

PRODOMUS CLASS GENERATOR

© Copyright 2004-11 ProDomus, Inc.

Table of Contents

PRODOMUS CLASS GENERATOR	1
Introduction.....	2
STEPS.....	4
TEMPLATE FILES	5
Global Prompts	9
Global Embeds.....	10
Instance Prompts	11
Instance Embeds.....	12
Control Template	13
CLASS FILES.....	14
Translation Files.....	17
Global Includes	19
Local Includes.....	20
Map Embed.....	21
Implements.....	22
Properties	23
Data	25
Properties	27
Methods	29
Method Template Embeds	30
Examples.....	32
PD SQL Query Template and Class	32
PD Window Manager Class Examples.....	32
ClarionLive Example.....	33
PDClGen.app.....	33
History	33

Introduction

Use ProDomus's Class Generator to generate template and class code that is ABC and Clarion Compliant! This takes the complexity out of creating wrapper templates and class definition files and assists developers in creating classes for both ABC and Clarion (Legacy) applications.

Wrapper Templates

- Creates class wrapper templates that are ABC and Clarion Compliant
- Creates global extensions, procedure extensions, control templates, and code templates.
- Creates global and local objects.
- Handles standalone and multi-dll applications.
- In the case of multi-dll applications, it creates classes that do not get compiled unless they are intentionally added to the application.

Class Files

- Creates translation files.
- Creates include file list within class include and clw files.
- Adds map embeds.
- Defines properties and methods and class attributes.
- Adds !,EXTENDS and !,FINAL attributes where needed.
- Adds optional class debug code with a check box.

Definition of ABC / Clarion Compliant

What is the standard Definition of ABC Compliant

- Must conform to ABC specifications in ABC Library Reference
- The Header containing the class declarations must be an .INC (“Include”) file.
- In C6 the Header Include file must be in the Libsrc directory *or in one specified in the LIBSRC entry in C60EE.Ini.*
- In C7 the Header Include file must be in the Libsrc\win or Accessory\Libsrc\Win directory.
- The Header Include file must begin with !ABCIncludeFile with an optional (*FamilyClass*) parameter.

What are some additional requirements, especially for SV and 3rd Party Developers

- Classes that are used infrequently or are 3rd Party **MUST have the FamilyClass** parameter to avoid being compiled in multi-DLL applications where not used.
- Where possible templates should work for both ABC and Clarion (Legacy) applications.
- Any class with literal strings, pictures, or windows should have a **.TRN File (Translation File)** containing the string equates and window definitions that developers can modify.
- Any class with literal strings, pictures, or windows **must provide for run time translation.**
 - %EnableRunTimeTranslator
 - Provide a TranslateWindow and TranslateString function.

The template may be used to create fully functional classes

See Also:

Steps

Class Wrapper Template Generator

Shell Class File Generator

STEPS

1. **Register PD Class Generation Template.** If not already done, register pdClaGen.tpl. Select Setup/Template Registry and then the Register button.
2. **Create Application.** Create a new application for the class.
3. **Global Data.** Add global data to the application. This will later be available for selection when entering global data to the class definition include file.
4. **Add a Procedure.** If you want to create a control template, create a window procedure and populate the controls that you would like to include in the template. If you are creating any other type of template, you can use a source procedure.
5. **Add Local Data.** Add any local data that will be used by the class in its member clw file.
6. **Add ProDomus Class Generator Extension.** Add the class generator extension to the procedure.
7. **Fill In Template Prompts.** See the ProDomus Class Wrapper Generator and Shell Class File Generator topics for instructions on filling in specific prompts in the template.
8. **Generate All.** From the Project Menu, select Generate All. It is important to select Generate All to be sure that all code is generated.
9. **Modify Templates and Class Files.** If there are further modifications you want to make to the generated template and class files, make the modifications.
10. **Register Template.** Select Setup/Template Registry to register the template.
11. **Copy Class Files to Libsrc Directory.** Copy the generated .inc, .clw, and .trn files to the libsrc directory.
12. **Refresh Application Builder Class Information.** Before using the class files, select the global classes tab on your application and select Refresh Application Builder Class Information. This is necessary whenever you change class definitions.

TEMPLATE FILES

See Also:

Global Prompts

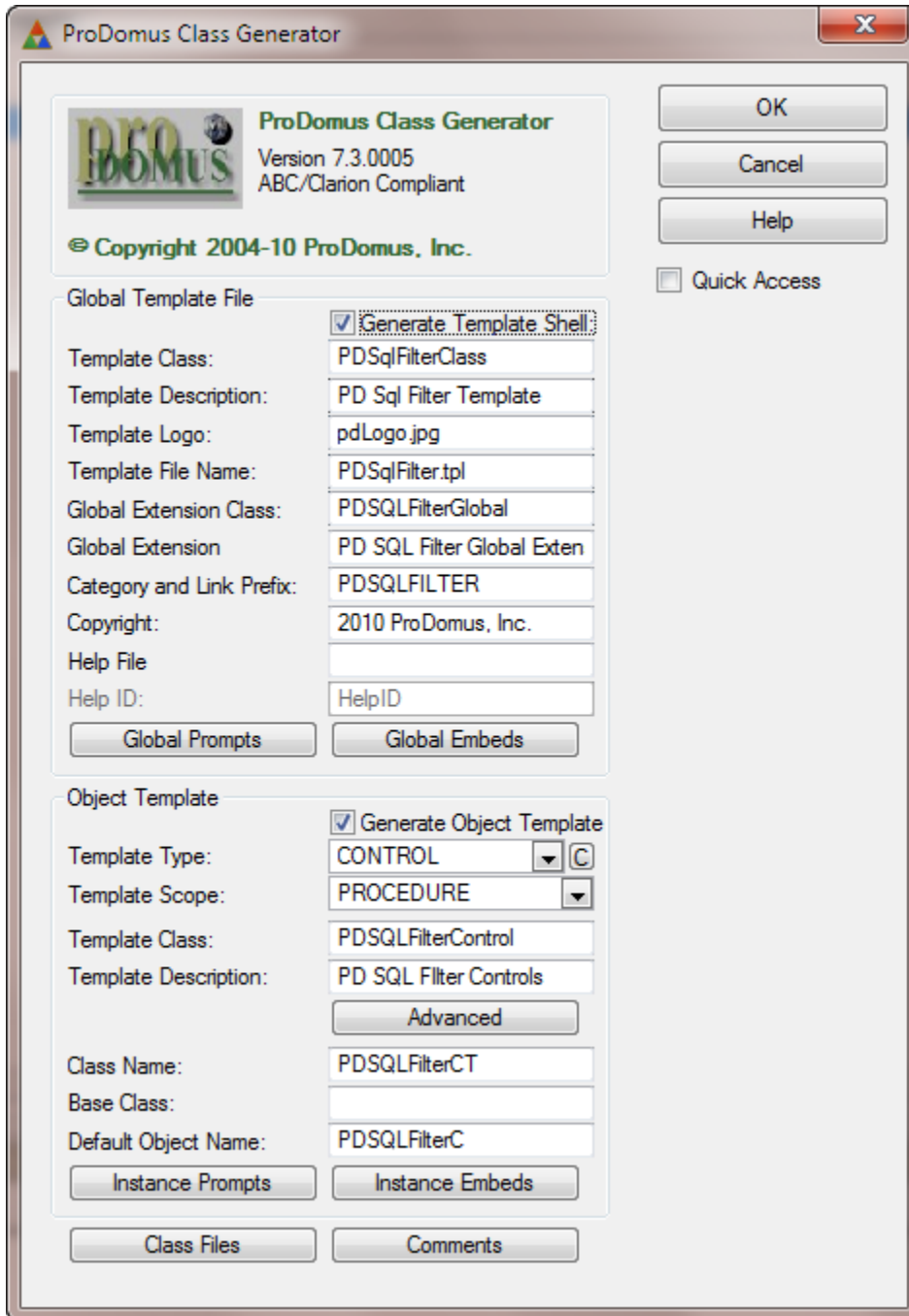
Global Embeds

Control Template

Instance Prompts

Instance Embeds

Class Files



Global Template File

Template Class. Enter the Template Class name (no spaces). This is the Class name that appears in #TEMPLATE.

