:CRCString  = 'Icetips Creative'

 Loc:CRC = ITU.GetCRC32(Loc:TestString)
 If ITU.CompareCRC32(Loc:CRCString,Loc:CRC)
   Message('The strings match:)' &|
           '|TestString = ' & Loc:CRC & |
           '|CRCString  = ' & ITU.GetCRC32(Loc:CRCString),'String
match:)',ICON:Exclamation)
 Else
   Message('The strings do NOT match:(' &|
           '|TestString = ' & Loc:CRC  &|
           '|CRCString  = ' & ITU.GetCRC32(Loc:CRCString),'String do NOT
match:(', ICON:Hand)
 End
```

See also:
GetCRC32 98

3.30.5.3 CreateDirectories — Utility Class - Methods

Prototype: **(String pDirectories, String pStartDir),Long**

pDirectories String containing the path to create below the start directory. Any directory or directories that do not exist will be created by the method. The string does not need to start with a backslash.

pStartDir String containing the full path to the start directory. This directory must exist.

Returns The number of directories created.

This is a very powerful function that will create as many nested directories as you want.

Example:

```
CurrentUser   CString(101)
ITU           ITUtilityClass
 Code
 ITU.CreateDirectories('Data\Temp',Path())
 CurrentUser = 'John'
 ITU.CreateDirectories('User\' & CurrentUser & '\Data\Temp\Stuff',Path())
```

In the first example, if Path() is C:\Clarion this would result in C:\Clarion\Data\Temp to be created. In the second example, if Path() is C:\Clarion this would result in C:\Clarion\John\Data\Temp\Stuff to be created.

This method is extremely useful when multiple levels of directories is needed. If any of the directories in pDirectories does not exist, it will be created. Directories in pStartDir will not be created, i.e. pStartDir must exist.

Note: *In Beta 2, there was potentially dangerous code in this method that changed the path using SetPath() This code has been removed in the Beta 3 release and this method should be perfectly safe.*

3.30.5.4 DirectoryExists — Utility Class - Methods

Prototype: **(String pDirectory),Byte**

pDirectory Full path to the directory to check.

Returns True or false depending on if the directory exists or not

This method uses the Exists function to check if the directory exists or not.

Example:

```
P    CString(1025)
ITU  ITUtilityClass
 Code
 P = 'C:\Clarion\Apps\MyApp'
 If ITU.DirectoryExists(P)
   Copy ('myfile.txt',P & 'myfile.txt')
 End
```

3.30.5.5 ErrorMsg — Utility Class - Methods

Prototype: **(Byte pStdErr=True, Byte pFilErr=False, <String pSeparator>),String**

pStdErr Indicates if error information from ErrorCode() and Error() should be included in the error string. This defaults to True

pFilErr Indicates if error information from FileErrorCode() and FileError() should be included in the string. This defaults to False.

[pSeparator] Indicates a separator string used between the standard errors and the file errors. If omitted it defaults to a single space character.

Returns String containing formatted error message.

This method returns a formatted error message in the format of:

(ErrorCode) Error Separator (FileErrorCode) FileError

If you are going to show the error string in a Message() function, then you could pass '<13,10>' as separator to put the standard error and file error on separate lines.

Note:
When working with SQL drivers any SQL errors that come from the backend database engine trigger errorcode 90. In that case the FileErrorCode() and FileError() will contain extended error information that comes from the database engine. If you are dealing with SQL engines, always pass True as the second parameter to make sure that you will get that extended error information. Otherwise you will simply get error code 90 - File Driver Error, which doesn't really tell you anything about the actual problem. If you know that you are only going to be getting errors from the database engine, i.e. error 90, then you can simply leave the first parameter, pStdErr as false. Then this method will only return the extended error information from the backend.

Example:

```
ITU  ITUtilityClass
 Code
 Add(MyFile)
 Case ErrorCode()
 Of 90
   Message('SQL Error: ' & ITU.ErrorMsg(False,True)
 Else
   Message('Error: ' & ITU.ErrorMsg(True,True,'<13,10>')
 End
```

3.30.5.6 FirstNonSpace

Prototype: **(String pS),Long**

pS String to check

Returns The first non space character position in the string pS

This method uses the internal StrSpn function to find the first non-space character in the string passed as pS. For more information about StrSpn please refer to the web, for example http://www.cplusplus.com/ref/cstring/strspn.html

Example:

```
S    String(255)
I    Long
ITU  ITUtilityClass
 Code
```

```
 S = ' This is a string'
 I = ITU.FirstNonSpace(S)
```

In this case I would be equal to 2.

3.30.5.7 GetClockFromString

Utility Class - Methods

Prototype: **(String pClock),Long**

pClock String formatted as HH:MM:SS

Returns Time value in 1/100 seconds.

This function takes a string formatted as HH:MM:SS and returns the time value from it, i.e. hundredths of seconds elapsed from midnight. See the Clock() function in Clarion.

Example:

```
S    String(10)
C    Long
ITU  ITUtilityClass
 Code
 S = '13:54:10'
 C = ITU.GetClockFromString(S)
```

C would now be (13*60*60*100) + (54*60*100) + (10*100) or 5,005,000

See also:
GetClockValue 97

3.30.5.8 GetClockValue

Utility Class - Methods

Prototype: **(Long pClock, Byte pIntervalMin, Byte pRoundUp),Long**

pClock Time value.

pIntervalMin Interval in minutes.

pRoundUp Rounds up if True or down if false.

Return Time value after rounding up or down to the given interval.

This function takes a clock value and rounds it up or down to the nearest clock indicated in the pIntervalMin. This is very useful for schedule type of calculation where an appointment should be scheduled for example with 15 minute intervals starting at the hour, i.e. at 18:00, 18:15, 18:30 and 18:45. This function makes it very easy to do that. Simply give it the time, interval in minutes and if it should round up or down and it will return the correct time value.

Example:

```
C1   Long
C2   Long
I    Byte
R    Byte
ITU  ITUtilityClass
```

```
 Code
 C1 = ITU.GetClockFromString('18:16:00')
 I = 15 ! 15 minute interval
 R = True
 C1 = ITU.GetClockValue(C1,I,R)
 I = 15 ! 15 minute interval
 R = False
 C2 = ITU.GetClockValue(C1,I,R)
```

C1 would be equal to 18:30:00 but C2 would be equal to 18:15:00

See also:
GetClockFromString 97

3.30.5.9 GetCommandLineParam — Utility Class - Methods

Prototype: **(String pFlag),String**

pFlag Command line flag to test for.

Returns The command line flag or parameter

This function parses out a flag in the command line parameters passed when the program starts up. If an equal sign is used in the parameter, for example /N=Test, the function will return Test only.

Example:

```
C    String(255)
ITU  ITUtilityClass
 Code
 ! The program was started with myprog.exe /N=Test /F=myfile.tps /Q

 C = ITU.GetCommandLineParam('/N')   ! Will return 'Test'
 C = ITU.GetCommandLineParam('/F')   ! Will return 'myfile.tps'
 C = ITU.GetCommandLineParam('/Q')   ! Will return '/Q'
```

See also:
EXEName 21 - Coreclass
ProgPath 21 - Coreclass
ProgramCommandLine 21 - CoreClass

3.30.5.10 GetCRC32 — Utility Class - Methods

Prototype: **(String pBuffer),Ulong**

pBuffer String to calculate CRC value for.

Returns The CRC32 value for the buffer string.

This methods takes a string buffer and calculates and returns the CRC value for it.

Example:

```
Loc:CRC        ULong
Loc:TestString String(20)
Loc:CRCString  String(20)
```

```
 Code
 Loc:TestString = 'Icetips Creative'
 Loc:CRCString  = 'Icetips Creative'

 Loc:CRC = ITU.GetCRC32(Loc:TestString)
 If ITU.CompareCRC32(Loc:CRCString,Loc:CRC)
   Message('The strings match:)' &|
           '|TestString = ' & Loc:CRC & |
           '|CRCString  = ' & ITU.GetCRC32(Loc:CRCString),'String
match:)',ICON:Exclamation)
 Else
   Message('The strings do NOT match:(' &|
           '|TestString = ' & Loc:CRC  &|
           '|CRCString  = ' & ITU.GetCRC32(Loc:CRCString),'String do NOT
match:(', ICON:Hand)
 End
```

See also:
CompareCRC32 94

3.30.5.11 GetExcelDate — Utility Class - Methods

Prototype: **(Long pClarionDate),Long**

pClarionDate Standard Clarion date.

Returns Returns Excel based date value.

A date in Excel is a value that is exactly 36161 days less than the Clarion date. So this method simply subtracts that value from the standard Clarion date value and the resulting value can be used as a date value in MS Excel.

Example:

```
ED  Long
 Code
 ED = ITU.GetExcelDate(Today())
```

See also:
GetUnixDateTime 101

3.30.5.12 GetFormatted100sec — Utility Class - Methods

Enter topic text here.

3.30.5.13 GetFileInfo — Utility Class - Methods

Prototype: **(String pFileName, Long pAtt=0, <*ANY pDate>, <*ANY pTime>, <*Long pSize>,<*Long pAttrib>)**

pFileName Full path name of the file to get information about.

pAtt Specifies the ff_ file attributes 167.

[pDate] The date of the file. This can be a LONG or a DATE variable. You must specify variable in the call or omit this parameter.

[pTime] The time of the file. This can be a LONG or a TIME variable. You must specify variable in the call or omit this parameter.

[pSize] The size of the file. You must specify variable in the call or omit this parameter.

[pAttrib] The attributes of the file. You must specify variable in the call or omit this parameter.

This method will retrieve information about the file passed as pFileName. Note that date, time, size and attributes are passed by address not value so it is up to you to create variables that are used when calling this method. pDate and pTime can be LONG or DATE/TIME variables so this will work with program variables as well as table columns from SQL tables etc.

this method does not return a value, rather it returns multiple values.

Example:

```
FD   Long
FAT  Long
FT   Long
FS   Long
FA   Long
F    CString(1025)
ITU  ITUtilityClass
 Code
 FAT = ff_:Normal
 F = 'C:\Clarion\Apps\MyTest\Test.app' ! Jan 1, 2006 at 13:01:40, 12345 bytes.
 ITU.GetFileInfo(F,FAT,FD,FT,FS,FA)
 ! FD is now equal to the date value for Jan 1, 2006
 ! FT is now equal to the clock value for '13:01:40'
 ! FS is now equal to 12345
 ! FA is now equal to ff_:Normal
```

3.30.5.14 GetHour

Prototype: **(Long pClock),Long**

pClock The time value.

Returns The hour part of the time value in pClock.

This returns just the hour part of the time value passed in pClock. This is handy to have when you need to just know the hour part.

Example:

```
T    Long
H    Byte
ITU  ITUtilityClass
 Code
 T = ITU.GetClockFromString('11:10:10')
 H = ITU.GetHour(T)  ! returns 11
```

See also:

GetMinute 101
GetClockFromString 97

3.30.5.15 GetMinute

Utility Class - Methods

Prototype: **(Long pClock),Long**

pClock The time value.

Returns The minute part of the time value in pClock.

This returns just the minute part of the time value passed in pClock. This is handy to have when you need to just know the minute part.

Example:

```
T    Long
M    Byte
ITU  ITUtilityClass
 Code
 T = ITU.GetClockFromString('11:10:10')
 M = ITU.GetMinute(T)  ! returns 10
```

See also:

GetHour 100
GetClockFromString 97

3.30.5.16 GetUnixDateTime

Utility Class - Methods

Prototype: **(*DECIMAL pUnixTime, <*Long pTime>),Long**

pUnixTime Parameter containing time value from a Unix system.

[pTime] Optional parameter that receives the time part of the Unix Time.

Returns Returns Clarion compatible date value.

This method takes a Unix date time, which is the number of 1/100 seconds since January 1, 1970 and returns the correct Clarion date value. It can also optionally accept a second parameter which upon return from this method will contain a Clarion compatible time value.

Example:

```
(none)
```

See also:
GetExcelDate 99

3.30.5.17 HTMLToColor

Utility Class - Methods

Prototype: **(String pHtmlColor),Long**

pHtmlColor HTML color value in the format of #RRGGBB

Returns Standard Clarion color value.

This method takes a standard HTML color value string in the form of #RRGGBB and turns it into a

standard Clarion color value.

Example:

```
HTMLCol String(7)
Col     Long
 Col = ITU.HTMLToColor(HTMLCol)
```

See also:
ColorToHTML 93

3.30.5.18 LongToHex

Prototype: **(Long pDecValue),String**

pDecValue Decimal value to get Hexadecimal value from

Returns The hexadecimal value of pDecValue

This method uses the internal sPrintf function to format the decimal value to a hexadecimal string type with 8 characters.

Example:

```
D    Long
H    String(10)
ITU  ITUtilityClass
 Code
 D = 123456
 H = ITU.LongToHex(L)  ! Returns 3039
```

3.30.5.19 MultiFileSelect

Prototype: **(String pMfS),Long**

pMfS String containing multiple file selection from the Clarion FileDialog or FileDialogA functions.

Returns Number of files selected

This function is very useful when you use FileDialog or FileDialogA to allow users to open multiple files. Then the selection is returned in a pipe delimited string where the first filename contains the path and the rest only contains the filenames. Example:

'C:\Clarion\Apps\Tests\Myapp.app|otherapp.app|thirdapp.app'

This method splits it up and puts this into the MSQ 92 queue where each entry contains the full path and filename of each selected file.

Example:

```
Fn   CString(10001)
I    Long
```

```
ITU  ITUtilityClass
 Code
 If FileDialog('Select files',Fn,'All Files
(*.*)|*.*',FILE:KeepDir+FILE:Multi+FILE:LongName)
   ITU.MultiFileSelect(FN)
   Loop I = 1 To Records(ITU.MSQ)
     Get(ITU.MSQ,I)
     ! Do something with the filename
   End
 End
```

3.30.5.20 Construct

Utility Class - Methods

Prototype: **None**

The constructor creates a new instance of the SELF.MSQ 92:

```
SELF.MSQ &= NEW IT_MS_Q
```

3.30.5.21 Destruct

Utility Class - Methods

Prototype: **None**

The Destructor disposes of the SELF.MSQ 92 queue:

```
If Not SELF.MSQ &= NULL
  Free(SELF.MSQ)
  Dispose(SELF.MSQ)
End
```

3.31 Version Class

3.31.1 Overview Version Class

```
ITVersionClass
Class(ITShellClass),TYPE,Module('ITVersionClass.clw'),Link('ITVersionClass',_ITUt
ilLinkMode_),DLL(_ITUtilDllMode_)

VersionNames               &ITVersionNameQueue
VersionInfo                &ITVersionInfoQueue
FileHasVersionInfo         Byte
FileExists                 Byte
HideDebugView              Byte

RetrieveFromSelf           Procedure(),Byte
RetrieveFromFile           Procedure(String pFile),Byte
QueryValue                 Procedure(Long pPtrBuffer, *Long pBufferSize, String
pName),String ! Returns value of the valuename
GetLanguageString          Procedure(Long pPtr),String
GetVersionInfo             Procedure(String pValueName),String ! Returns
ValueInfo
GetDisplayName             Procedure(String pKeyName),String
LoadVersionNames           Procedure
AddVersionName             Procedure(String pVerName, String pUseName)
AddClarionResources        Procedure
Construct                  Procedure
Destruct                   Procedure
PTD                        Procedure(String pS, Byte pHideDebug=False),VIRTUAL
                         End
```

3.31.2 Properties Version Class

Enter topic text here.

3.31.3 Methods Version Class

Enter topic text here.

3.32 Window Manager Class

3.32.1 Overview Window Manager Class

```
ITWindowManagerClass
CLASS(WindowManager),TYPE,Module('ITWindowManagerClass.clw'),Link('ITWindowManage
rClass',_ITUtilLinkMode_),DLL(_ITUtilDllMode_)

ThreadClass               &ITGlobalThreadClass
ErrorClass                &ErrorClass
WindowRef                 &Window
Init                       Procedure(Window pWIN, ITGlobalThreadClass
pThreadClass, ErrorClass pErrorClass)
Kill                       PROCEDURE,PROC,BYTE,VIRTUAL    ! Level:Notify means
dead already
TakeWindowEvent            PROCEDURE,VIRTUAL,BYTE,PROC
                         END
```

3.32.2 Properties Window Manager Class

Enter topic text here.

3.32.3 Methods Window Manager Class

Enter topic text here.

3.33 Windows Class

3.33.1 Overview Windows Class

The Windows Class is the second class in the hierarchy and is derived from the Core Class 18. The Windows class contains various useful functions that call on various core API functions.