Template Description. Enter the template description.

TemplateLogo. Logo to display in template.

Template File Name. Enter the name of the template file. Template code will be generated into this file.

Global Extension Class. A global template will be generated. Enter the Class Name (no spaces). This will be the first parameter of #EXTENSIO(Class,Description).

Global Extension. Enter the global extension description.

Category and Link Prefix. Enter the category and link prefix (no spaces). This entry will be used for both the category parameter of !ABCIncludeFile(CATEGORY) and the LinkMode and DLLMode equates. The template will make the Category upper case as required by the clarion environment.

Copyright. Enter copyright year and name.

Help File. Enter the name of the help file.

Help ID. Enter the help ID for the template.

Global Prompts Button. See the Global Prompts topic.

Instance Prompts Button. See the Instance Prompts topic.

Object Template File

The object template file adds template code to instantiate one or more objects. If it is an application extension, the code will be integrated with the Global Extension. Otherwise, a separate extension, code, or control template will be created.

Template Type. Select the type of template that instantiate the object.

“C” Button. If the template type is a control template, press the C button to enter the control properties. Controls should be added to the window associated with the Class Generator extension.

Template Scope. If the Template Type is an extension template, select whether it is an Application template or a Procedure template.

Class Name. Enter the label of the class to be generated.

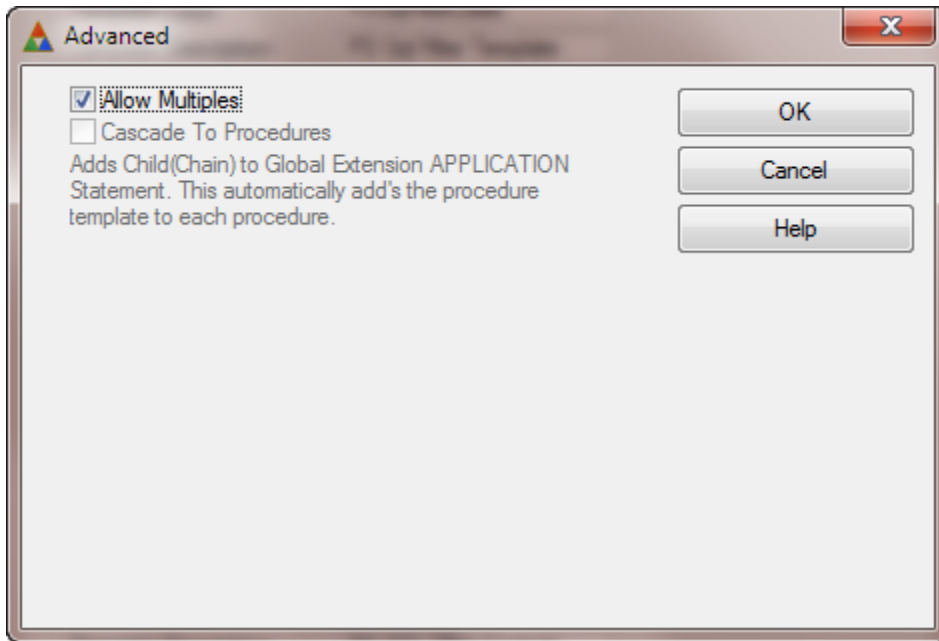
Base Class. If the class is a derived class, enter the base class.

Default Object Name. Enter the name of the default object.

Instance Prompts. See the Instance Prompts topic.

Instance Embeds. See the Instance Embeds topic.

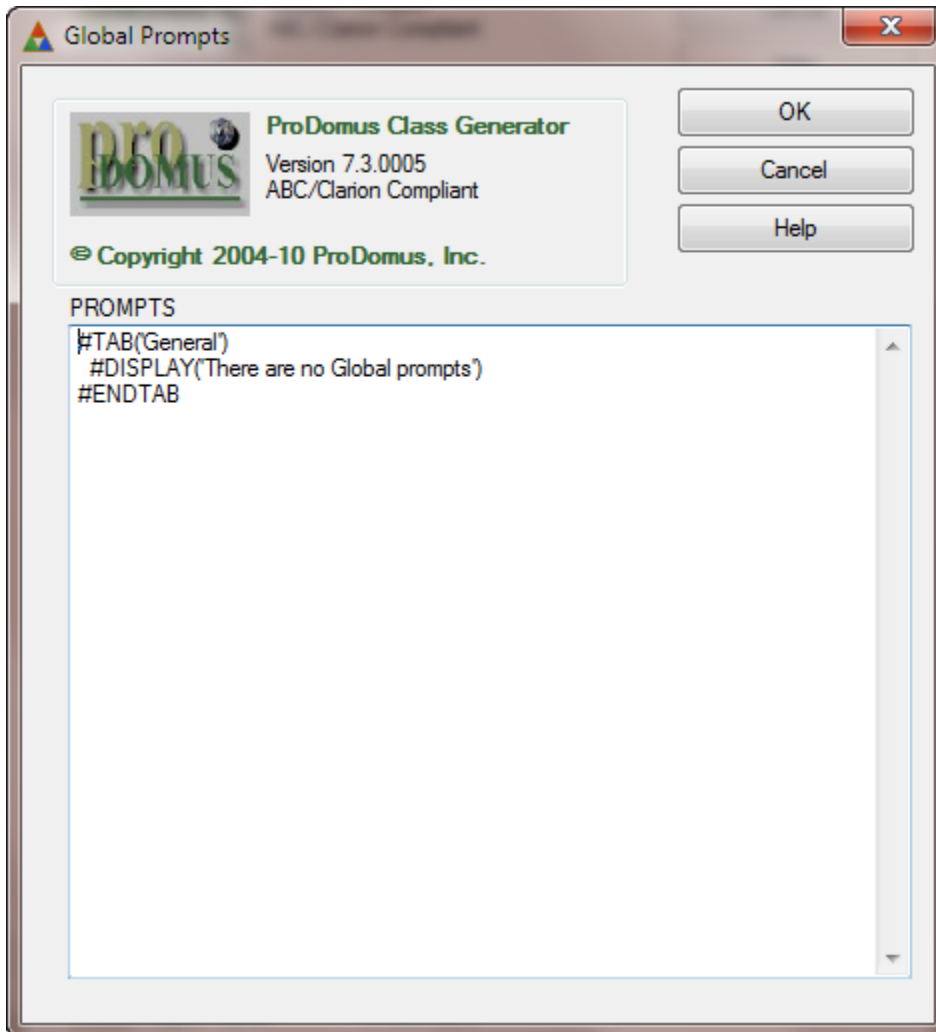
Shell Class Files. See the Shell Class File Generator Topic.



Advanced Button

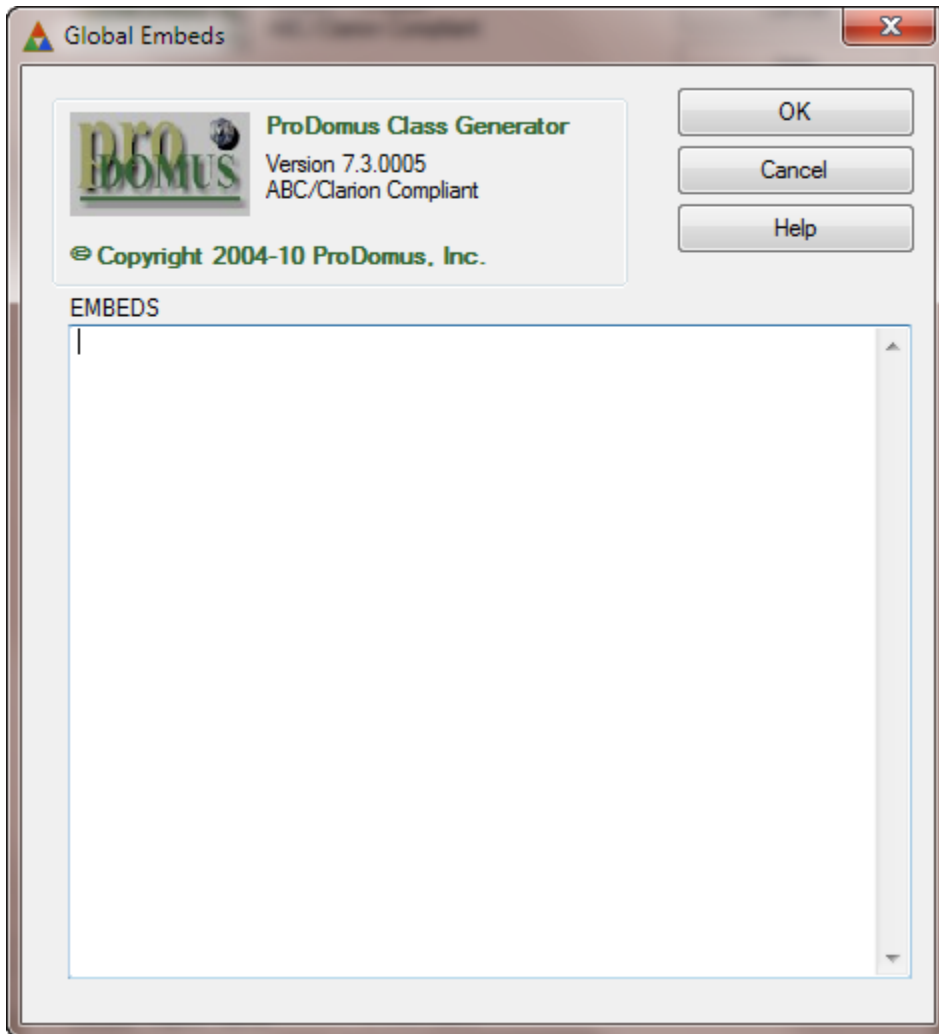
- **Allow Multiplies.** Click to allow multiple instances of the object template.
- **Cascade to Procedures.** In the case of a global extension, click to have the template applied to all procedures.

Global Prompts



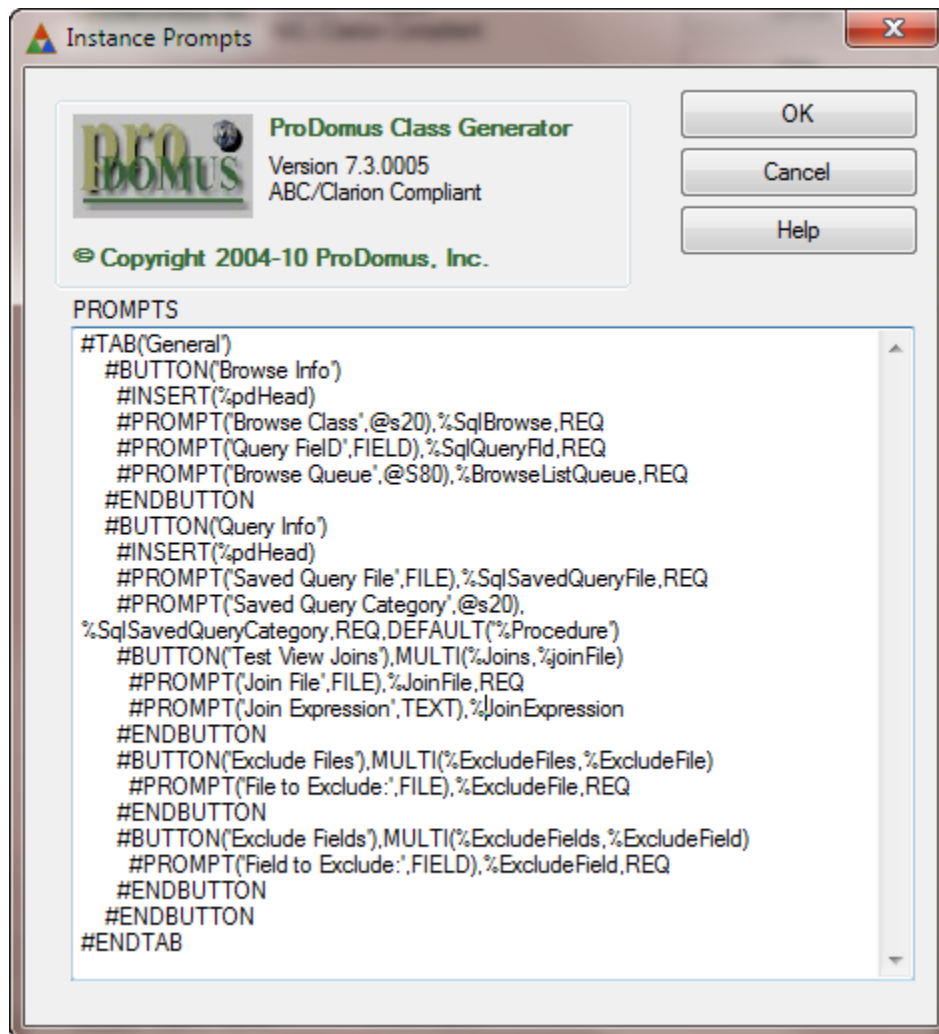
Enter any prompts and prompt related template code for the Global Extension. Note that any code entered will be surrounded with #SHEET and #ENDSHEET.

Global Embeds



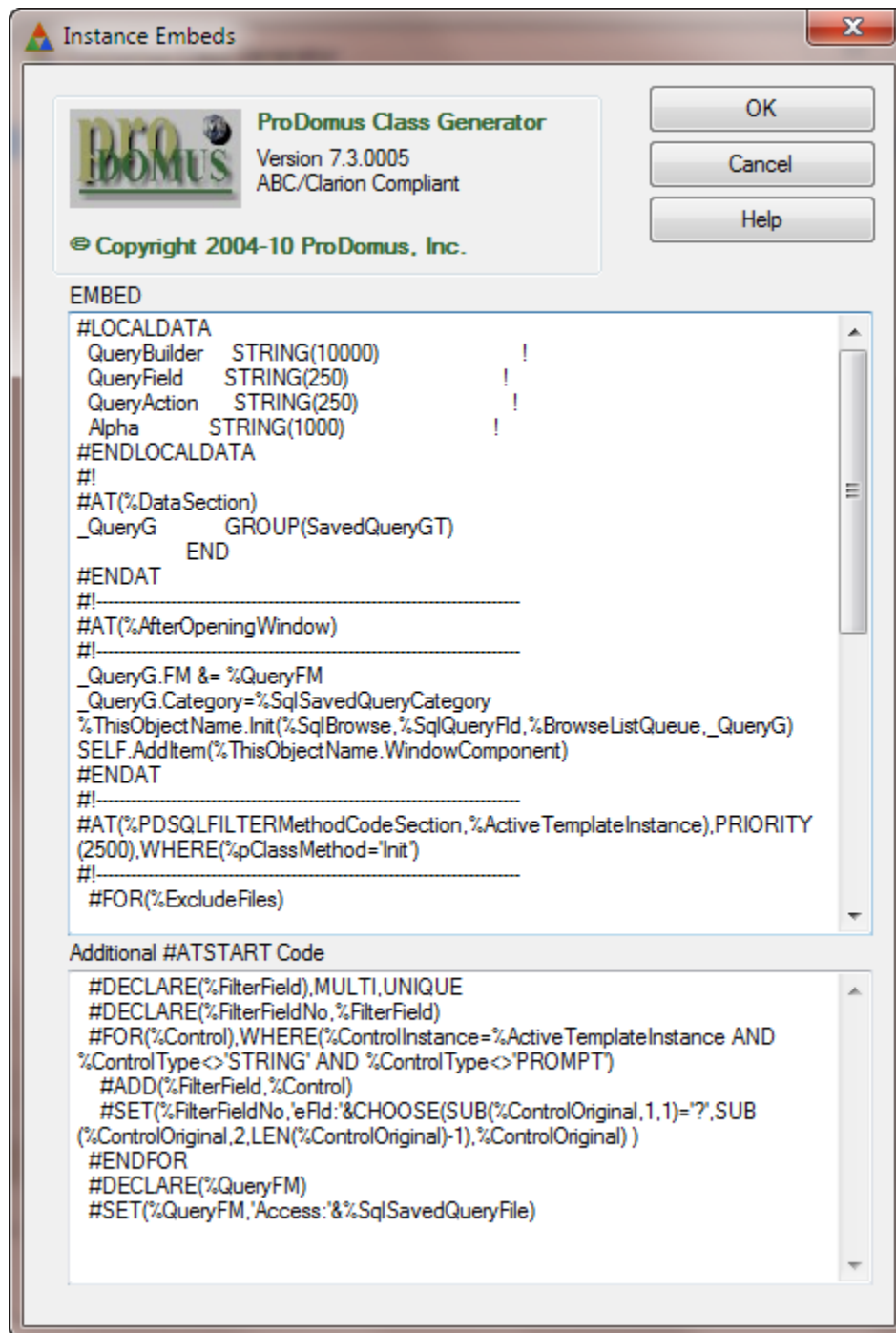
Enter template code that will generate code in global embed points.