```
ITWindowsClass
Class(ITCoreClass),TYPE,Module('ITWindowsClass.clw'),Link('ITWindowsClass',_ITUti
lLinkMode_),DLL(_ITUtilDllMode_)

AppframeClientHandle 109       Long
ChildWindows 109               &ChildWindowQ 108
FrameColor 115                 Long
IsVista 109                    Byte !! True if MajorVersion => 6
IsWindowOnTop 109              Byte
MajorVersion 110               Long
MinorVersion 110               Long
ModuleWindows 111              &ChildWindowQ 108
SaveNewBrush 112               Long
SaveOldBrush 112               Long
ThemedControls 112             &tThemedControls 108
TopWindows 112                 &ChildWindowQ 108
VersionBuildNr 113             Long
VersionInformation 113         String(128)
VistaHasUAC 115                Byte
VersionPlatformID 114          Long
W95HiBuildNr 115               Short
W95LoBuildNr 115               Short
WindowStyle 116                Long

APIErrorHandler 117            Procedure(String pCaption),VIRTUAL
EnumChildWin 117               Procedure(Long phWnd),Long  ! Returns the number of
enumerated child windows
EnumModuleWin 118              Procedure(String pModuleName),Long
EnumTopWin 119                 Procedure(),Long  ! Returns the number of enumerated
top windows
FindWindow 119                 Procedure(String pCaption, Byte pFindInCaption=True,
Byte pFullMatch=False),Long,PROC,VIRTUAL  ! Returns hwnd or 0
GetBaseControlName 121         Procedure(Long pFEQ),String
GetBaseControlName 121         Procedure(String pLabel, Byte pUpper),String
GetCommandLineLen 121          Procedure(),Long
GetControlName 122             Procedure(Long pFEQ, Byte pRemoveQM=0),String
GetDialogUnit 122              Procedure(Byte pVertical),Long
GetExeFromWindowHandle 123     Procedure(Long pHwnd),String
GetPIDFromWindowHandle 123     Procedure(Long pHwnd),IT_DWORD
GetPixelHeight 124             Procedure(Long pFEQ),Long
GetPixelPos 124                Procedure(Long pFEQ, Long pC),Long
GetPixelPosition 125           Procedure(Long pFEQ,<*Long pX>,<*Long pY>,<*Long
pW>,<*Long pH>)
GetPixelWidth 125              Procedure(Long pFEQ),Long
GetPixelXPos 126               Procedure(Long pFEQ),Long
GetPixelYPos 126               Procedure(Long pFEQ),Long
GetPopupXY 126                 Procedure(Long pFEQ,<*Long pX>,<*Long pY>)
GetScreenBaseDPIRatio 127      Procedure(),Real
GetScreenDPI 127               Procedure(),Long
GetScreenDPIRatio 128          Procedure(),Real
GetScreenX 128                 Procedure(Long pFEQ),Long
```

```
GetScreenY 128                  Procedure(Long pFEQ),Long
GetSysMetrics 129               Procedure(Long pIndex),Long  ! Wrapper for
GetSystemMetrics
GetTaskbarHeight 129            Procedure(),Long
GetThemedPanelFEQ 129           Procedure(Long pPanelFEQ),LONG
GetWindowVersion 130            Procedure(),String,PROC
MakeLangID 131                  Procedure(UShort usPrimaryLanguage, UShort
usSubLanguage),UShort
PlaceControlForDPI 131          Procedure(Long pFEQ, <Long pDesignDPI>)  ! Sets X,
Y, W and H
RedrawClientArea 132            Procedure
RemoveWindowColor 132           Procedure(),BYTE
ResizeControlForDPI 132         Procedure(Long pFEQ, <Long pDesignDPI>)  ! Sets W
and H
SetControlFonts 133             Procedure(Long pFrom, Long pTo)
SetControlPositions 133         Procedure(Long pFrom, Long pTo)
SetControlProp 133              Procedure(Long pFEQ, Long pProperty, String pValue)
SetPixelHeight 134              Procedure(Long pFEQ, Long pValue)
SetPixelPos 134                 Procedure(Long pFEQ, Long pC, Long pValue)
SetPixelPosition 135            Procedure(Long pFEQ,<Long pX>,<Long pY>,<Long
pW>,<Long pH>)
SetPixelWidth 136               Procedure(Long pFEQ, Long pValue)
SetPixelXPos 136                Procedure(Long pFEQ, Long pValue)
SetPixelYPos 136                Procedure(Long pFEQ, Long pValue)
SetToolboxCaption 137           Procedure(Long phwnd, Byte pSetOn=True)
SetWindowColor 138              Procedure(Long pColor)
SetWindowNotOnTop 138           Procedure
SetWindowOnTop 138              Procedure
SetWindowPosition 139           Procedure(Long pX, Long pY, Byte pSetPixels=True)
SetWindowSize 139               Procedure(Long pWidth, Long pHeight, Byte
pSetPixels=True)
ThemeAPanel 139                 Procedure(Long pPanelFEQ)
UsesClearType 140               Procedure(),Byte  !! Returns true/false if the
computer is using ClearType
UsingLargeFonts 140             Procedure(),Byte
WindowInfoToODS 141             Procedure(String pProcedureName),VIRTUAL

Construct 117                   Procedure
Destruct 117                    Procedure
                            End
```

For examples of how to use the Window class, please refer to the Example Program 164 procedures for the Windows Class 106.

See also:
Windows Functions on MSDN

3.33.2 Data Types — Windows Class

The Windows class uses one special data type for child window enumeration.

ChildWindowQ 108

See also:
ChildWindows 109
EnumChildWin 117
EnumChildWindowsProc 142

3.33.2.1 tThemedControls

The **tThemedControls** queue is used to keep track of controls that have been themed with the ThemeAPanel[139] method. Use the GetThemedPanelFEQ[129] to retrieve the original panel FEQ.

```
tThemedControls    QUEUE,PRE(tThemeControls),TYPE
OriginalFEQ          Long
NewFEQ               Long
                   End
```

See also:
ThemeAPanel[139]
GetThemedPanelFEQ[129]

3.33.2.2 ChildWindowQ

The **ChildWindowQ** is used to store the child window enumeration information, such as the handle, style, extra style bits and the window caption (text).

```
ChildWindowQ              QUEUE,TYPE
CwHwnd                      Long
Style                       Long
ExStyle                     Long
Text                        CString(1025)
UpperText                   CString(1025)
ModuleName                  CString(1025)
ModuleEXEName               CString(101)  ! Uppercased name of the EXE

                          END
```

The Text contains the caption text of the window. The UpperText contains the same text all upper case. The ModuleName contains the SHORTPATH of the module which the window belongs to. Note that the ModuleName is ONLY filled in the EnumTopWin[119], not in EnumChildWin[117] since all child windows of a process will belong to the main process.

See also:
ChildWindows[109]
EnumChildWin[117]
EnumTopWin[119]
EnumChildWindowsProc[142]
EnumTopWindowsProc[141]

3.33.3 Properties — Windows Class

The Windows class has several public properties.

```
ChildWindows[109]              &ChildWindowQ ! Reference to a Child Windows Queue
IsWindowOnTop[109]             Byte
MajorVersion[110]              Long
MinorVersion[110]              Long
TopWindows[112]                &ChildWindowQ              ! Reference to a Top
Windows Queue
VersionBuildNr[113]            Long
VersionInformation[113]        String(128)
VersionPlatformID[114]         Long
W95HiBuildNr[115]              Short
W95LoBuildNr[115]              Short
```

```
WindowStyle 116          Long
```

3.33.3.1 AppframeClientHandle

Windows Class - Properties

This property is used in the SetWindowColor 138 and RemoveWindowColor 132 methods to store the window client handle:

```
SELF.AppframeClientHandle = 0{Prop:ClientHandle}
```

The RemoveWindowColor 132 is called automatically when EVENT:CloseWindow is fired.

See also:
SetWindowColor 138
RemoveWindowColor 132

3.33.3.2 ChildWindows

Windows Class - Properties

ChildWindows is a queue of type ChildWindowQ 108 that is used in the EnumChildWin 117 method to store information about the enumerated child windows.

```
ChildWindows             &ChildWindowQ ! Reference to a Child Windows Queue
```

See also:
ChildWindowQ 108
EnumChildWin 117
EnumChildWindowsProc 142

3.33.3.3 IsVista

Windows Class - Properties

This property is set in the Constructor 117 method:

```
 SELF.IsVista            = Choose(SELF.MajorVersion 110=>6,True,False)
```

This property is true for Vista and Windows Server 2008 For more information about the windows versions, please see http://msdn.microsoft.com/en-us/library/ms724451(VS.85).aspx and http://msdn.microsoft.com/en-us/library/ms724833(VS.85).aspx

See also:
Constructor 117
GetWindowVersion 130

3.33.3.4 IsWindowOnTop

Windows Class - Properties

This property is set to true or false in the SetWindowOnTop and SetWindowNotOnTop methods. It's value indicates if the window is set to be at the top of the z-order of windows. This means that the window will be on top of all other windows.

Example:

```
ITW  ITWindowsClass
 Code
 ! ...
 If Not ITW.IsWindowOnTop
```

```
   ITW.SetWindowOnTop
 End
```

See also:
SetWindowOnTop 138
SetWindowNotOnTop 138

3.33.3.5 MajorVersion

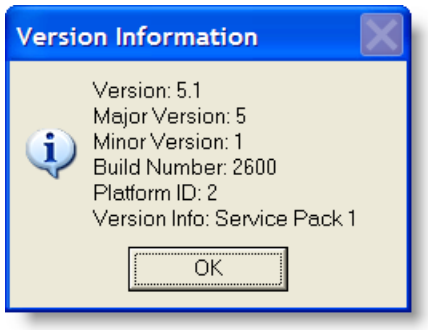
This property is a windows version property and indicates the major version number. For example 5 is the version number of Windows XP.

Value	Meaning
4	Windows NT 4.0, Windows Me, Windows 98, or Windows 95
5	Windows Server 2003 R2, Windows Server 2003, Windows XP, or Windows 2000
6	Windows Vista or Windows Server "Longhorn"

Example:

```
ITW  ITWindowsClass
VS   CString(101)
 Code
 !...
 VS = ITW.GetWindowVersion()
 Message('Version:           ' & VS &|
        '|Major Version:     ' & ITW.MajorVersion &|
        '|Minor Version:     ' & ITW.MinorVersion &|
        '|Build Number:      ' & ITW.VersionBuildNr &|
        '|Platform ID:       ' & ITW.VersionPlatformID &|
        '|Version Info:      ' & ITW.VersionInformation, |
        'Version Information',ICON:Information)
```

On Windows XP Home, service pack 1, the results can be seen in the screenshot below.



See also:
GetWindowVersion 130
MinorVersion 110
VersionBuildNr 113
VersionPlatformID 114
VersionInformation 113

3.33.3.6 MinorVersion

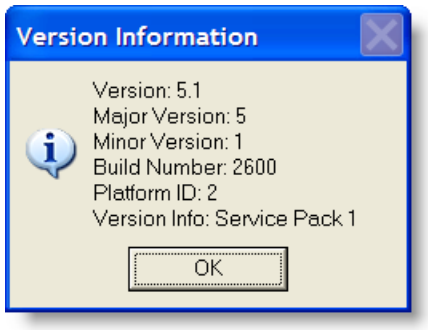
This property is a windows version property and indicates the minor version number. For example 5 is the major version number of Windows XP and 1 is the minor version number of Service Pack 1.

Value	Meaning
0	Windows Vista, Windows Server "Longhorn", Windows 2000, Windows NT 4.0, or Windows 95
1	Windows XP
2	Windows Server 2003 R2, Windows Server 2003, or Windows XP Professional x64 Edition
10	Windows 98
90	Windows ME

Example:

```
ITW  ITWindowsClass
VS   CString(101)
 Code
 !...
 VS = ITW.GetWindowVersion()
 Message('Version:           ' & VS &|
        '|Major Version:    ' & ITW.MajorVersion &|
        '|Minor Version:    ' & ITW.MinorVersion &|
        '|Build Number:     ' & ITW.VersionBuildNr &|
        '|Platform ID:      ' & ITW.VersionPlatformID &|
        '|Version Info:     ' & ITW.VersionInformation, |
        'Version Information',ICON:Information)
```

On Windows XP Home, service pack 1, the results can be seen in the screenshot below.



See also:
GetWindowVersion 130
MajorVersion 110
VersionBuildNr 113
VersionPlatformID 114
VersionInformation 113

3.33.3.7 ModuleWindows

Windows Class - Properties

ModuleWindows is a queue of type ChildWindowQ 108 that is used in the EnumModuleWin 118 method to store information about the enumerated top windows for a specific module. By TopWindows we mean windows that are defined as Top-Level windows. Unlike child windows, top windows do not have parent windows. For a Clarion application this would be the appframe window. ChildWindows 109 would be any windows opened by that appframe window.

```
ChildWindows              &ChildWindowQ ! Reference to a Child Windows Queue
```

See also:

EnumTopWin 119
EnumTopWindowsProc 141

3.33.3.8 SaveNewBrush

Windows Class - Properties

This property is set in the SetWindowColor 138 method and used in the RemoveWindowColor 132 method. It saves the new brush created for the window, which is then restored in RemoveWindowColor 132.

```
SELF.SaveNewBrush = IT_CreateSolidBrush(SELF.FrameColor 115)
```

See also:
SetWindowColor 138
RemoveWindowColor 132

3.33.3.9 SaveOldBrush

Windows Class - Properties

This property is set in the SetWindowColor 138 method and used in the RemoveWindowColor 132 method. It saves the original brush used for the window, which is then restored in RemoveWindowColor 132.

```
SELF.SaveOldBrush = IT_GetClassLong(SELF.AppframeClientHandle 109,
IT_GCL_HBRBACKGROUND)
```

See also:
SetWindowColor 138
RemoveWindowColor 132

3.33.3.10 ThemedControls

Windows Class - Properties

ThemedControls is a queue that is created with the Constructor 117 and added to by the ThemeAPanel 139 method. It contains the FEQ of the panel being themed and the tab Sheet that is created to replace the panel. It is used by the GetThemedPanelFEQ 129 to get the Sheet FEQ that matches the panel FEQ that was replaced. If XPThemes 22 are not present or the window is not themed, both the OriginalFEQ and the NewFEQ will be the same and equal to the Panel FEQ.