Instance Prompts



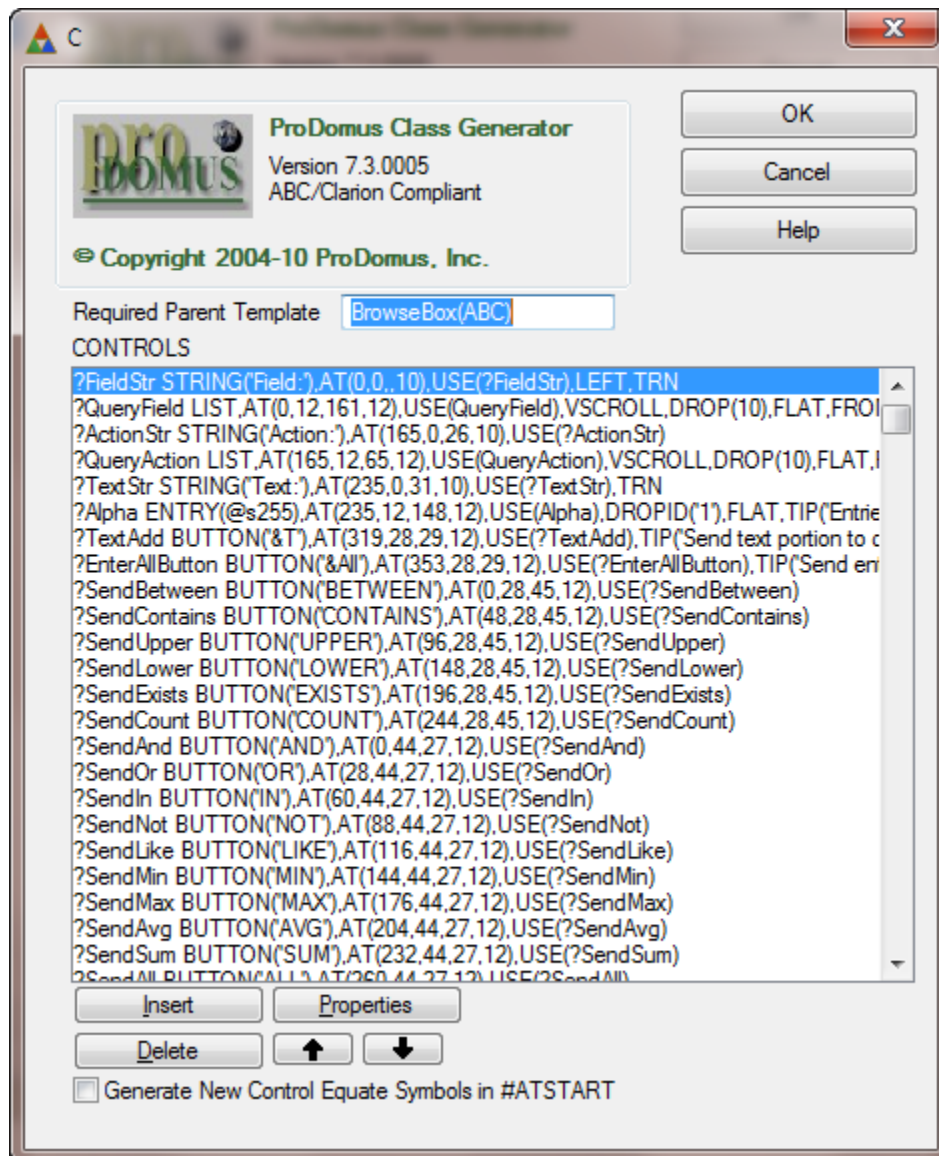
Enter template prompt code related to the class instantiation.

Instance Embeds



Enter any embed code related to the instance.

Control Template

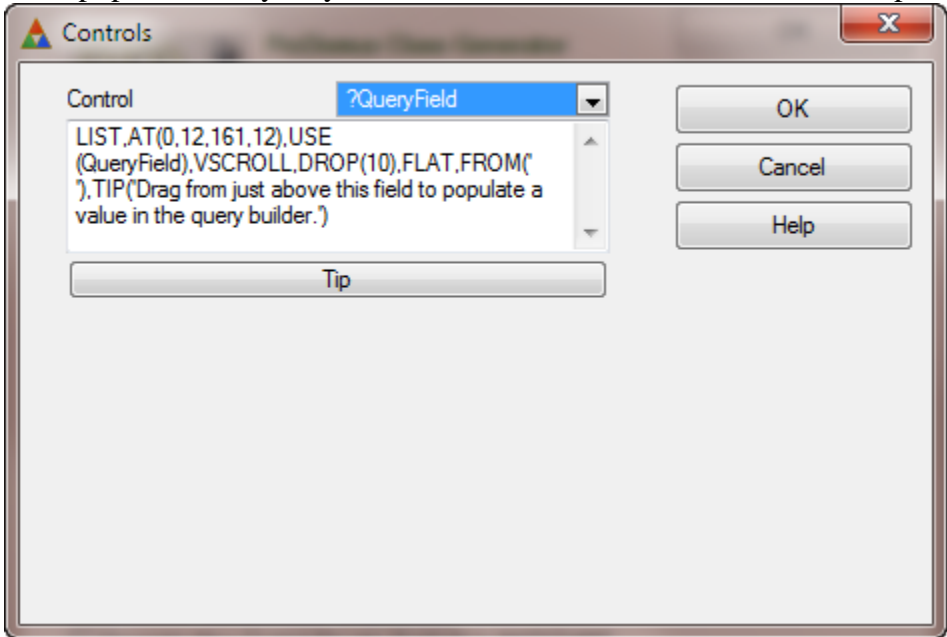


Controls may be populated from the window to which the template is attached. When the template is generated, relative control positions, as required by control templates, will be automatically calculated. The first control's X and Y positions will be blanked.

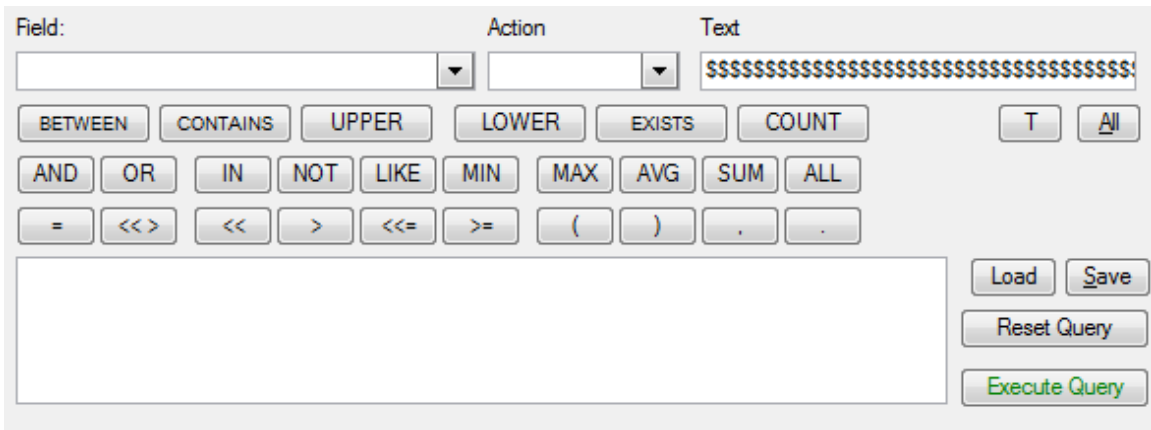
Steps

- **Required Parent Template.** If a parent template is required, enter it's name.
- **Controls.** Add any controls that are part of the control template.
- **Generate New Control Equate Symbols.** If you want to use the Control Field Equate symbols in your template code, you can check Generate New Control Equate Symbols. This creates new control symbols for each control that capture any field equate name changes the end user might have added. The new control name is the original control name with "Set" added, i.e. the symbol for ?FieldStr becomes %FieldStrSet.

Controls may be populated on a window associated with the Class Generator template. Once populated, they may be selected for inclusion in the control template.



NOTE: If the properties of any control change in the window, reselect the control in the list of controls to refresh the control statement.



Enter the properties of each control to be included in the control template.

CLASS FILES

See Also

Translation Files

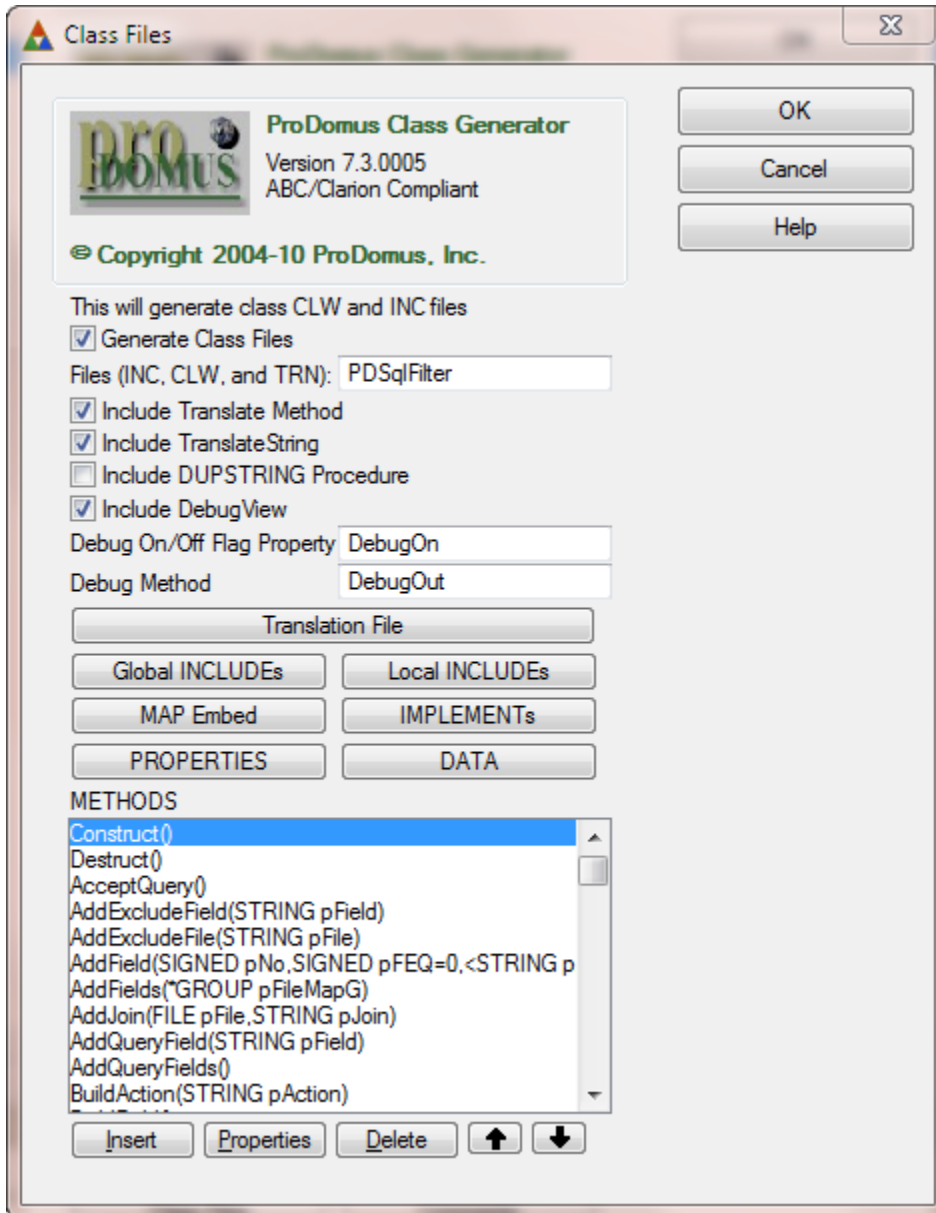
Global Includes

Local Includes

Map Embed

Implements

Properties
Data
Methods



The Class Generator Template can generate class source code consisting of an ABC/Clarion compliant include file (.inc), a translation file (.trn), and member source code file (.clw).

Generate Class Files. Check to have the template generate the class files.

Filed Label. The name of the file excluding the file extension. Translation, include, and member source code files will use this name.

Include Translate and TranslateString Methods. Check these to include translation methods in the class. The template will create calls to Translator in the virtual embeds if the Enable Runtime Translator is turned on in Global Application Properties. Any

window in the class should include a call to SELF.Translate(TheWindow) and any literal strings should call the SELF.TranslateString(TheString).

Include DUPSTRING procedure. This is a handy procedure included in some ABC classes to generate a reference string with a value.

Translation Files Button. See Translation Files. Use this code the contents of the translation file.

Global Includes Button. See Global Includes. Use this to identify files to be included in the class include file. If the file has a section reference, it may be entered as “thisFile’,’thisSection. The template will add beginning and trailing single quotes.

Local Includes Button. See Local Includes. This will add include files to the member .clw file. Note the class definition include file will automatically be added.

Map Embed Button. See Map Embed. Add any items that should be included in the MAP structure.

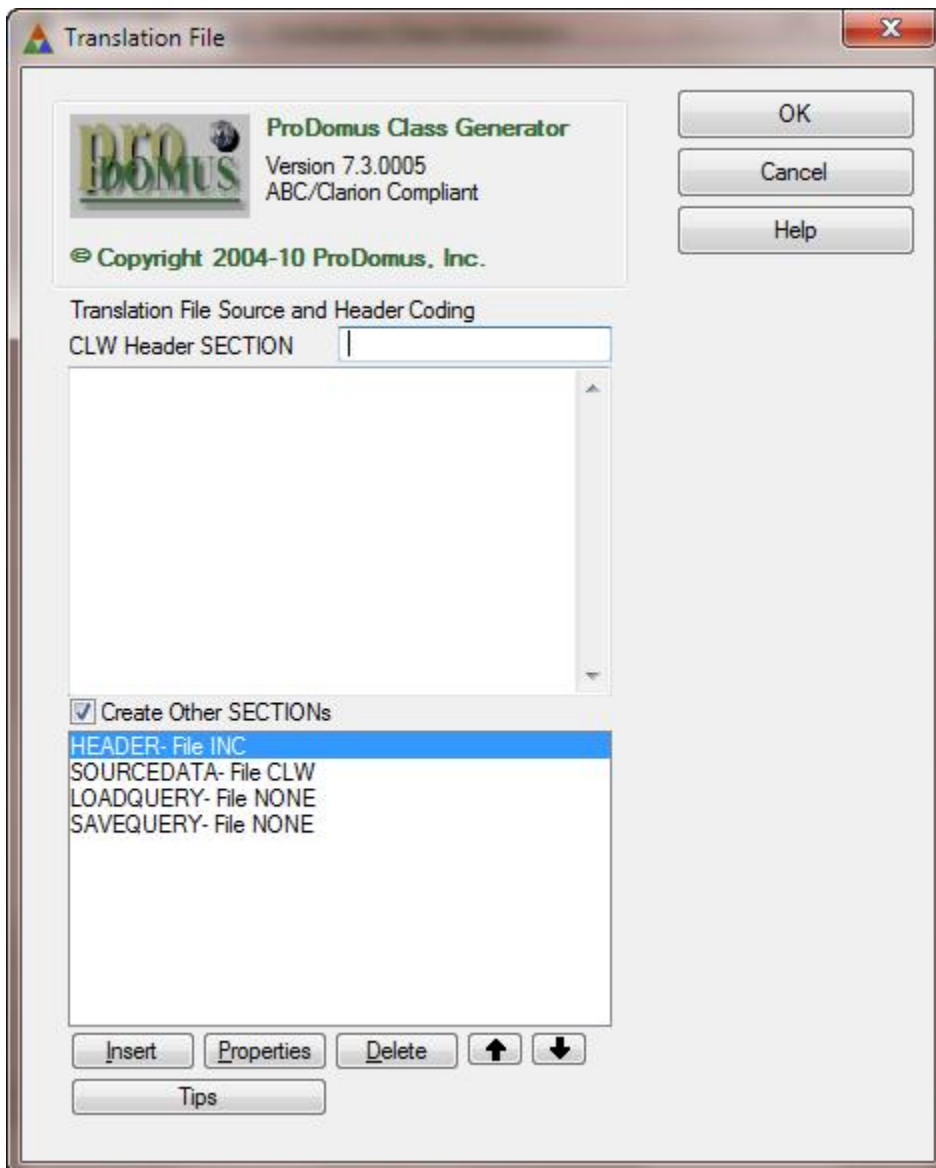
Implements Button. See Implements. Add any interfaces to be implemented by the class.