```
Add(SELF.ThemedControls,SELF.ThemedControls.OriginalFEQ)
```

See also:
Constructor 117
GetThemedPanelFEQ 129
ThemeAPanel 139

3.33.3.11 TopWindows

Windows Class - Properties

TopWindows is a queue of type ChildWindowQ 108 that is used in the EnumTopWin 119 method to store information about the enumerated top windows. By TopWindows we mean windows that are defined as Top-Level windows. Unlike child windows, top windows do not have parent windows. For a Clarion application this would be the appframe window. ChildWindows 109 would be any windows opened by that appframe window.

```
ChildWindows                &ChildWindowQ ! Reference to a Child Windows Queue
```

See also:

ChildWindowQ 108
EnumTopWin 119
EnumTopWindowsProc 141

3.33.3.12 VersionBuildNr

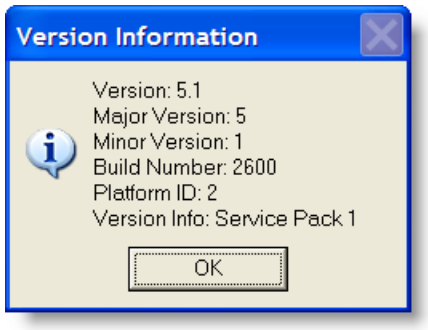
Windows Class - Properties

This property is a windows version property and indicates the build number. On Windows 95/98/ME, the low-order word 115 contains the build number of the operating system. The high-order word 115 contains the major and minor version numbers. On other versions, this contains a build number.

Example:

```
ITW  ITWindowsClass
VS   CString(101)
 Code
 !...
 VS = ITW.GetWindowVersion()
 Message('Version:           ' & VS &|
        '|Major Version:    ' & ITW.MajorVersion &|
        '|Minor Version:    ' & ITW.MinorVersion &|
        '|Build Number:     ' & ITW.VersionBuildNr &|
        '|Platform ID:      ' & ITW.VersionPlatformID &|
        '|Version Info:     ' & ITW.VersionInformation, |
        'Version Information',ICON:Information)
```

On Windows XP Home, service pack 1, the results can be seen in the screenshot below.



See also:
GetWindowVersion 130
MajorVersion 110
MinorVersion 110
VersionPlatformID 114
VersionInformation 113

3.33.3.13 VersionInformation

Windows Class - Properties

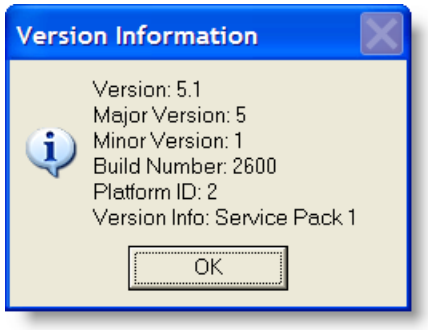
This property is a windows version property and indicates additional version information. This can indicate a service pack or on Windows 95/98/ME it can be additional version information.

Example:

```
ITW  ITWindowsClass
VS   CString(101)
 Code
 !...
 VS = ITW.GetWindowVersion()
 Message('Version:           ' & VS &|
```

```
        '|Major Version:    ' & ITW.MajorVersion &|
        '|Minor Version:    ' & ITW.MinorVersion &|
        '|Build Number:     ' & ITW.VersionBuildNr &|
        '|Platform ID:      ' & ITW.VersionPlatformID &|
        '|Version Info:     ' & ITW.VersionInformation, |
        'Version Information',ICON:Information)
```

On Windows XP Home, service pack 1, the results can be seen in the screenshot below.



See also:
GetWindowVersion 130
MajorVersion 110
MinorVersion 110
VersionPlatformID 114
VersionBuildNr 113

3.33.3.14 VersionPlatformID

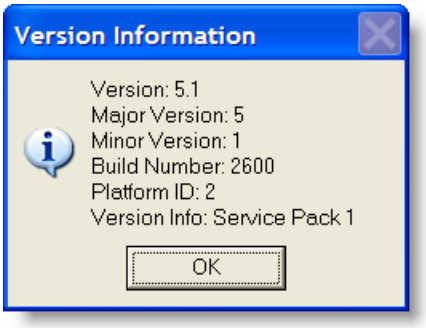
Windows Class - Properties

This property is a windows version property and indicates version platform ID. This is either 1 (IT_VER_PLATFORM_WIN32_WINDOWS)
or 2 (IT_VER_PLATFORM_WIN32_NT), i.e. 1 indicates Windows 95/98/ME and 2 indicates Windows Vista, Windows Server "Longhorn", Windows Server 2003, Windows XP, Windows 2000, or Windows NT.

Example:

```
ITW  ITWindowsClass
VS   CString(101)
 Code
 !...
 VS = ITW.GetWindowVersion()
 Message('Version:           ' & VS &|
        '|Major Version:    ' & ITW.MajorVersion &|
        '|Minor Version:    ' & ITW.MinorVersion &|
        '|Build Number:     ' & ITW.VersionBuildNr &|
        '|Platform ID:      ' & ITW.VersionPlatformID &|
        '|Version Info:     ' & ITW.VersionInformation, |
        'Version Information',ICON:Information)
```

On Windows XP Home, service pack 1, the results can be seen in the screenshot below.



See also:
GetWindowVersion 130
MajorVersion 110
MinorVersion 110
VersionPlatformID 114
VersionBuildNr 113

3.33.3.15 VistaHasUAC

Windows Class - Properties

This property will tell is UAC is turned on. To our knowledge this is the best way to determine if the UAC is on or not.

```
SELF.VistaHasUAC = GetReg(REG_LOCAL_MACHINE,'Software\Microsoft\Windows
\CurrentVersion\Policies\System','EnableLUA')
```

See also:
IsVista 109

3.33.3.16 W95HiBuildNr

Windows Class - Properties

This contains the High-order word of the VersionBuildNr 113. Currently this value is not set.

See also:
VersionBuildNr 113

3.33.3.17 W95LoBuildNr

Windows Class - Properties

This contains the Low-order word of the VersionBuildNr 113. Currently this value is not set.

See also:
VersionBuildNr 113

3.33.3.18 WindowColor

Windows Class - Properties

WindowColor is set in the SetWindowColor 138 method:

```
SELF.WindowColor = pColor
```

It is currently only used in that method.

See also:

SetWindowColor 138

3.33.3.19 WindowStyle
Windows Class - Properties

This is used in SetToolboxCaption 137 to store the Window Style value as returned by the GetWindowLong api call. This is stored in order to be able to change the caption bar to toolbox caption and then later change it back with all the previous styles intact.

See also:
SetToolboxCaption 137

3.33.3.20 WindowsColorChanged
Windows Class - Properties

WindowsColorChanged is set in SetWindowColor 138 and is also used in RemoveWindowColor 132. It is set to either True or False depnding on if the background color for the window has been changed using SetWindowColor 138.

See also:
SetWindowColor 138
RemoveWindowColor 132

3.33.4 Methods
Windows Class

The Windows class has 34 methods, including the Constructor and Destructor.

```
Construct 117                   Procedure
Destruct 117                    Procedure

APIErrorHandler 117             Procedure(String pCaption),VIRTUAL
EnumChildWin 117                Procedure(Long phWnd),Long  ! Returns the number of
enumerated child windows
GetBaseControlName 121          Procedure(Long pFEQ),String
GetBaseControlName 121          Procedure(String pLabel, Byte pUpper),String
GetControlName 122              Procedure(Long pFEQ),String
GetDialogUnit 122               Procedure(Byte pVertical),Long
GetLastAPIError 26              Procedure(<*Long pErrorCode>),String
GetLastAPIErrorCode 26          Procedure(),Long
GetPixelHeight 124              Procedure(Long pFEQ),Long
GetPixelPos 124                 Procedure(Long pFEQ, Long pC),Long
GetPixelPosition 125            Procedure(Long pFEQ,<*Long pX>,<*Long pY>,<*Long
pW>,<*Long pH>)
GetPixelWidth 125               Procedure(Long pFEQ),Long
GetPixelXPos 126                Procedure(Long pFEQ),Long
GetPixelYPos 126                Procedure(Long pFEQ),Long
GetPopupXY 126                  Procedure(Long pFEQ,<*Long pX>,<*Long pY>)
GetScreenBaseDPIRatio 127       Procedure(),Real
GetScreenDPI 127                Procedure(),Long
GetScreenDPIRatio 128           Procedure(),Real
GetScreenX 128                  Procedure(Long pFEQ),Long
GetScreenY 128                  Procedure(Long pFEQ),Long
GetSysMetrics 129               Procedure(Long pIndex),Long  ! Wrapper for
GetSystemMetrics
GetWindowVersion 130            Procedure(),String,PROC
ODS 29                          Procedure(String pS),VIRTUAL
PTD 56                          Procedure(String pS, Byte pHideDebug=False),VIRTUAL
```

ResizeControlForDPI 132	Procedure(Long pFEQ, <Long pDesignDPI>)
SetControlFonts 133	Procedure(Long pFrom, Long pTo)
SetControlPositions 133	Procedure(Long pFrom, Long pTo)
SetControlProp 133	Procedure(Long pFEQ, Long pProperty, String pValue)
SetToolboxCaption 137	Procedure(Long hwnd, Byte pSetOn=True)
SetWindowNotOnTop 138	Procedure
SetWindowOnTop 138	Procedure
UsingLargeFonts 140	Procedure(),Byte

3.33.4.1 APIErrorHandler

Windows Class - Methods

Prototype: **(String pCaption),VIRTUAL**

pCaption Message Caption

This is a placeholder virtual method that is used in the ShellClass 87 and can be used anywhere to create a special API error handler if needed.

3.33.4.2 Construct

Windows Class - Methods

Prototype: **None**

This constructor creates new instances of the ChildWindowQ 108 as ChildWindows 109 and TopWindows 112 properties. It also creates a new

```
 SELF.ChildWindows  &= NEW ChildWindowQ
 SELF.TopWindows    &= NEW ChildWindowQ
 SELF.ModuleWindows &= NEW ChildWindowQ
 IT_PROCESS_ALL_ACCESS = IT_STANDARD_RIGHTS_REQUIRED + IT_SYNCHRONIZE +
IT_PROCESS_ALL_ACCESS_ADDIN
 V = SELF.GetWindowVersion()
 SELF.ThemedControls &= NEW (tThemedControls)
```

See also:
Destruct 117

3.33.4.3 Destruct

Windows Class - Methods

Prototype: **None**

This Destructor cleans up the ChildWindows 109 and TopWindows 112 property queues.

See also:
Construct 117

3.33.4.4 EnumChildWin

Windows Class - Methods

Prototype: **(Long phWnd),Long**

phWnd Window handle to enumerate child windows for, i.e. parent window

Returns Number of child windows

This method is used to enumerate all child windows of a parent window specified in the phWnd parameter. Please note that this does not enumerate non-MDI windows that have been opened by the appframe if the appframe handle is the phWnd passed to this method. It will only enumerate MDI windows. Also note that all controls are considered child windows.

Example:

```
SetupWindow          ROUTINE
 Data
I  Long
 Code
 If ITW.EnumChildWin(pParentHandle)
   Loop I = 1 To Records(ITW.ChildWindows)
     Get(ITW.ChildWindows,I)
     ChildWindows.CW:CwHwnd   = ITW.ChildWindows.CwHwnd
     ChildWindows.CW:Style    = ITW.ChildWindows.Style
     ChildWindows.CW:ExStyle  = ITW.ChildWindows.ExStyle
     ChildWindows.CW:Text     = ITW.ChildWindows.Text
     Add(ChildWindows)
   End
 End
```

Please refer to the Example Program[164] TestEnumChildWindows[163] procedure for examples of implementation.

See also:
EnumTopWin[119]

3.33.4.5 EnumModuleWin

Prototype: **(String pModuleName)!!,Long**

pModuleName Name of the EXE file to enumerate top windows for. This must be in the form of FileName+Extension, such as NOTEPAD.EXE. Note that this parameter is converted to LongPath() for compatibility with the ModuleEXEName[108] member of the ChildWindowQ[108] queue type.

Returns Returns the number of windows enumerated for the module.

Beta 3.3: This method has not been tested thoroughly yet and should be used carefully!

This method can be used to enumerate all windows that are related to a specified executable module, specified in the pModuleName parameter. This method uses the EnumTopWin method to enumerate all top windows for the module.

Example:

```
 If ITW.EnumModuleWin('EXPLORER.EXE')
 End
```

See also:
EnumTopWin[119]
ModuleWindows[111]
ChildWindowQ[108]

3.33.4.6 EnumTopWin

Prototype: **(),Long**

Returns Number of top windows

This method is used to enumerate all top windows running on the machine. This can come in handy when searching for a window where only part of the caption is known, i.e. "EXCEL" or "WORD". The TopWindows queue contains both the caption in the Text variable and also the upper cased text in the UpperText variable.

Example:

```
SetupWindow          ROUTINE
 Data
I  Long
 Code
 If ITW.EnumTopWin()
   Loop I = 1 To Records(ITW.TopWindows)
     Get(ITW.TopWindows,I)
     TopWindows.CW:CwHwnd   = ITW.TopWindows.CwHwnd
     TopWindows.CW:Style    = ITW.TopWindows.Style
     TopWindows.CW:ExStyle  = ITW.TopWindows.ExStyle
     TopWindows.CW:Text     = ITW.TopWindows.Text
     Add(TopWindows)
   End
 End
```

Here is an example that will only load windows based on a search criteria.