Properties Button. See Properties. Add properties to be added to the class definition file.

Data Button. See Data. Allows you to add data to either the include file or the clw file. Data may be selected global data declared class generating application or hand coded.

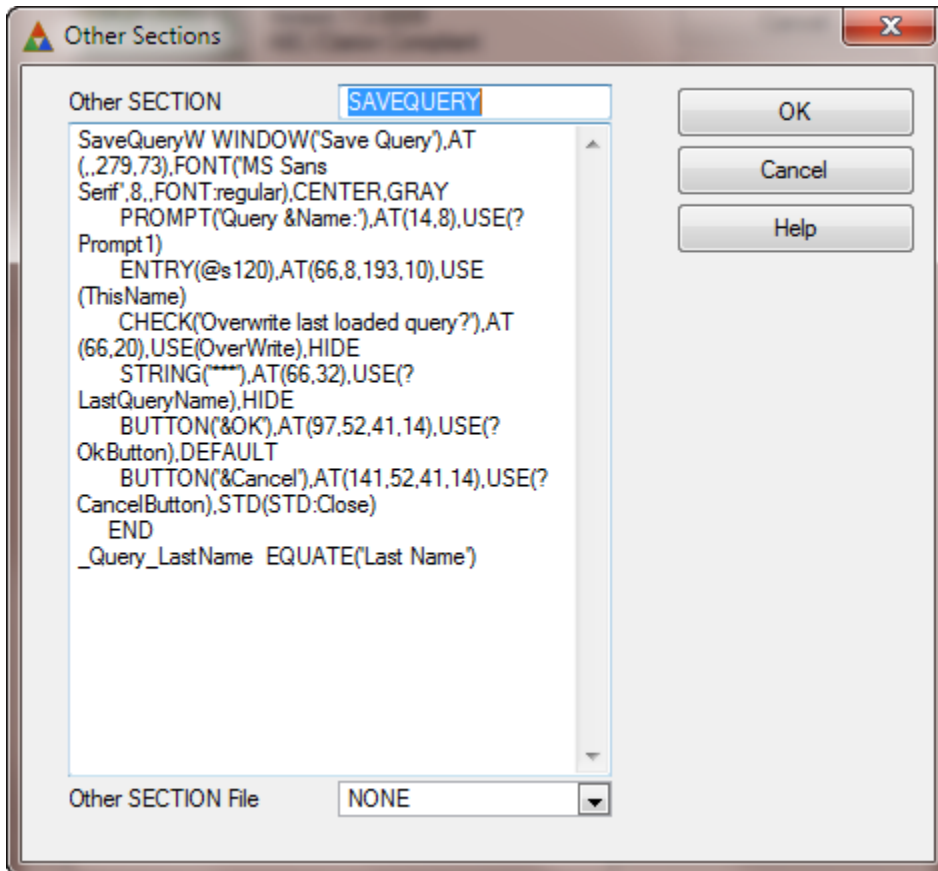
Methods. See Methods. Use this method to declare the methods that will be included in the class.

Translation Files



Translation files contain equates or other declarations that a developer can use to customize the user interface or behavior of the class library. If changes are made, the file should be copied to a directory with common changes or the application directory so that they do not get overwritten. Examples are `Aberror.trn`, `Abbrows.trn`, `Abquery.trn`, and `abtoolba.trn`. These files can then be customized by the developer.

A useful coding tool is the compiler `SECTION` directive. This, according to the Clarion help file, identifies the beginning of a block of executable source code or data declarations which may be included in source code in another file. The `SECTION`'s string parameter is used as an optional parameter of the `INCLUDE` directive to include a specific block of source code. A `SECTION` is terminated by the next `SECTION` or the end of the file.



The Class Generator allow you to create multiple sections and specify whether these should be located in the Class CLW file or INC file headers or hand coded. Hand coded include sections would normally be associated with specific methods, i.e. a method window definition that the developer could then customize.

If the class file contains windows, it can be helpful to the end developer to include the window declaration in the TRN file so that it can be customized. If it includes translation pairs, it is useful to include the TRN file or Section of the TRN file in the class include file.

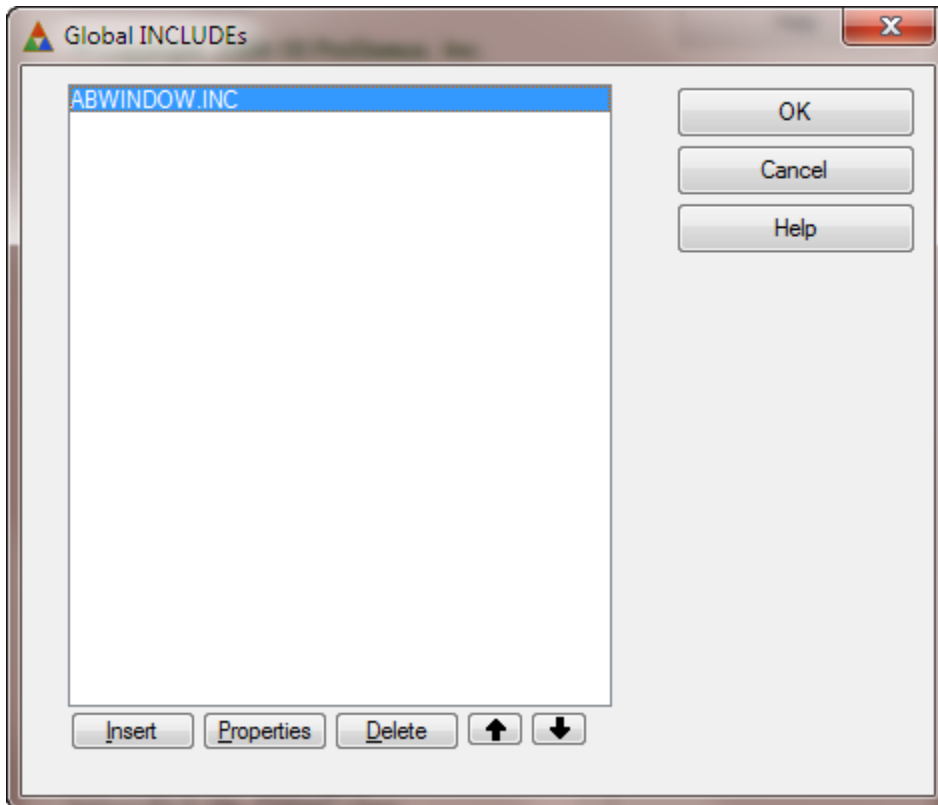
Include in Data Section. If checked, the translation file will be included in the data are of the class include file. This would typically be translation pairs to be that can be used by the TranslatorClass.

Section. The parameter of the compiler SECTION directive if just a SECTION in the translation file should be included in the class include file.