```
FindTopWindows          ROUTINE
 Data
I  Long
FS CString(Size(Loc:Search)+1)
 Code
 FS = Upper(Clip(Loc:Search))
 Free(TopWindows)
 If ITW.EnumTopWin()
   Loop I = 1 To Records(ITW.TopWindows)
     Get(ITW.TopWindows,I)
     If Instring(FS,ITW.TopWindows.UpperText,1,1)
       TopWindows.CW:CwHwnd   = ITW.TopWindows.CwHwnd
       TopWindows.CW:Style    = ITW.TopWindows.Style
       TopWindows.CW:ExStyle  = ITW.TopWindows.ExStyle
       TopWindows.CW:Text     = ITW.TopWindows.Text
       Add(TopWindows)
     End
   End
 End
```

Please refer to the Example Program[164] TestEnumChildWindows[163] procedure for examples of implementation.

See also:
EnumChildWin[117]

3.33.4.7 FindWindow

Prototype: **(String pCaption, Byte pFindInCaption=True, Byte pFullMatch=False),Long,VIRTUAL**

pCaption	String to find
pFindInCaption	Indicates if the Caption or the EXE name should be searched. This parameter defaults to True
pFullMatch	If true the stirng is compared directly. If False the string is compared with Instring. This parameter defaults to False. **The search is ALWAYS case insensitive!**
Returns	Window handle if a window was found. If no window was found the return value is 0 (zero)

This method searches the caption text of all top windows using Instring. Note that the TopWindows property is loaded with the enumerated top windows after a call to this method, so you can perform more detailed search if the FindWindow returns zero. Please note that this will return the first window found and not indicate if there are more windows that fit the search criteria. For that type of code, please see the examples for EnumTopWin 119 and the code in the Example Program 164 for EnumTopWin 119.

NEW Beta 3.3

This method can now be used to search for a top window where the EXE name matches the pCaption parameter. The other new parameter can be used to specify exact search string, i.e. it will not find "NOTEPAD" in "NOTEPAD.EXE" if this parameter is set to TRUE. If the parameter is false it would find "NOTEPAD" in "NOTEPAD.EXE" Note that when you pass a executable name in pCaption, it is stripped to the filename and extension only and then compared to the ModuleEXEName which also contains just the filename and extension. If the pCaption contains a '\' then it is treated as a possible path and only the filename and extension are extracted from it.

PLEASE NOTE: This method simply returns the first window handle that matches. There may be multiple top windows for a given process. Our next release will have a method that enumerates all top windows for a given executable along with more information about each window that is enumerated.

Example:

```
TestFindWindow              ROUTINE
 Data
hWnd Long
 Code
 hWnd = ITW.FindWindow(Loc:Search,Loc:SearchInCaption)
 If hWnd
   Message('Window found, handle returned = ' & hWnd,'Window
found',ICON:Exclamation)
 Else
   Message('Window not found','Window not found',ICON:Hand)
 End
```

If Loc:Search contains "Notepad.exe" and Loc:SearchInCaption is true, then this will search for a top window belonging to Notepad.exe.

See also:
EnumTopWin 119
EnumChildWin 117

3.33.4.8 GetBaseControlName

Prototype: **(String pLabel, Byte pUpper),String**

pLabel The Field EQuate label name.

pUpper Indicates if the Base name should be uppercased before it is returned

Returns String containing the field name

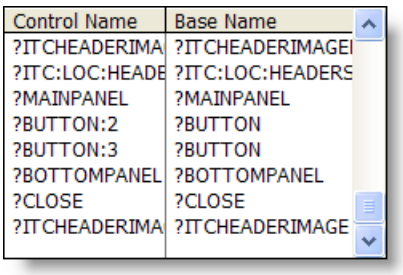
This method is called by the GetBaseControName function with the label of the control to get the base control name for. The base control name is for example ?INSERT for an control called ?INSERT:3 i.e. it strips any numbers off of the end so any control with that base name can be address for example when looping through all controls on a window or a report.

Example:

```
LoadControlQ             ROUTINE
 Data
I  Long
 Code

 Loop I = FirstField() To LastField()
   Loc:CQ.Loc:CQName     = ITW.GetControlName(I)
   Loc:CQ.Loc:CQBaseName = ITW.GetBaseControlName(I)
   Add(Loc:CQ)
 End
```

Below is a screenshot of the Loc:CQ in a listbox. Please note the ?BUTTON:2 and ?BUTTON:3 in the Control Name column and also in the Base Name column. Also note that ?CLOSE has not changed.



See also:
GetControlName 122

3.33.4.9 GetCommandLineLen

Prototype: **(),Long**

Returns Returns the length of a command line that a program can accept

This method attempts to calculate the total length of a command line acceptable based on the operating system. This is not failsafe and should be used only as a guideline. Additional information about the maximum size of the command line can be found on Microsoft's Support website at http://support.microsoft.com/kb/830473 and on MSDN website at http://msdn2.microsoft.com/en-us/library/ms724834.aspx

This method can be used to determine if a constructed command line is too long for the operating system to pass it to another program. If it is too long and you are passing filenames to another program, consider using ShortPath() on the filename before placing it in the command line.

Example:

```
L  Long
ITW ITWindowsClass
 Code
 L = ITW.GetCommandLineLen()
 Message('Maximum length of the command line is ' & L & ' characters.')
```

3.33.4.10 GetControlName

Prototype: **(Long pFEQ),String**

pFEQ The Field EQuate label of a control to get the name for.

Returns Returns a string containing the label of the control.

This method uses an undocumented function in the Clarion Runtime Library that returns the label of a control, for example '?String1' or '?MYF:Field1' as a string rather than a numeric value.

Example:

```
LoadControlQ            ROUTINE
 Data
I  Long
 Code

 Loop I = FirstField() To LastField()
   Loc:CQ.Loc:CQName     = ITW.GetControlName(I)
   Loc:CQ.Loc:CQBaseName = ITW.GetBaseControlName(I)
   Add(Loc:CQ)
 End
```

See also:
GetBaseControlName 121

3.33.4.11 GetDialogUnit

Prototype: **(Byte pVertical),Long**

pVertical Indicates if the Dialog unit value should be for Vertical or Horizontal.

Returns Returns the dialog base units for height or width of a standard system font character.

The Dialog Units are used in Clarion (and other languages) to get uniform sizes of dialogs independent on what fonts are used for the window and what resolution is used. This method returns the calculated values based on the GetDialogBaseUnits api. This method is used in the UsingLargeFonts 140 method to determine the size of the dialog units.

Example:

```
Loc:DialogUnitsString = 'Dialog units: ' &|
                        ITW.GetDialogUnit(False) & 'x' &|
                        ITW.GetDialogUnit(True)
```

See also:
UsingLargeFonts 140

3.33.4.12 GetExeFromWindowHandle — Windows Class - Methods

Prototype: **(Long pHwnd),String**

pHwnd Handle to a window.

Returns Returns EXE filename that owns the window passed to the method

This method returns the name of the process that owns the window handle that is passed to the method. This can be very useful if you know a handle but need to find the executable that it belongs to.

Example:

```
Mn  CString(2049)
ITW ITWindowsClass
 Code
 Mn = ITW.GetExeFromWindowHandle(0{Prop:Handle})
 Message('Owner of this window is: ' &
Mn,'GetExeFromWindowHandle',ICON:Exclamation)
```

See also:
EnumTopWin 119
EnumModuleWin 118
EnumChildWin 117
EnumChildWindowsProc 142
EnumTopWindowsProc 141
GetPIDFromWindowHandle 123

3.33.4.13 GetPIDFromWindowHandle — Windows Class - Methods

Prototype: **(Long pHwnd),IT_DWORD**

pHwnd Handle to a window.

Returns Returns the Process ID (PID) for the process that owns the window.

This method returns the process ID of a window. It can be useful to identify what process a window belongs to in order to get process information.

Example:

```
ITW ITWindowsClass
```

```
PID  IT_DWORD
 Code
 PID = ITW.GetPIDFromWindowHandle(0{Prop:Handle})
 Message('Process ID of this window is: ' &
PID,'GetPIDFromWindowHandle',ICON:Exclamation)
```

See also:
EnumTopWin 119
EnumModuleWin 118
EnumChildWin 117
EnumChildWindowsProc 142
EnumTopWindowsProc 141
GetExeFromWindowHandle 123

3.33.4.14 GetPixelHeight

Prototype: **(Long pFEQ),Long**

pFEQ The Field EQuate label of the control to get the Height in pixels for.

Returns Returns the height of the control in pixels

This method returns the height of the passed control in pixels.

Example:

```
(none)
```

See also:
GetPixelWidth 125
GetPixelXPos 126
GetPixelYPos 126

3.33.4.15 GetPixelPos

Prototype: **(Long pFEQ, Long pC),Long**

pFEQ The Field EQuate label for the control to get position in pixels for.

pC Property to use, this could be PROP:XPos, PROP:YPos, PROP:Width or PROP:Height

Returns Returns the appropriate property value in Pixels.

This method is used by GetPixelHeight 124, GetPixelWidth 125, GetPixelXPos 126 and GetPixelYPos 126 to get the pixel coordinates and size for the pFEQ control.

Example:

```
(none)
```

See also:
GetPixelHeight 124
GetPixelWidth 125
GetPixelXPos 126

GetPixelYPos 126

3.33.4.16 GetPixelPosition

Windows Class - Methods

Prototype: **(Long pFEQ,<*Long pX>,<*Long pY>,<*Long pW>,<*Long pH>)**

pFEQ	The Field EQuate label for the control to get position in pixels for.
[pX]	Optional parameter to receive the pFEQ contols X position in pixels.
[pY]	Optional parameter to receive the pFEQ contols Y position in pixels.
[pW]	Optional parameter to receive the pFEQ controls Width in pixels.
[pH]	Optional parameter to receive the pFEQ controls Height in pixels.

This method can be used to retrieve one or more of a control's coordinates and size in pixels.

Example:

```
GetButtonSizeInPixels  ROUTINE
 Data
X  Long
Y  Long
W  Long
H  Long
 Code
 ITW.GetPixelPosition(?Button,X,Y,W,H)
```

See also:
GetPixelPos 124
GetPixelHeight 124
GetPixelWidth 125
GetPixelXPos 126
GetPixelYPos 126

3.33.4.17 GetPixelWidth

Windows Class - Methods

Prototype: **(Long pFEQ),Long**

pFEQ The Field EQuate label of the control to get the Width in pixels for.

Returns Returns the width of the control in pixels

This method calls GetPixelPosition 125 and returns the Width of the pFEQ control in pixels.

Example:

```
(none)
```

See also:
GetPixelHeight 124
GetPixelXPos 126
GetPixelYPos 126

3.33.4.18 GetPixelXPos

Prototype: **(Long pFEQ),Long**

pFEQ The Field EQuate label of the control to get the X position in pixels for.

Returns Returns the height of the control in pixels

This method calls GetPixelPosition[125] and returns the X position of the pFEQ control in pixels.

Example:

```
(none)
```

See also:
GetPixelHeight[124]
GetPixelWidth[125]
GetPixelYPos[126]

3.33.4.19 GetPixelYPos

Prototype: **(Long pFEQ),Long**

pFEQ The Field EQuate label of the control to get the Y position in pixels for.

Returns Returns the height of the control in pixels

This method calls GetPixelPosition[125] and returns the Y position of the pFEQ control in pixels.

Example:

```
(none)
```

See also:
GetPixelHeight[124]
GetPixelWidth[125]
GetPixelXPos[126]

3.33.4.20 GetPopupXY

Prototype: **(Long pFEQ,<*Long pX>,<*Long pY>)**

pFEQ The Field EQuate label for the control to get the Popup X and Y information for
[pX] Optional parameter to receive the X pixel position value
[pY] Optional parameter to receive the Y pixel position value

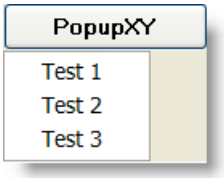
This method can be used to get the X and Y coordinates for the Clarion POPUP statement so the popup menu appears in an exact location relative to a specified control, passed in the pFEQ parameter. This is useful if you want a menu to appear below a button for example.

Example:

```
ShowPopupMenu           ROUTINE
 Data
```

```
X    Long
Y    Long
P    Short
 Code
 ITW.GetPopupXY(?PopupMenuButton,X,Y)
 Y += ITW.GetPixelHeight(?PopupMenuButton)
 P = Popup('Test 1|Test 2|Test 3',X,Y)
```

This results in a button with a popup menu below it as seen in the screenshot below:



See also:
GetPixelHeight 124

3.33.4.21 GetScreenBaseDPIRatio

Windows Class - Methods

Prototype: **(),Real**

Returns Ratio between 96 and the Screen DPI.

This method returns the ratio between a screen using 96DPI (normal font size) and the current screen DPI as returned by GetScreenDPI 127. This can be useful to scale controls **down** when designing on windows with large font settings. In order to scale **up**, use GetScreenDPIRatio 128.

Example:

```
(none)
```

See also:
GetScreenDPI 127
GetScreenDPIRatio 128

3.33.4.22 GetScreenDPI

Windows Class - Methods

Prototype: **(),Long**

Returns Returns the value from GetDeviceCaps for LOGPIXELSX on the desktop window.

This method uses the GetDeviceCaps api to determine the number of pixels per logical inch along the screen width. This can be used to determine the size of a control in inches or other units as seen on the screen. For example if the DPI is 96 then a control that is 96 pixels wide should be exactly 1 inch wide or 25.4mm.

Example:

```
(none)
```

See also:
GetScreenBaseDPIRatio 127
GetScreenDPIRatio 128

3.33.4.23 GetScreenDPIRatio

Prototype: **(),Real**

Returns Ratio between 96 and the Screen DPI.

This method returns the ratio between a screen using 96DPI (normal font size) and the current screen DPI as returned by GetScreenDPI 127. This can be useful to scale controls **up** when designing on windows with large font settings. In order to scale **down**, use GetScreenBaseDPIRatio 127.

Example:

(none)

See also:
GetScreenDPI 127
GetScreenBaseDPIRatio 127

3.33.4.24 GetScreenX

Prototype: **(Long pFEQ),Long**

pFEQ The Field EQuate label for the control to get the Screen X position for.

Returns Returns the screen X position in pixels for the control

This method returns the Screen X Position for the pFEQ control. This uses the ClientToScreen api to retrieve the screen coordinates for the control.

Example:

(none)

See also:
GetScreenY 128

3.33.4.25 GetScreenY

Prototype: **(Long pFEQ),Long**

pFEQ The Field EQuate label for the control to get the Screen Y position for.

Returns Returns the screen Y position in pixels for the control

This method returns the Screen Y Position for the pFEQ control. This uses the ClientToScreen api to retrieve the screen coordinates for the control.

Example:

(none)

See also:
GetScreenY 128

3.33.4.26 GetSysMetrics

Windows Class - Methods

Prototype: **(Long pIndex),Long**

pIndex The Index value for GetSystemMetrics.

Returns Returns the value from GetSystemMetrics.

This method calls the GetSystemMetrics api directly. Note that most or all the index values are available in the ITUtilityClass. See the ITWin32Equates.inc for more information. Search for "!! GetSystemMetrics equates" in the file or IT_SM_ to get to the appropriate section in the file. This method simply calls GetSystemMetrics and returns the value returned from the api.

Example:

```
(none)
```

See also:
GetSystemMetrics

3.33.4.27 GetTaskbarHeight

Windows Class - Methods

Prototype: **(),Long**

Returns Return the height of the Windows taskbar in pixels

This method uses the FindWindow and GetWindowRect api functions to get the height of the shell tray window and thus the taskbar. There are other ways to do this that can also take into account other dockable toolbars such as MS Office etc. and we may add those later if there is interest.

Example:

```
ITW ITWindowsClass
TBH  Long
 Code
 TBH = ITW.GetTaskBarHeight()
 Message('Toolbar Height: ' & TBH)
```

See also:
FindWindow
GetWindowRect

3.33.4.28 GetThemedPanelFEQ

Windows Class - Methods

Prototype: **(Long pPanelFEQ)**

pPanelFEQ The Field EQuate label of the original panel.

Returns Returns the Field EQuate label of the tab sheet that was created

This method can be used if you use ThemeAPanel 139 method to theme a panel using XPThemes 22. The ThemeAPanel 139 hides the panel and creates a wizard tabsheet instead creating an illusion of a themed panel! But if you want to change any settings on the "new" panel, i.e. the tabsheet, you need to use the GetThemedPanelFEQ to get the control FEQ.

Example:

```
ITW   ITWindowsClass
SFEQ  Long
PFEQ  Long
 Code
 PFEQ =  ?MainPanel
 SFEQ = ITW.GetThemedPanelFEQ(PFEQ)
 Message('Sheet FEQ = ' & SFEQ & '|Panel FEQ = ' & PFEQ &|
         '||Because the ?MainPanel is not themed in this demo app, the numbers
are always equal.','GetThemedPanelFEQ',ICON:Exclamation)
```

See also:
ThemeAPanel 139
XPThemesPresent 22

3.33.4.29 GetWindowVersion

Prototype: **(),String,PROC**

Returns Returns windows version in x.x format

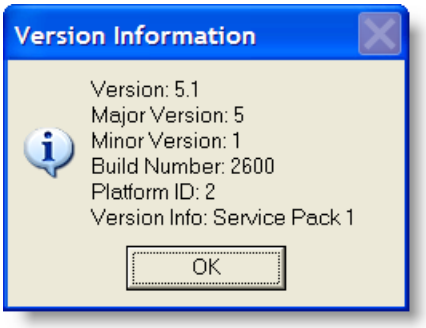
This method uses the GetVersionEx api function to retrieve the windows version information. It returns the Major and Minor version numbers in a x.x format. The method also fills in the MajorVersion, MinorVersion, VersionBuildNr, VersionPlatformID and VersionInformation properties of the ITWindowClass. After a call to this method you can use these properties to get extensive version information about the operating system that your code is running on.

For more information about the windows versions, please see http://msdn.microsoft.com/en-us/library/ms724451(VS.85).aspx and http://msdn.microsoft.com/en-us/library/ms724833(VS.85).aspx

Example:

```
ITW  ITWindowsClass
VS   CString(101)
 Code
 !...
 VS = ITW.GetWindowVersion()
 Message('Version:          ' & VS &|
        '|Major Version:    ' & ITW.MajorVersion &|
        '|Minor Version:    ' & ITW.MinorVersion &|
        '|Build Number:     ' & ITW.VersionBuildNr &|
        '|Platform ID:      ' & ITW.VersionPlatformID &|
        '|Version Info:     ' & ITW.VersionInformation, |
        'Version Information',ICON:Information)
```

On Windows XP Home, service pack 1, the results can be seen in the screenshot below.



See also:
MajorVersion 110
MinorVersion 110
VersionPlatformID 114
VersionBuildNr 113
VersionInformation 113
IsVista 109

3.33.4.30 MakeLangID

Windows Class - Methods

Prototype: **(UShort usPrimaryLanguage, UShort usSubLanguage),UShort**

usPrimaryLanguage Primary Language
usSubLanguage Sub Language

Returns Language ID for use with Locale api calls

This method performs the same task as C MakeLangID macro. The value returned can be used in various Locale api functions to retrieve specific language information. For more information on values for Primary Language and Sub Language please check this page on the MSDN website. For more information on national language support api functions check out this page.

Example:

(none)

See also:
MakeLangID

3.33.4.31 PlaceControlForDPI

Windows Class - Methods

Prototype: **(Long pFEQ, <Long pDesignDPI>)**

pPFEQ Control Field EQuate label
pDesignDPI Dots Per Inch resolution of the machine used to design the window/control

This method attempts to resize a control based on the original DPI information (if any) or the DPI ratio of the machine it is running on. For example if a control is designed in 120DPI resolution and is being run on a computer with 96DPI resolution, the control may need to be scaled down. This mostly applies to image controls that can look very bad if they are being displayed in a different resolution

than they were designed for.

Example:

```
(no example available)
```

See also:
GetScreenDPI 127
GetScreenDPIRatio 128

3.33.4.32 RedrawClientArea

Windows Class - Methods

Prototype: **(none)**

This method uses InvalidateRect to force a redraw of the window client area. Sometimes residual bits may be left on a screen, particularly when dealing with non-native Clarion controls. This method takes care of cleaning and redrawing the screen.

Example:

```
ITW.RedrawClientArea
```

See also:

3.33.4.33 RemoveWindowColor

Windows Class - Methods

Prototype: **(),BYTE**

Returns Returns LEVEL:Benign

This method removes background color drawn on a window with the SetWindowColor 138 method. This method, and SetWindowColor, behave differently on an AppFrame window than they do on a standard window. When you use it on an appframe it can only be done when the window opens. On an appframe this method is called automatically when the window closes and should be treated as a private method. On a normal window you can use this method anywhere to go back to the original color of the window.

If SetWindowColor has not been called, this method does not do anything.

Example:

```
(no example)
```

See also:
SetWindowColor 138

3.33.4.34 ResizeControlForDPI

Windows Class - Methods

Prototype: **(Long pFEQ, <Long pDesignDPI>)**

pFEQ Field EQuate label of the control to resize

[pDesignDPI] The design DPI of the target.

This method resizes the passed control with the ratio of either the DPI Ratio as calculated by GetScreenDPIRatio 128 or based on the pdesignDPI. This can be used to aid in resizing controls to work with different DPI resolutions.

Example:

```
ITW.ResizeControlForDPI(?List1)
```

See also:
GetScreenDPIRatio 128

3.33.4.35 SetControlFonts — Windows Class - Methods

Prototype: **(Long pFrom, Long pTo)**

pFrom Field EQuate label of a control to use font information from

pTo Field EQuate label of a control to set font information.

This method simply sets all font information for the pTo control exactly the same as the font information for the pFrom control. This includes font name, size, style, color and character set.

Example:

```
ITW.SetControlFonts(?Button1, ?Button2)
```

3.33.4.36 SetControlPositions — Windows Class - Methods

Prototype: **(Long pFrom, Long pTo)**

pFrom Field EQuate label of a control to use position information from

pTo Field EQuate label of a control to set position.

This method simply sets all position information for the pTo control exactly the same as the position information for the pFrom control. This includes the X and Y coordinates as well as Height and Width. Basically it places one control in exactly the same as the other control and makes them the same size.

Example:

```
ITW.SetControlPostions(?Button1,?Button2)
```

3.33.4.37 SetControlProp — Windows Class - Methods

Prototype: **(Long pFEQ, Long pProperty, String pValue)**

pFEQ Field EQuate label for the control to change

pProperty Property to set

pValue Value to set the Property to.

This method does simple property assignment, but only if the property value has changed. This can reduce flicker caused by setting properties on controls repeatedly inside of a loop. This method simply sets the property only if it has changed.

Example:

```
 If EVENT() = EVENT:Timer
   If TimerActive
     Loop 100 Times
       Next(MyFile)
       If ErrorCode()
         TimerActive = False
         Close(MyFile)
         Break
       End
       ITW.SetControlProp(?String1,PROP:Text,MYF:Name) ! Only changes PROP:Text
if MYF:Name has changed
     End
   End
 End
```

3.33.4.38 SetPixelHeight

Prototype: **(Long pFEQ, Long pValue)**

pFEQ The Field EQuate label of the control to set height for.

pValue The height of the control to set in pixel

This method uses the SetPixelPos 134 method to set the height of a control in pixels.

Example:

```
ITW  ITWindowsClass
 Code
 ITW.SetPixelHeight(?Button1,25)  !! Set the button 25 pixels tall.
```

See also:

GetPixelHeight 124

SetPixelPos 134

SetPixelWidth 136

SetPixelXPos 136

SetPixelYpos 136

3.33.4.39 SetPixelPos

Prototype: **(Long pFEQ, Long pC, Long pValue)**

pFEQ The Field EQuate label of the control to set height for.

pC	The property to set. This can be PROP:Height, PROP:Width, PROP:Xpos or PROP:Ypos
pValue	The height of the control to set in pixel

This method sets the appropriate property such as height, width, x-position or y-position in pixels. It is used by SetPixelHeight[134], SetPixelWidth[136], SetPixelXPos[136] and SetPixelYPos[136].

Example:

```
ITW  ITWindowsClass
 Code
 ITW.SetPixelPos(?Button1, PROP:Heigth, 25)  !! Set the button 25 pixels tall.
```

See also:

SetPixelHeight[134]
SetPixelWidth[136]
SetPixelXPos[136]
SetPixelYpos[136]

3.33.4.40 SetPixelPosition

Windows Class - Methods

Prototype: **(Long pFEQ,<Long pX>,<Long pY>,<Long pW>,<Long pH>)**

pFEQ	The Field EQuate label of the control to set position for.
pX	The X coordinates for the control in pixels.
pY	The Y coordinates for the control in pixels.
pW	The Width of the control in pixels.
pH	The Height of the control in pixels.

This method is basically identical to the Clarion SetPosition statement except it uses Pixel positioning rather than Dialog Unit positioning. This allows you to precisely place a control or set the size of a control with one method call.

Example:

```
ITW  ITWindowsClass
 Code
 ITW.SetPixelPosition(?Button,100,50,75,25)  !! set size of ?Button to 100 pixels
from left edge, 50 from top edge, 75 pixels wide and 25 tall.
```

See also:

SetPixelHeight[134]
SetPixelWidth[136]
SetPixelXPos[136]
SetPixelYPos[136]
SetPixelPos[134]

3.33.4.41 SetPixelWidth

Prototype: **(Long pFEQ, Long pValue)**

pFEQ The Field EQuate label of the control to set width for.
pValue The width of the control to set in pixel

This method uses the SetPixelPos 134 method to set the width of a control in pixels.

Example:

```
ITW  ITWindowsClass
 Code
 ITW.SetPixelWidth(?Button1,100)  !! Set the button 100 pixels wide.
```

See also:
GetPixelWidth 125
SetPixelPos 134
SetPixelWidth 136
SetPixelHeight 134
SetPixelXPos 136
SetPixelYpos 136

3.33.4.42 SetPixelXPos

Prototype: **(Long pFEQ, Long pValue)**

pFEQ The Field EQuate label of the control to set X-position for.
pValue The X coordinates of the control to set in pixel

This method uses the SetPixelPos 134 method to set the X-position of a control in pixels.

Example:

```
ITW  ITWindowsClass
 Code
 ITW.SetPixelXPos(?Button1,100)  !! Set the button 100 pixels in from the left
window edge.
```

See also:
GetPixelXPos 126
SetPixelPos 134
SetPixelWidth 136
SetPixelHeight 134
SetPixelYpos 136

3.33.4.43 SetPixelYPos

Prototype: **(Long pFEQ, Long pValue)**

pFEQ The Field EQuate label of the control to set Y-position for.

pValue The Y coordinates of the control to set in pixel

This method uses the SetPixelPos 134 method to set the Y-position of a control in pixels.

Example:

```
ITW  ITWindowsClass
 Code
 ITW.SetPixelYpos(?Button1,50)  !! Set the button 50 pixels down from the top
edge of the window.
```

See also:

GetPixelXPos 126
SetPixelPos 134
SetPixelWidth 136
SetPixelHeight 134
SetPixelYpos 136

3.33.4.44 SetToolboxCaption

Windows Class - Methods

Prototype: **(Long pHwnd, Byte pSetOn=True)**

pHwnd Handle of a window to set toolbox caption on.

pSetOn Indicates if the effect is to be turned on or off. True turns it on, False turns it off.

This method sets a toolbox caption for the window. A toolbox caption has a smaller caption bar but also can only have the close button and nothing else, i.e. no minimize, maximize buttons or system menu are visible. This works well for toolbox windows. This method resizes the window to match the correctly small and large caption bars as reported by GetSysMetrics 129 with the SM_CYCAPTION and SM_CYSMCAPTION parameters.

Note:
Even though this method takes the handle of the window as a parameter, it presumes that the window is also the current target, i.e. it uses 0 as window reference to get and set the size of the window. If you are using this in a situation where the window is not the current target, please keep this in mind and use **SetTarget** before and after calling this method.

Example:

```
ITW.SetToolboxCaption(0{Prop:Handle} , Choose(Loc:IsToolbox=True, False, True))
Loc:IsToolbox = Choose(Loc:IsToolbox=True, False, True)
```

This will toggle the window between being a toolbox window and being a normal window. All styles of the window are preserved so that when the toolbox is turned off, all properties of the window as it was originally are restored.

Normal window:

Same window with SetToolboxCaption turned on:

See also:
GetSysMetrics 129

3.33.4.45 SetWindowColor

Prototype: **(Long pColor)**

pColor Color to use

When this method is used on an appframe window it should be called during the startup process of the window i.e. in ThisWindow.Init, after the window is opened (priority around 8000).

When this method is used on other windows it can be called anywhere and at any time after the window is opened and called repeatedly.

Example:

```
ITW  ITWindowsClass
Col  Long
 Code
 If ColorDialog('Select Color',Col)
   ITW.SetWindowColor(Col)
 End
```

See also:
RemoveWindowColor 132

3.33.4.46 SetWindowNotOnTop

Prototype: **None**

This method can be used to make the window not be on top after it is set with SetWindowOnTop.

Example:

```
ITW.SetWindowNotOnTop
```

See also:
SetWindowOnTop 138

3.33.4.47 SetWindowOnTop

Prototype: **None**

This method can be used to set windows to be on top of other windows.

Example:

```
ITW.SetWindowOnTop
```

See also:
SetWindowNotOnTop 138

3.33.4.48 SetWindowPosition — Windows Class - Methods

Prototype: **(Long pX, Long pY, Byte pSetPixels=True)**

pX The X coordinate for the window

pY The Y coordinate for the window

pSetPixels Indicates if PROP:Pixels is turned on during the positioning

This method does the same thing as the clarion Position() function, but can alternatively be set to use pixels for more accurate placement.

Example:

```
ITW.SetWindowPosition(0,0,True)
```

See also:
SetWindowSize 139

3.33.4.49 SetWindowSize — Windows Class - Methods

Prototype: **(Long pWidth, Long pHeight, Byte pSetPixels=True)**

pWidth The width of the window

pHeight The height of the window

pSetPixels Indicates if PROP:Pixels is turned on during the sizing

This method does the same thing as the clarion Position() function, but can alternatively be set to use pixels for more accurate sizing.

Example:

```
ITW.SetWindowSize(800,600,True)
```

See also:
SetWindowPosition 139

3.33.4.50 ThemeAPanel — Windows Class - Methods

Prototype: **(Long pPanelFEQ)**

pPanelFEQ The Field EQuate label of the panel to theme

This method themes a panel using the XP Theme methods.

NOTE: This is only going to work if your application is using the XP Theme product from www.cwtemplates.com

Example:

```
ITW.ThemeAPanel(?Panel1)
```

3.33.4.51 UsesClearType

Prototype: **(),Byte**

Returns Returns True if the system is using Clear Type

This method checks a registry key and returns true if it is set to use Clear Type font smoothing. This can come in handy for Clarion 6 and older which do not correctly support Clear Type fonts in entry fields. This problem is fixed in Clarion 7. This code was inspired by information found here.

Example:

```
I Long
 Code
 If ITW.UsesClearType
   Loop I = FirstField() To LastField()
     If I{Prop:Type} = CREATE:ENTRY
       I{Prop:FontName} = 'MS Sans Serif'
       I{Prop:FontSize} = 8
     End
   End
 End
```

3.33.4.52 UsingLargeFonts

Prototype: **(),Byte**

Returns Returns True if the system is using larger than 100% fonts, false if it is not.

This method uses a very simply formula to detect if the fonts used are larger than 100% i.e. using large fonts. It checks if the horizontal dialog units are more than 16 pixels wide. If so the return value is true otherwise it is false.

Example:

```
 If ITW.UsingLargeFonts()
   ?Button{Prop:FontName} = 'Microsoft Sans Serif'
   ?Button{Prop:FontSize} = 8
 Else
   ?Button{Prop:FontName} = 'Microsoft Sans Serif'
   ?Button{Prop:FontSize} = 9
 End
```

See also:
GetDialogUnit 122

3.33.4.53 WindowInfoToODS

Prototype: **(String pProcedureName),VIRTUAL**

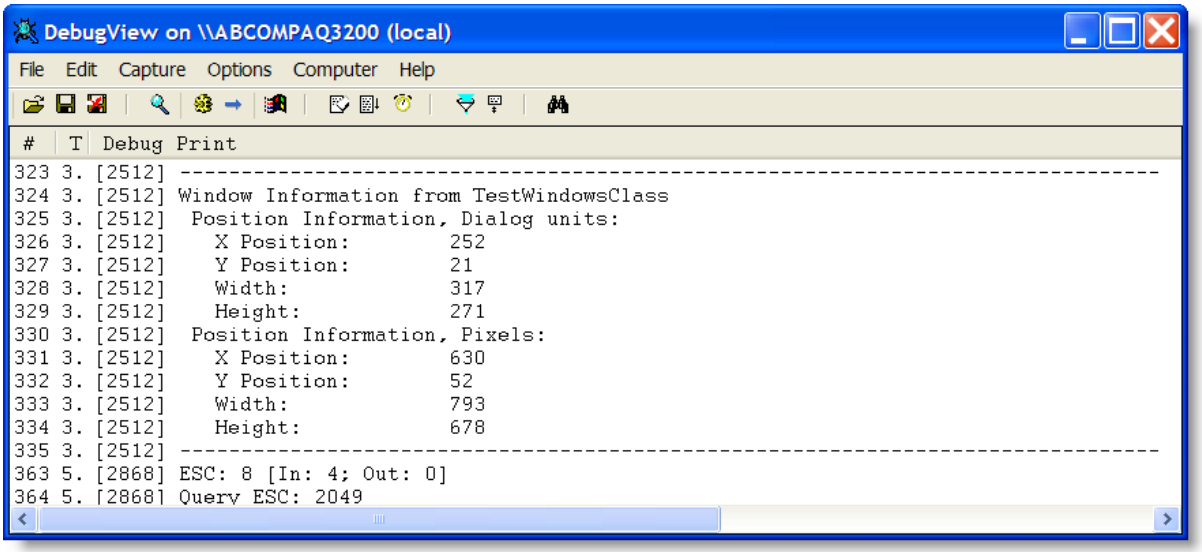
pProcedureName Name of the procedure where the window is.

This method sends position information in both dialog units and pixels to OutputDebugString using the ODS 29 method.

Example:

```
ITW.WindowInfoToODS('MyProcedure')
```

Results in this capture in DebugView:



See also:
ODS 29

3.33.5 Procedures

The Windows class contains one enumeration procedure that is a callback procedure for the EnumChildWindows API call used to enumerate the child windows in EnumChildWin 117. It also contains a copy of that procedure that is a callback procedure to enumerate top windows.

3.33.5.1 EnumTopWindowsProc

Prototype: **(Long hwnd, Long lParam),BOOL,PASCAL**

hwnd Handle to the window

lParam Reference to the Windows class

Returns True

This is a standard enumeration procedure for the EnumWindow API call defined as

EnumWindowsProc *(links active as of June 13, 2007)*

See also:

EnumChildWin 117

3.33.5.2 EnumChildWindowsProc

Prototype: **(Long hwnd, Long lParam),BOOL,PASCAL**

hwnd Handle to the window

lParam Reference to the Windows class

Returns True

This is a standard enumeration procedure for the EnumChildWindow API call defined as EnumChildProc *(links active as of June 13, 2007)*

See also:

EnumChildWin 117

Part

Chapter 4 - Templates

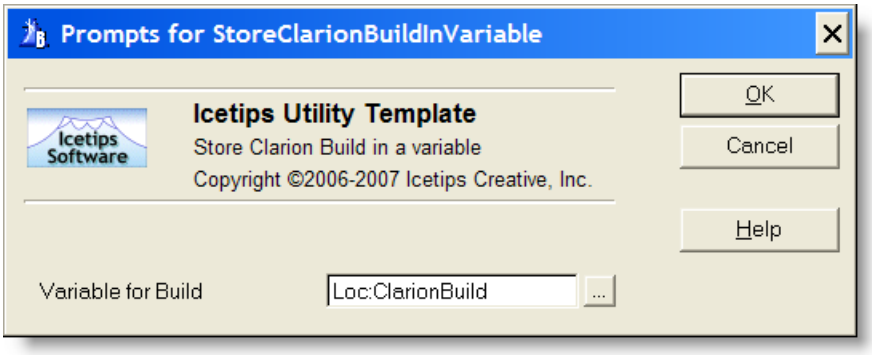
4 Templates

Enter topic text here.

4.1 Code Templates

4.1.1 Store Clarion Build in a variable Code Templates

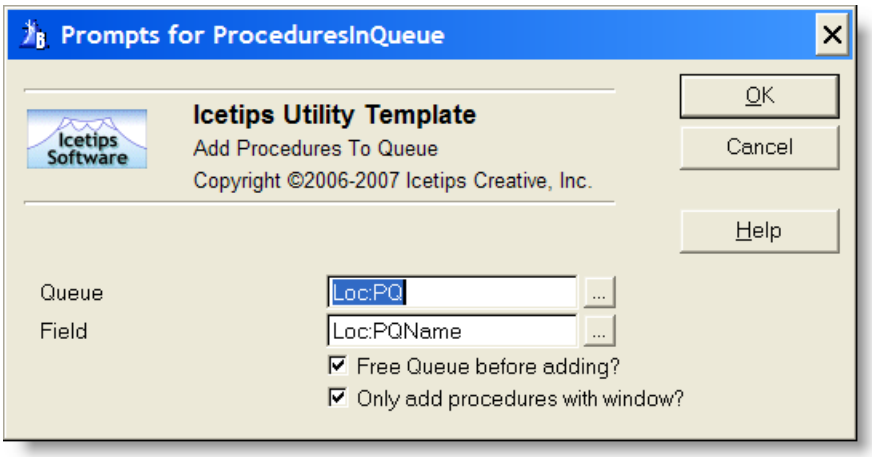
This template stores the value of the %CWVersion template symbol in a variable.



Variable for Build This variable receives the build number. The build number is a 4 digit integer that ranges from 2000 for Clarion for Windows 2.0 to 7000 for Clarion 7. Note that the build number that Softvelocity issues for each of their builds is not included in this version information. I.e. Clarion 6.3 build 9053 shows as 6300. Same does Clarion 6.3 build 9057.

4.1.2 Add Procedures To Queue Code Templates

This code template loads the names of all procedures in the application into a local queue.



This template allows you to specify which queue and field to fill with information.

Queue The label of a queue which will be loaded with the procedure names.

Field The label of a field in the Queue which will receive the procedure name.

Free Queue... When this is checked, a FREE() is executed on the queue just before it is filled. If this is not checked, the queue is not FREEd. This is checked by default.

Only add procedures... When this is checked the template will add procedures with a window ONLY. This will exclude source procedures and any other procedures that do not have a window. Note that report procedures will be included if they also have a progress window. This is checked by default.

The use of this template is demonstrated in the TestTemplate 165 procedure in the UtilDemo.app 164.

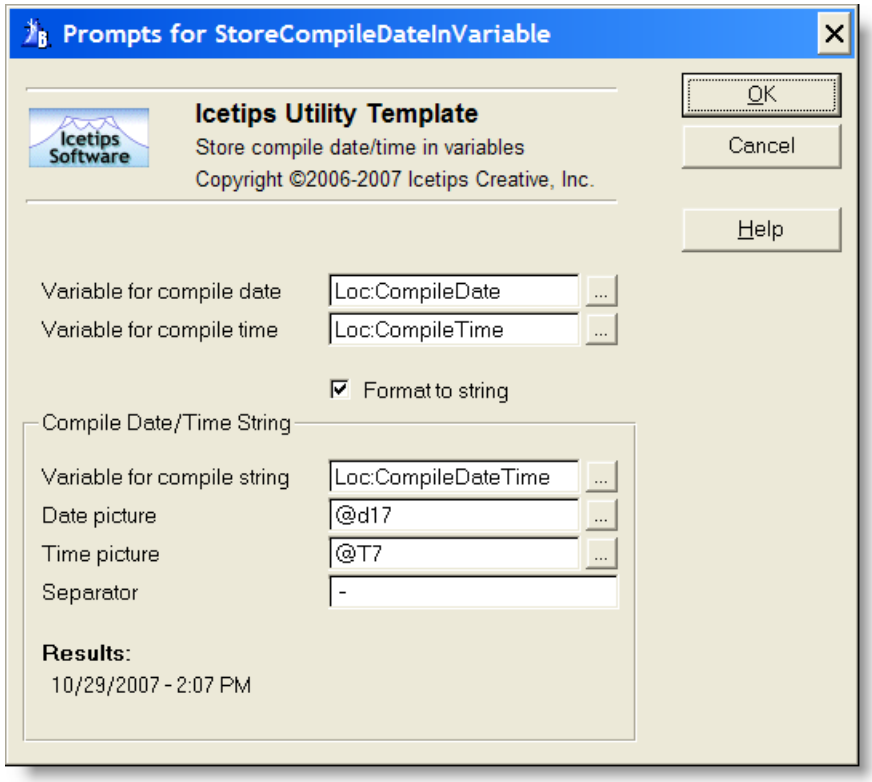
See also:
Example app: TestTemplate 165

4.1.3 Create File View Code Code Templates

Beta 3: Still under construction.

4.1.4 Store compile date/time in variables Code Templates

This template will store the compile date and/or time in variables that you specify. It can also store the concatenated date/time in a single string variable formatted the way you want it.



Variable for compile date This variable receives the compile date value. This value is created when the application code is generated and does not change at runtime. This variable should be a DATE or a LONG variable.

Variable for compile time This variable receives the compile time value. This value is created when the application code is generated and does not change at runtime. This variable should be a TIME or a LONG variable.

Format to string When this checkbox is checked, it enables the "Compile Date/Time String" group below. This allows you to place the formatted compile date and or time into a single string variable to place on a window or a report.

Variable for compile string This variable receives the formatted compile date and/or time values depending on the pictures specified for each variable. This should be a STRING or CSTRING variable. The required size varies based on format picture that you choose.

Date picture The format picture for the date. Type in a different picture if you need or use the [...] button to select a new picture.

Time picture The format picture for the time. Type in a different picture if you need or use the [...] button to select a new picture.

Separator String that is placed between the date and the time.

Results Shows the formatted date and time as it will appear on the window or report.

See also:
Example app: TestTemplate 165

4.2 Control Templates

4.2.1 Icetips MS Window header Control Templates

Beta 3: Still under construction.

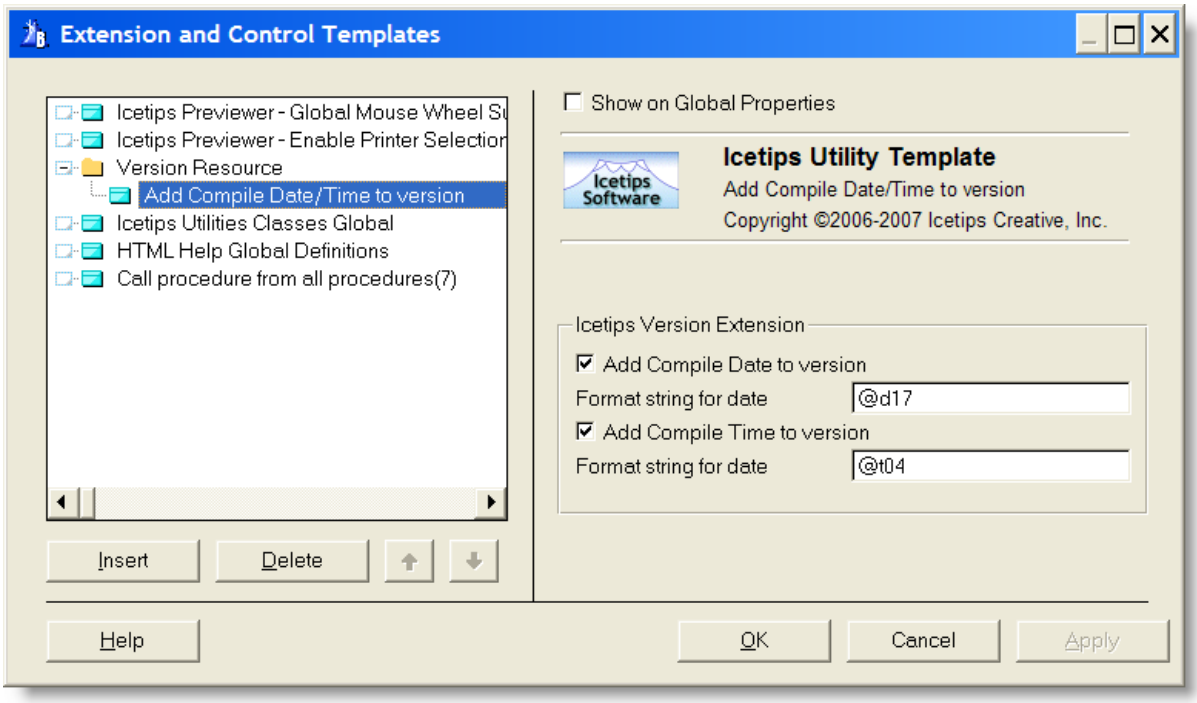
4.3 Extentsion Templates

Enter topic text here.

4.3.1 Global Extensions — Extentsion Templates

4.3.1.1 Add Compile Date/Time to version — Extentsion Templates - Global Extensions

This template makes it possible for you to add the compile date and time to the version resources that are linked into the application. This makes it very easy to see exactly when a DLL or EXE was compiled to narrow down problems with client software.



Add Compile Date Check this to add the compile date to the version information

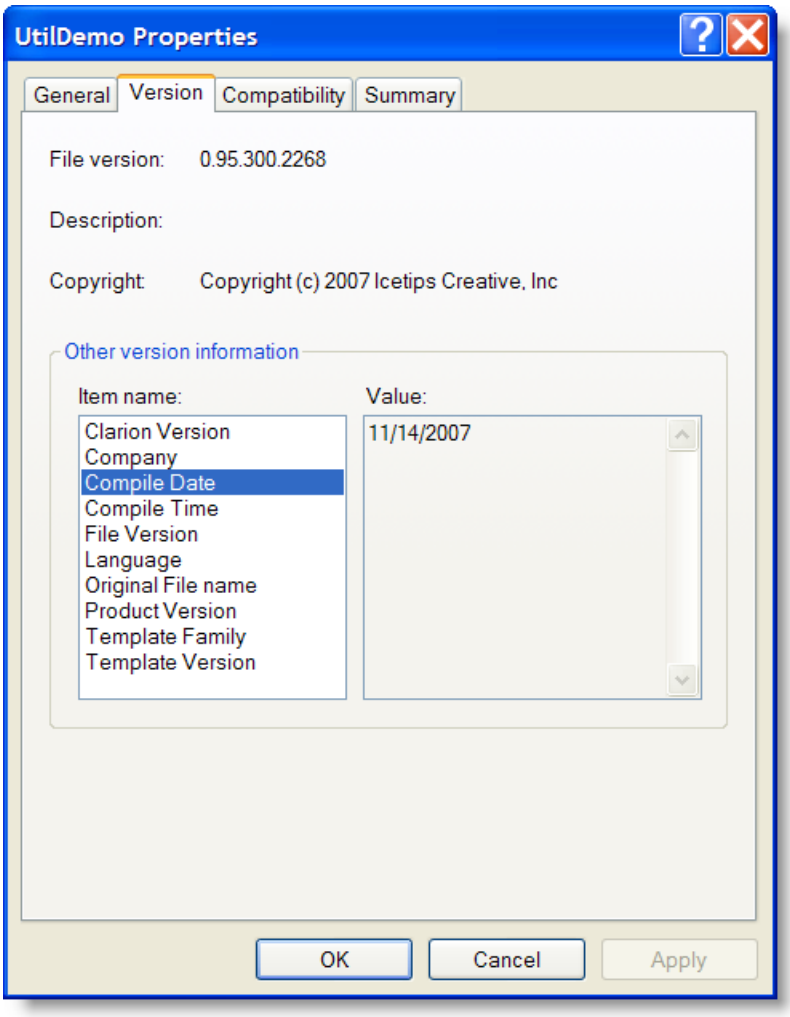
Format string for date Enter the picture for the date. This is used to format the date in the version resource. This defaults to windows short format, @d17.

Add Compile Time Check this to add the compile time to the version information

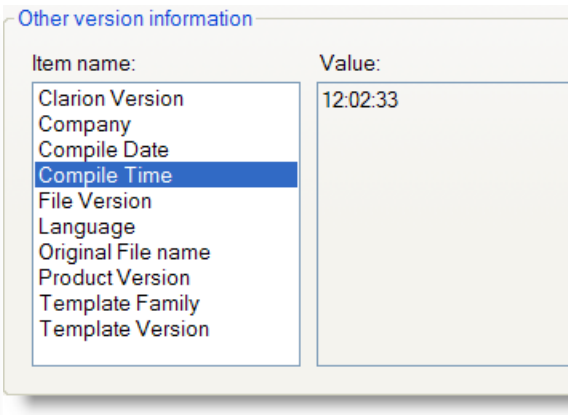
Format string for time Enter the picture for the time. This is used to format the compile time in the version resource. This defaults to hh:mm:ss format, @t4.

This results in two additional items in the Version information. Below are screenshots that show how this looks for the UtilDemo.exe version 0.95.300 compiled on November 14, 2007 at 12:02:33

Version information showing the Compile Date



Version information showing the Compile Time

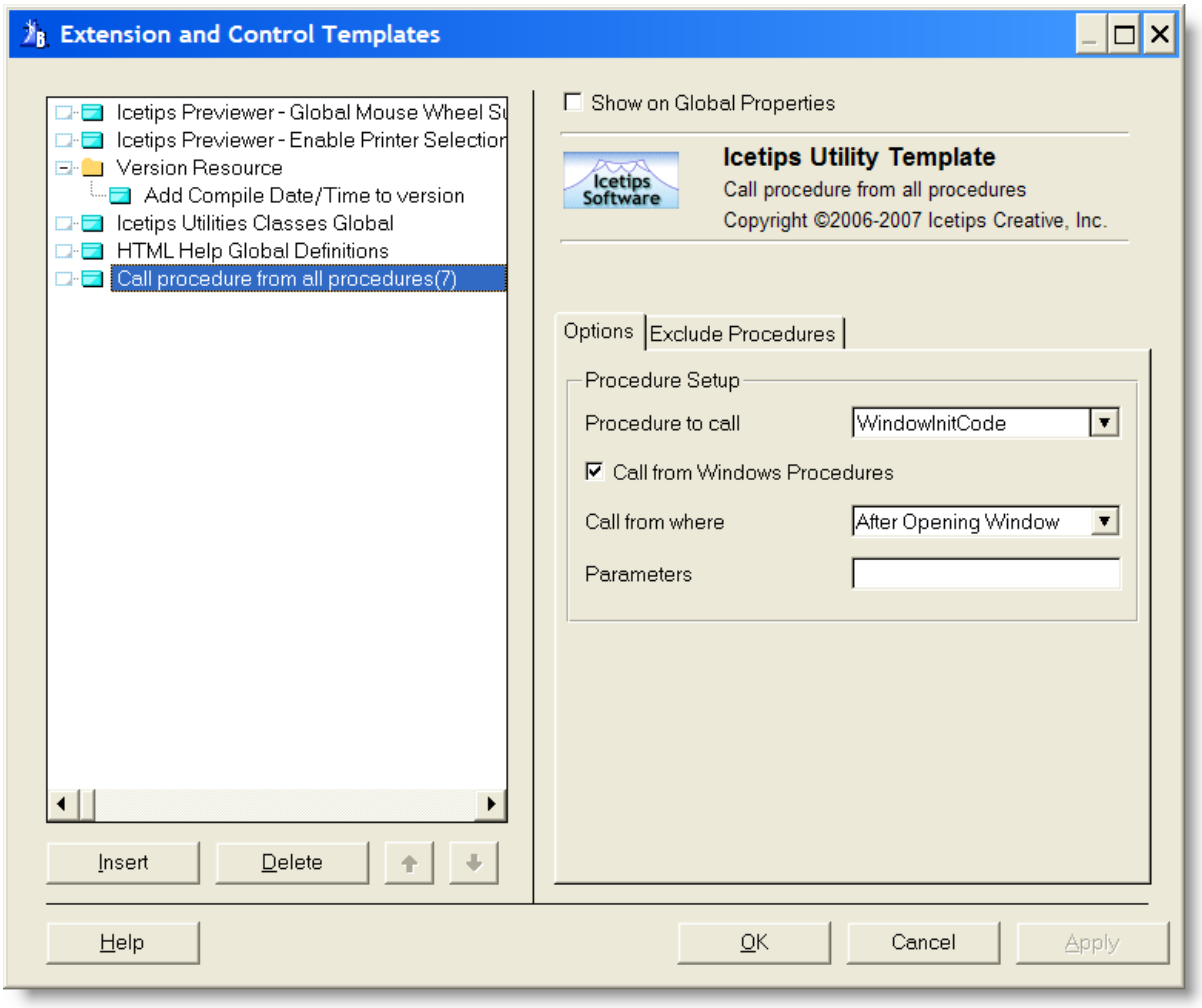


4.3.1.2 Call procedure from all procedures

Extentsion Templates - Global Extensions

This template allows you to call a single procedure from all procedures in your application. This is extremely useful for procedures that set up windows and controls. For an example of how this is beneficial, check out the WindowInitCode 164 procedure in the UtilDemo.app 164 in your "Clarion\3rdParty\Examples\ITUtilities" folder.

The template has tvo tabs, Options and Exclude Procedures. The Options tab includes the main options:



Procedure to call Select the procedure that you want to be called from all the procedures in your application except those selected on the "Exlcude Procedure" tab.

Call from Window... Check this if you only want this to be called from procedures that have a window. As of Beta 3.3 this parameter MUST be set to true or the template will not generate any code.

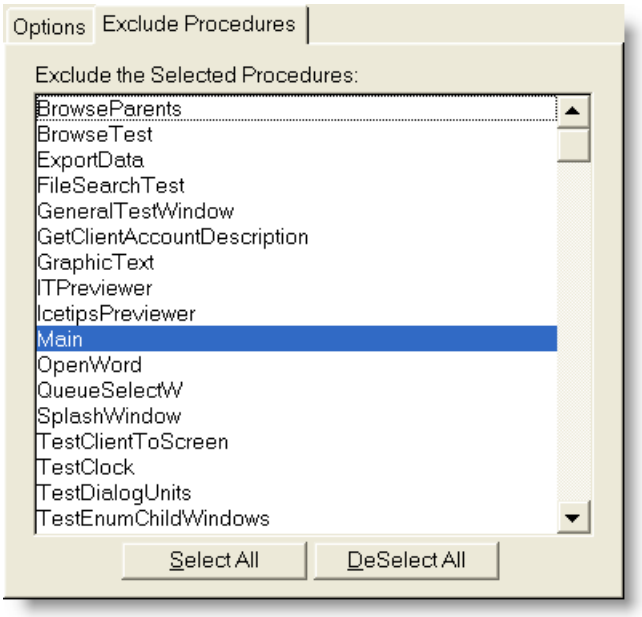
Call from where Select the embed where you want the procedure to be called from. There are 4 options:

Embed	Actual embed location
Start of Procedure	WindowManager.Init, Priority 500
Before Opening Files	WindowManager.Init, Priority 7300
Before Opening Window	WindowManager.Init, Priority 7800
After Opening Window	WindowManager.Init, Priority 8100

Parameters Optional parameters to pass to the procedure. This is currently fixed on global level so it has limited use.

The Exclude Procedures tab includes a list where you can select procedures that you want to

exclude:

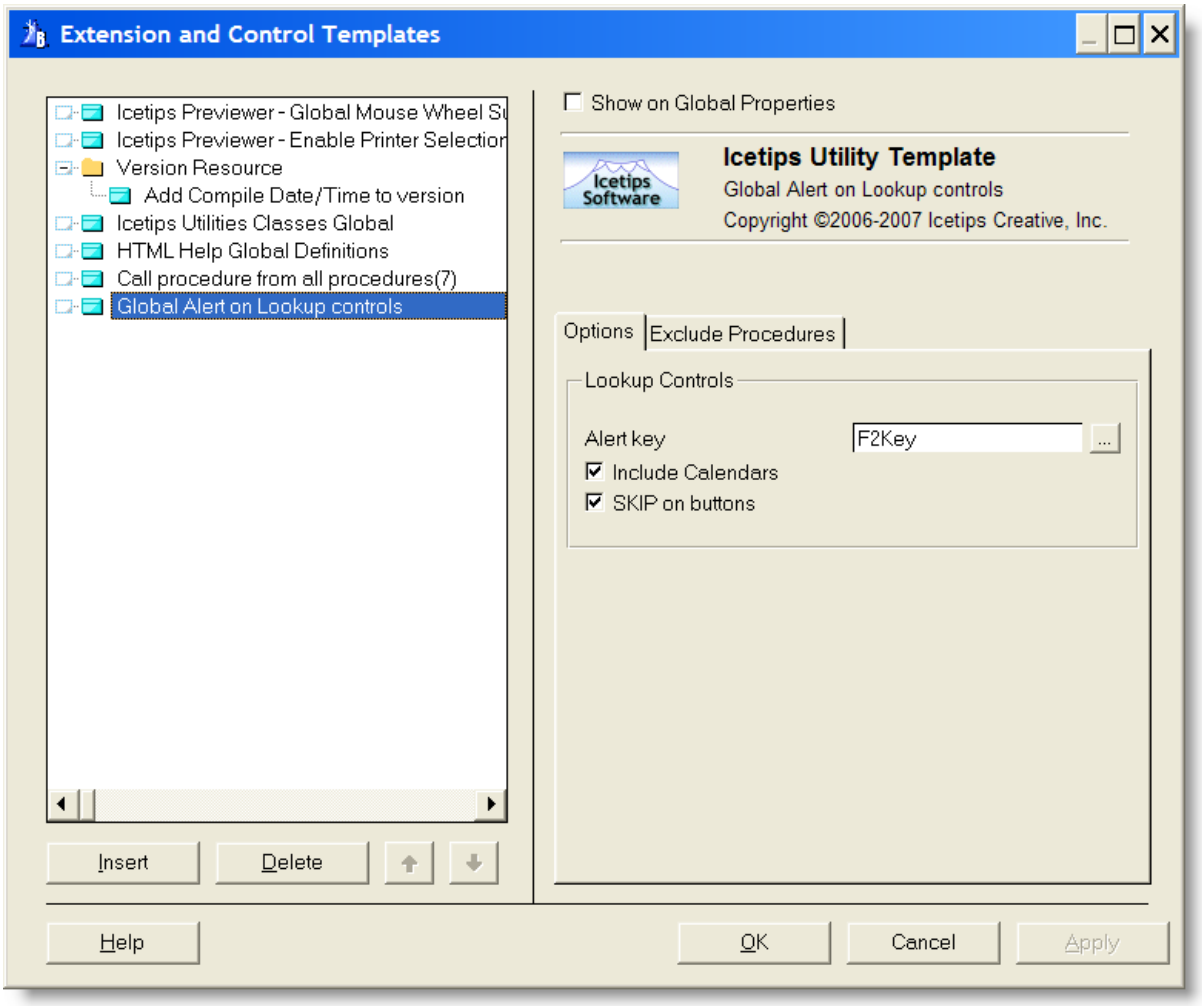


Select the procedures from the list that you do NOT want to call the selected procedure.

4.3.1.3 Global Alert on Lookup controls

Extentsion Templates - Global Extensions

This template allows you to specify a single alert key that is used on all fields that have a lookup button associated with it or a calendar button. This allows the user of your software to hit a consistent key on the keyboard to bring up lookup browses and calendars.

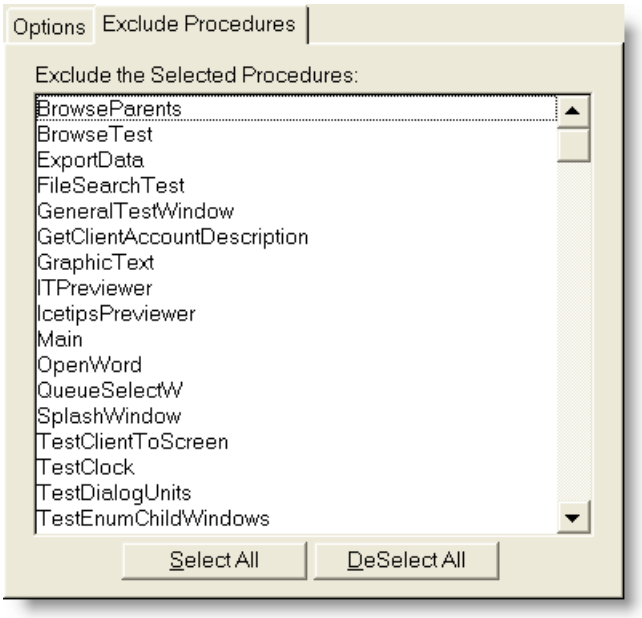


Alert Key Select the alert key that you want to use. In this screenshot we used the F2 key

Include calendars Check this if you want to include calendar lookups.

SKIP on buttons Check this if you want to add the SKIP attribute to the lookup buttons. That way users can use the key on the entry fields and skip the buttons unless they click on them with the mouse, i.e. the Tab key will skip over the buttons.

You can select procedures to exclude fromt he list on the "Exclude Procedures" tab:



4.3.1.4 Global Call ShowRecord from Browse

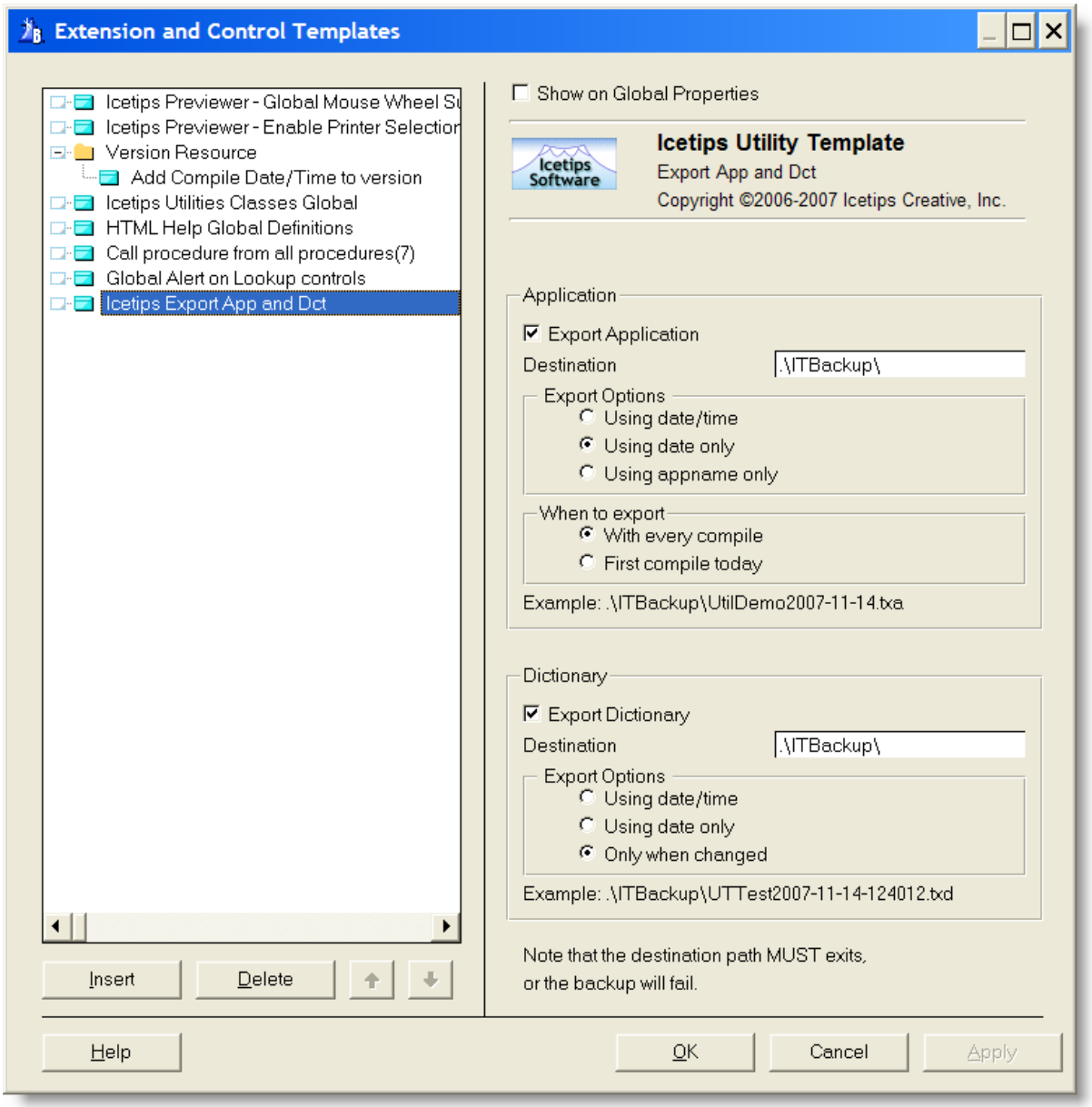
Extentsion Templates - Global Extensions

Beta 3: Still under construction.

4.3.1.5 Icetips Export App and Dct

Extentsion Templates - Global Extensions

This template exports the application and dictionary to TXA and TXD. NOTE: There have been multiple reports of problems with TXAs and TXDs generated this way with templates in the past, so DO NOT rely on those as your only backups. This problem is in the Clarion export system and there is nothing we can do to fix that. These TXAs and TXDs however can be very useful in order to restore, or look at, previous versions of embedded code and table/column information, which is all preserved.



Export Application Check this to export the application to TXA.

Destination The destination folder for the TXA. Note that this folder **MUST EXIST** or the export will fail.

Export Options Determines how to export the app.

Using date/time When using this option a new .TXA is created every time you generate or compile the application.

Using date only This option will only generate one TXA for the day. It is then overwritten when you generate again.

Using appname only When this option is used, there is only one TXA generated and overwritten with each generation.

When to export **Beta 3.3: NOT IMPLEMENTED YET!**
This is used to determine when you want to export the app.

With every compile	This will generate the TXA every time you generate or compile the application
First compile today	This will generate the TXA only once, the first time you compile it on the current date.

Export Dictionary Check this to export the dictionary to TXD.

Destination The destination folder for the TXD. Note that this folder **MUST EXIST** or the export will fail.

Export Options Determines how to export the dictionary.

Using date/time	When using this option a new .TXD is created every time you generate or compile the application.
Using date only	This option will only generate one TXD for the day. It is then overwritten when you generate again.
Only when changed	When this option is used, the TXD is only generated when the dictionary changes. NOTE: If you use this the dictionary will NOT be exported until you change it. We suggest that you use the "**Using date only**" option, then generate your application to create the first TXD. Then change this back to "**Only when changed**" and then the TXD will be created every time the dictionary changes from now on.

4.3.1.6 Icetips Global Threaded Window Manager

Extentsion Templates - Global Extensions

Beta 3: Still under construction.

4.3.1.7 Icetips Hide Windows while loading

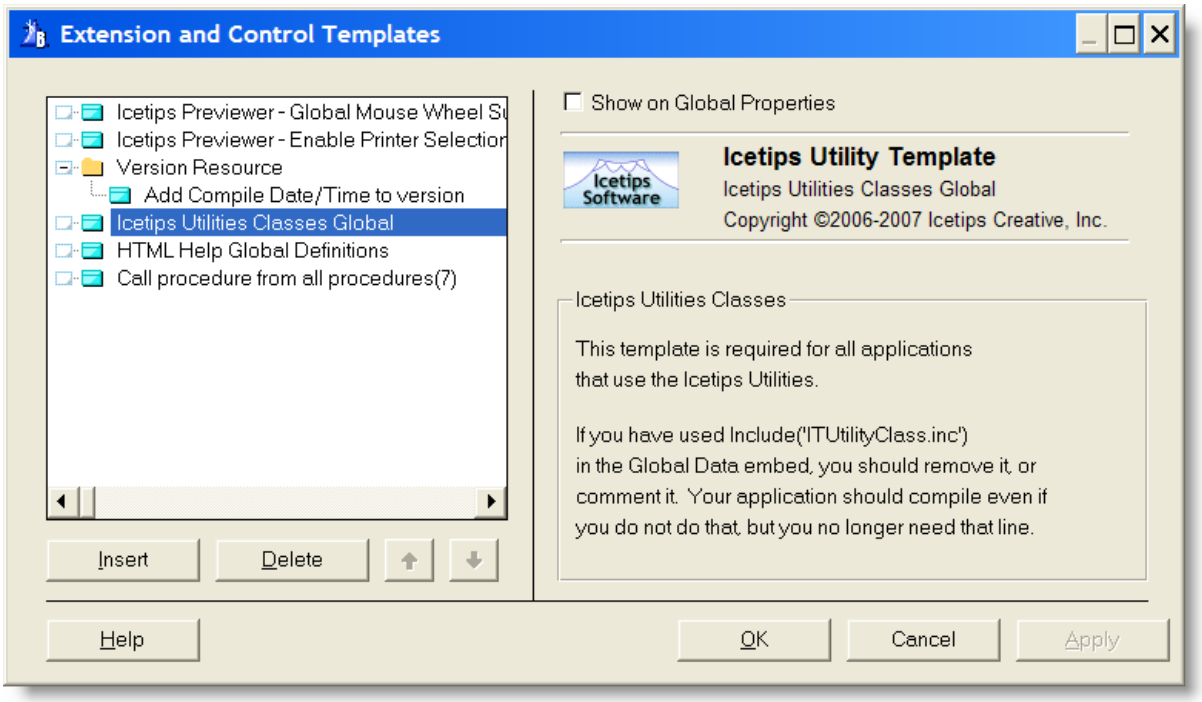
Extentsion Templates - Global Extensions

Beta 3: Still under construction.

4.3.1.8 Icetips Utility Classes Global

Extentsion Templates - Global Extensions

This template is used to add the Icetips Utilities Classes to your application. You MUST add this to all applications that need to use the Icetips Utilities Classes.



This template has no prompts.

4.3.1.9 Write Version info to INI File

Extentsion Templates - Global Extensions

Beta 3: Still under construction.

4.3.1.10 Write Template info to file

Extentsion Templates - Global Extensions

Beta 3: Still under construction.

4.3.2 Procedure Extensions

Extentsion Templates

4.3.2.1 Add Header Sort to Queue

Extentsion Templates - Procedure Extensions

Beta 3: Still under construction.

4.3.2.2 Bind/Unbind local variables

Beta 3: Still under construction.

4.3.2.3 Icetips Browse Checkbox update

Beta 3: Still under construction.

4.3.2.4 Icetips Call Threaded Window Manager

Beta 3: Still under construction.

4.3.2.5 Icetips Create File View

Beta 3: Still under construction.

4.3.2.6 Icetips Fill Queue from View

Beta 3: Still under construction.

4.3.2.7 Icetips Pre and post prime ABC Browse

Beta 3: Still under construction.

4.3.2.8 Icetips Resize Options

Beta 3: Still under construction.

4.3.2.9 Icetips Resize Options With Information

Beta 3: Still under construction.

4.3.2.10 Icetips SQL Queue Process Construction

Beta 3: Still under construction.

4.3.2.11 Icetips SQL Queue Report Construction

Beta 3: Still under construction.

4.4 Utility Templates

4.4.1 Write Modules and procedure information to Filex Utility Templates

Beta 3: Still under construction.

Part V

Chapter 5 - Example Applications

5 Example Applications

Enter topic text here.

5.1 CoreClassDemo.app

The CoreClassDemo.app is dedicated to the CoreClass 18 only. By default the UtilDemo.app Example app is installed in your Clarion\3rdParty\Examples\ITUtilities folder.

Procedure	Demonstrates
CoreClassDemo	Shows how to create GUIDs with CreateGUID 22, retrieve and change file attributes with GetFileAttrib 24 and SetFileAttrib 32. It also shows how to get temp filenames with GetTempFile 26 and the default folder for temp files with GetTempFolder 27.
SplashWindow	Shows a splash window.

5.2 WindowsClassDemo.app

The WindowsClassDemo.app is dedicated to the WindowsClass 106 only. By default the WindowsClassDemo.app Example app is installed in your Clarion\3rdParty\Examples\ITUtilities folder. The application has 5 procedures:

Procedure	Demonstrates
Main	Shows how to color the background of an appframe with SetWindowColor 138
TestWindowsClass	Shows how to use a number of the methods in the WindowsClass, such as enumerate child and top windows, search window titles, pinpoint popup menus, change the window to use toolbox caption and set the window color. This procedure is not done and we will be modifying and adding to it as we complete the demo.
TestEnumChildWindows	Enumerates all windows that are child windows of the parent handle that is passed to the procedure. This includes controls that are on the window being enumerated.
TestEnumTopWindows	This enumerates all top windows (parent windows) that are running.
SplashWindow	Shows a splash window.

5.2.1 Procedures — WindowsClassDemo.app

Enter topic text here.

5.2.1.1 TestEnumTopWindows — WindowsClassDemo.app - Procedures

Enter topic text here.

5.2.1.2 TestEnumChildWindows — WindowsClassDemo.app - Procedures

Enter topic text here.

5.3 UtilDemo.app

The Example application contains procedures that demonstrate the use of the various classes and templates in the Icetips Utilities. By default the UtilDemo.app Example app is installed in your Clarion\3rdParty\Examples\ITUtilities folder.

5.3.1 Procedures UtilDemo.app

Enter topic text here.

5.3.1.1 WindowInitCode UtilDemo.app - Procedures

This procedure is an example of how you can standardize various parts of your application from a single procedure. This procedure is called from all window procedure by using the "Call procedure from all procedures" Global extension template. This procedure uses a local queue to store all control FEQs in it. This is done to prevent any possible problems with certain properties altering the processing order of controls, which can happen when certain properties are changed. This is done with the following example code:

```
Loop
  I = 0{PROP:NextField,I}
  If Not I
    Break
  End
  If Not pIsFrame
    If I < 0
      Cycle
    End
  End
  FEQs.FEQ = I
  Add(FEQs)
End
```

Then this queue is processed and each controls is checked for it's type:

```
Loop X = 1 To Records(FEQs)
  Get(FEQs,X)
  I = FEQs.FEQ
  Loc:FEQ = I
  Case I{PROP:Type}
  ...
End
```

Each control type get's specific settings, such as this section for various listbox types:

```
  Of CREATE:List
    I{PROP:FontName} = 'Tahoma'
    I{PROP:FontSize} = 9
    F = 0
    Loop
      F += 1
      If Not I{PROPLIST:Exists,F}
        Break
      End
      I{PROPLIST:HeaderCenter,F} = True
      ! Right adjust decimal adjusted fields and set offset to 2
      If I{PROPLIST:Decimal,F} = True
        I{PROPLIST:Right,F} = True
        I{PROPLIST:RightOffset,F} = 2
      End
```

```
  End
OrOf CREATE:DropList
OrOf CREATE:DropCombo
OrOf CREATE:Combo
  I{PROP:Flat}    = True
  I{PROP:Vscroll} = True
  If I{PROP:Drop} > 0
    I{PROP:VCR}        = False
  Else
    I{PROP:VCR}        = True
  End
  I {PROP:FontName} = Loc:ListFont
  I {PROP:FontSize} = Loc:ListFontSize
```

And this part that deals with entry and spin fields:

```
Of CREATE:Entry
OrOf CREATE:Spin
  If Themed = False
    I {PROP:Flat} = TRUE
  End
  If ClearType  And Loc:ClarionBuild < 7000
    I{PROP:FontName} = 'MS Sans Serif'
  End
  I{PROP:FontSize} = 8
  If I{PROP:Decimal} = True Or Instring('D',Upper(I{PROP:Text}),1,1) = 1
    I{PROP:Right}      = True
    I{PROP:RightOffset} = 1
  End
```

In this case it changes the font name if the user is using ClearType and if this is compiled with Clarion prior to version 7.0 Clarion 6.3 has problems with TrueType fonts when used with ClearType, so we change the font to MS Sans Serif which is a bitmap font and works fine. May not look the best, but it beats chopped up character which you will get if you use TrueType fonts with ClearType!

See also:
Call procedure from all procedures 150

5.3.1.2 TestTemplate — UtilDemo.app - Procedures

This procedure is used to demonstrate some of the templates that are included in the Icetips Utilities.

Add Procedures To Queue 145
Sort Queue using Header Sort 165
Store compile date/time in variables 146

5.3.1.2.1 TestTemplateQSort

This is a simple window that demonstrates the Add Header Sort to Queue 157 procedure extension template.

5.3.1.3 TestUtilityClass — UtilDemo.app - Procedures

Enter topic text here.

Part VI

Chapter 6 - File Attributes

6 File Attributes

The File Attributes are used by the Directory function.

```
ff_:NORMAL    EQUATE(0)       ! Normal files
ff_:READONLY  EQUATE(1)       ! Not for use as attributes parameter
ff_:HIDDEN    EQUATE(2)       ! Hidden files
ff_:SYSTEM    EQUATE(4)       ! System files
ff_:DIRECTORY EQUATE(10H)     ! Directories
ff_:ARCHIVE   EQUATE(20H)     ! NOT Win95 compatible
```

Part VII

Chapter 7 - API Reference

7 API Reference

This topic contains links to various sections of the MSDN website. We use it for our own reference and thought it might benefit others who are doing research into Win32 application apis on the MSDN website. Note that these links are created in June 2007.

Windows API Reference, Functions by category
Windows API Reference, Functions in alphabetical order

Index

- D -

- E -

- F -

- G -

- H -

- I -

- L -

- M -

- N -

- O -

- P -

- Q -

- R -

- S -

- T -

- U -

- V -

- W -

- X -

- Y -

- Z -