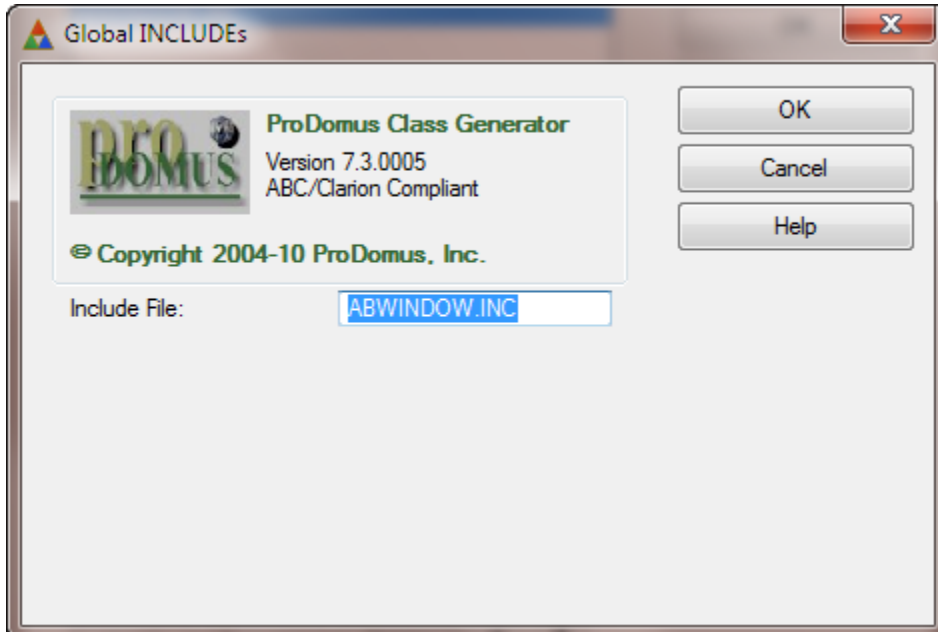
Text Entry. Enter the code that should be included in the translation file including any sections.

Other Section File: Specifies that the file gets included in the CLW, INC header or in does not get included,

Global Includes



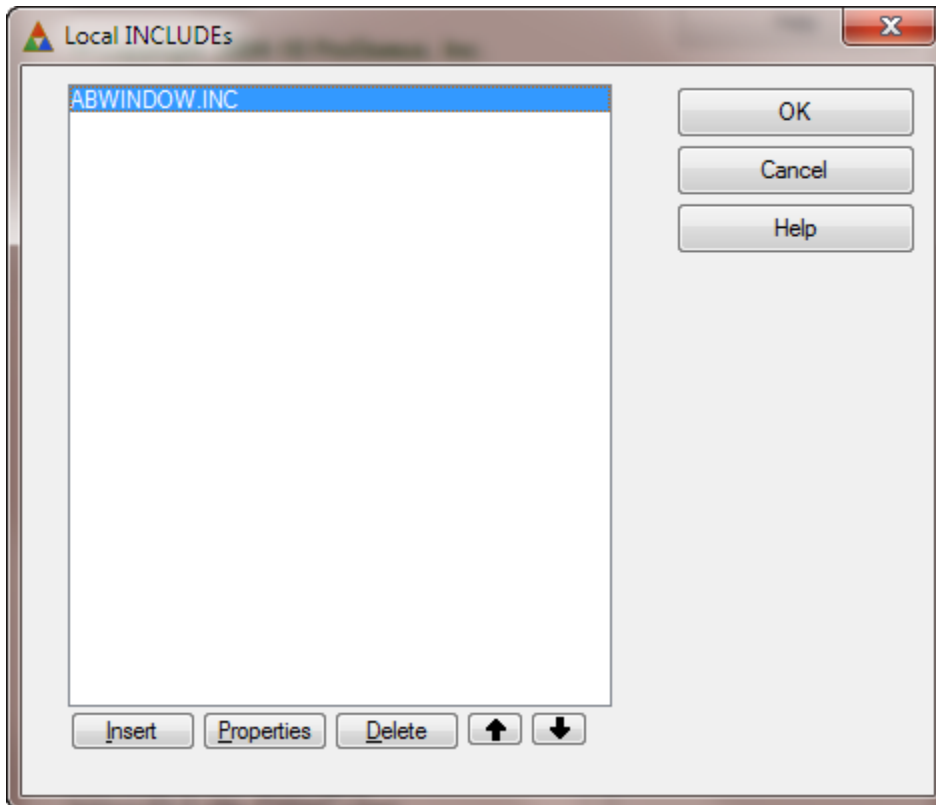
List of files to be included in the class include file.



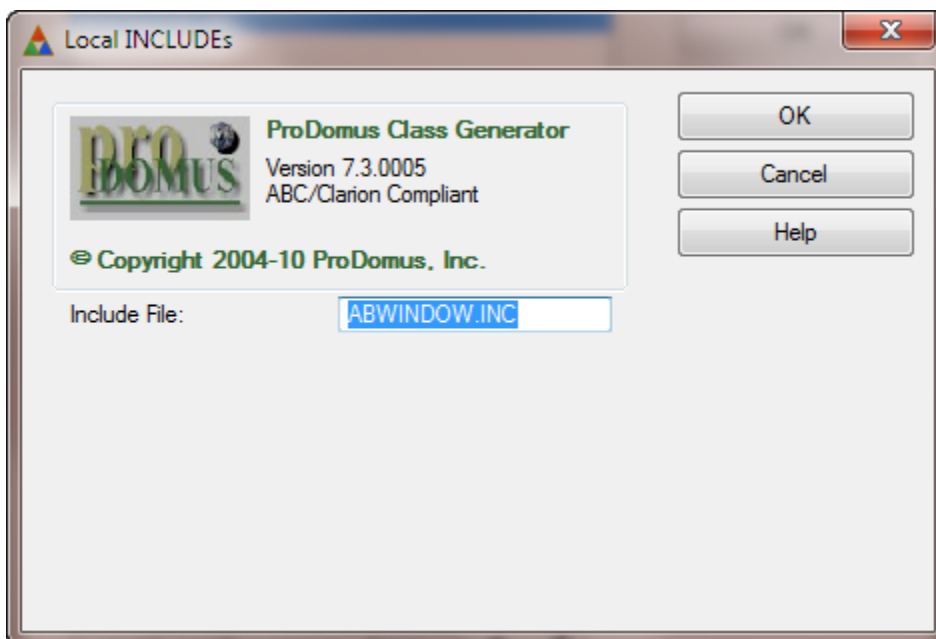
If the file has a section reference, it may be entered as "thisFile", 'thisSection. The template will add beginning and trailing single quotes. It will also add the "ONCE" attribute.

TIP: If the ABC Window Manager window component interface is implemented in the class, the ABCWINDOW.INC should be included in the INC file header.

Local Includes

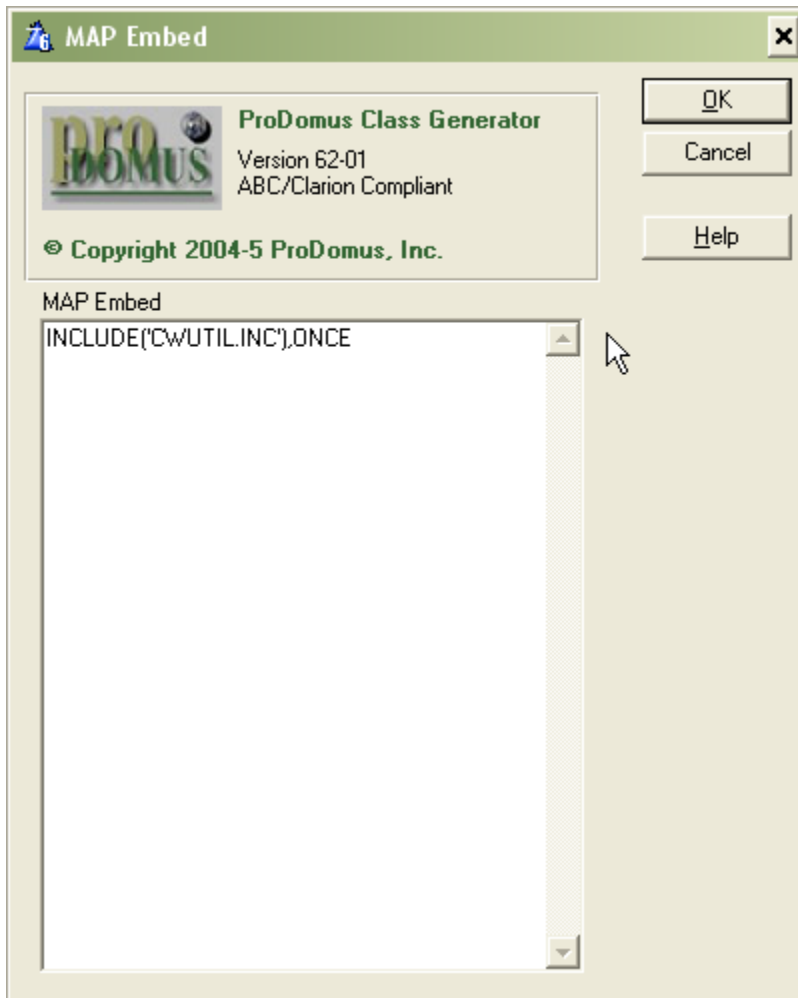


List of files to be included in the class member clw file.

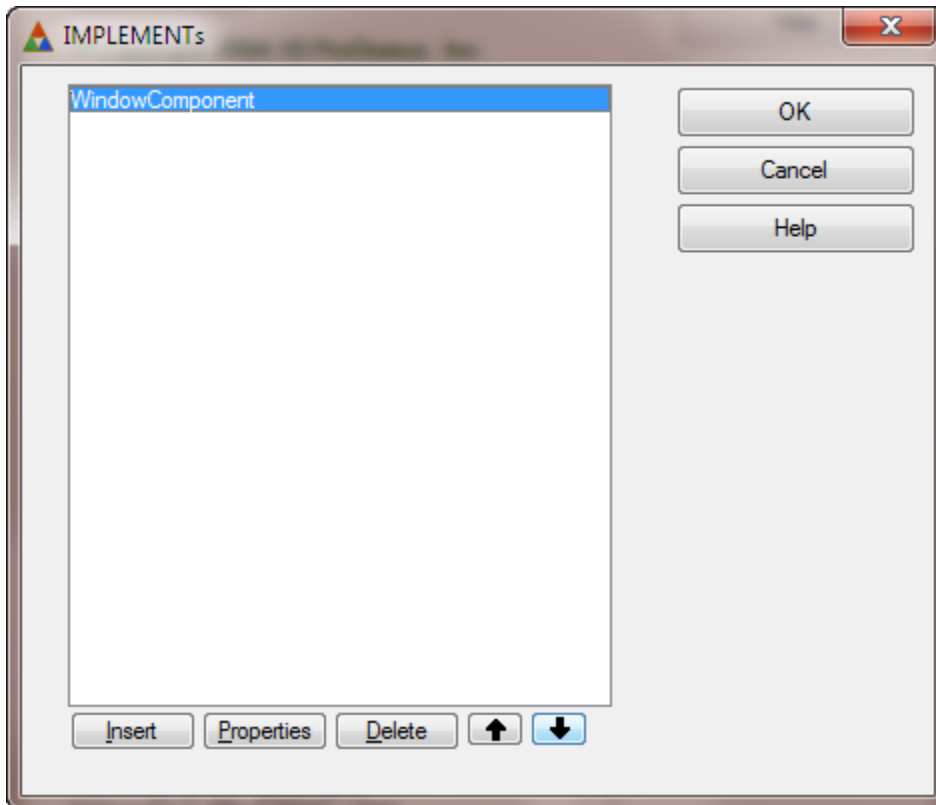


If the file has a section reference, it may be entered as "thisFile", 'thisSection". The template will add beginning and trailing single quotes. It will also add the "ONCE" attribute. The Class include file will be included automatically in the CLW file.

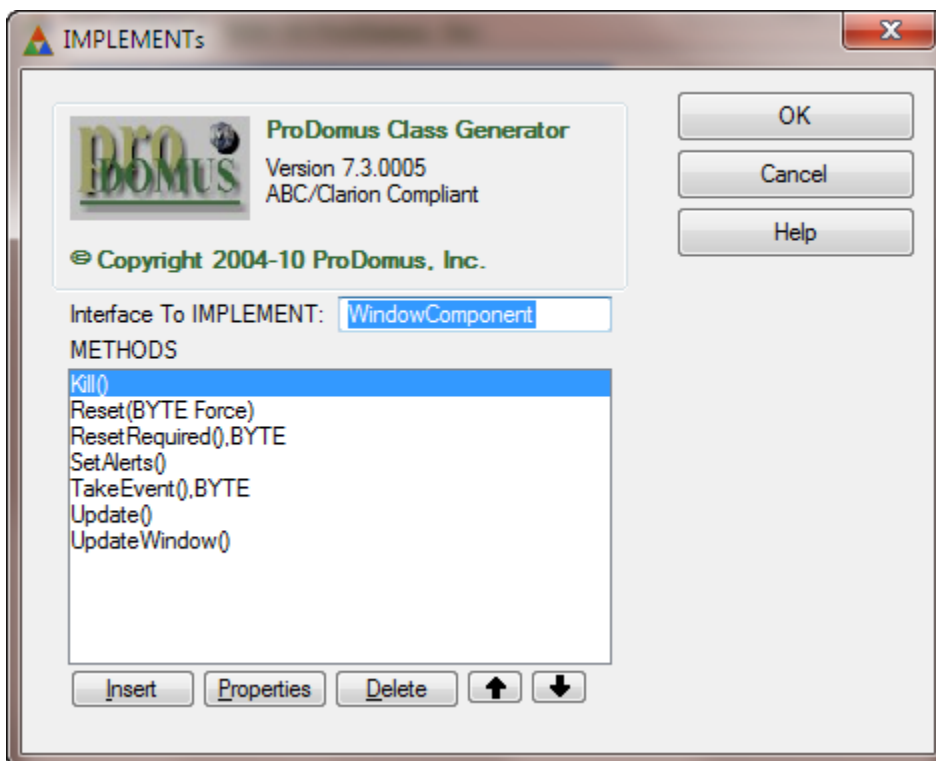
Map Embed

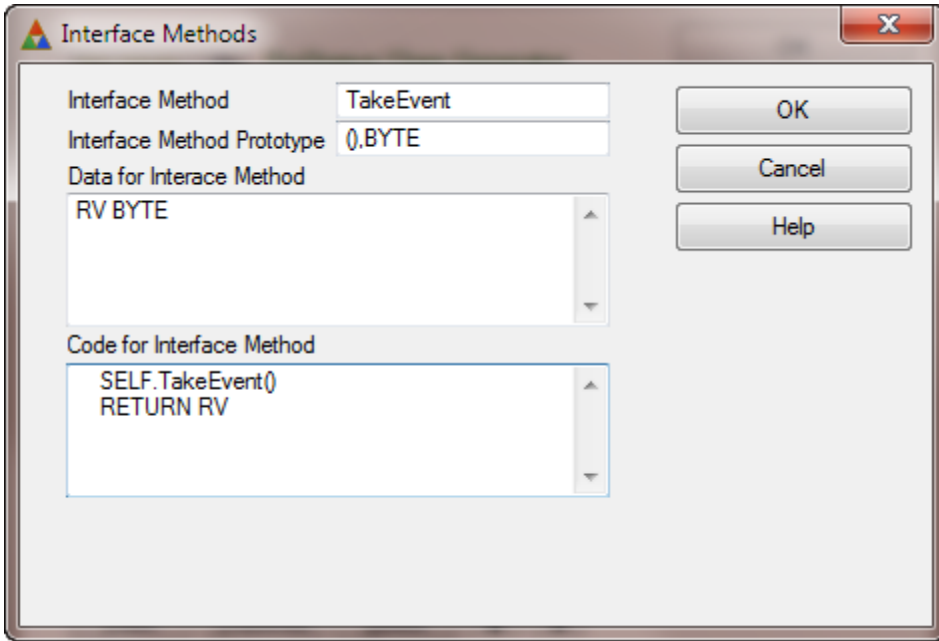


Implements



List any interfaces and each of its related methods to be implemented by the class.





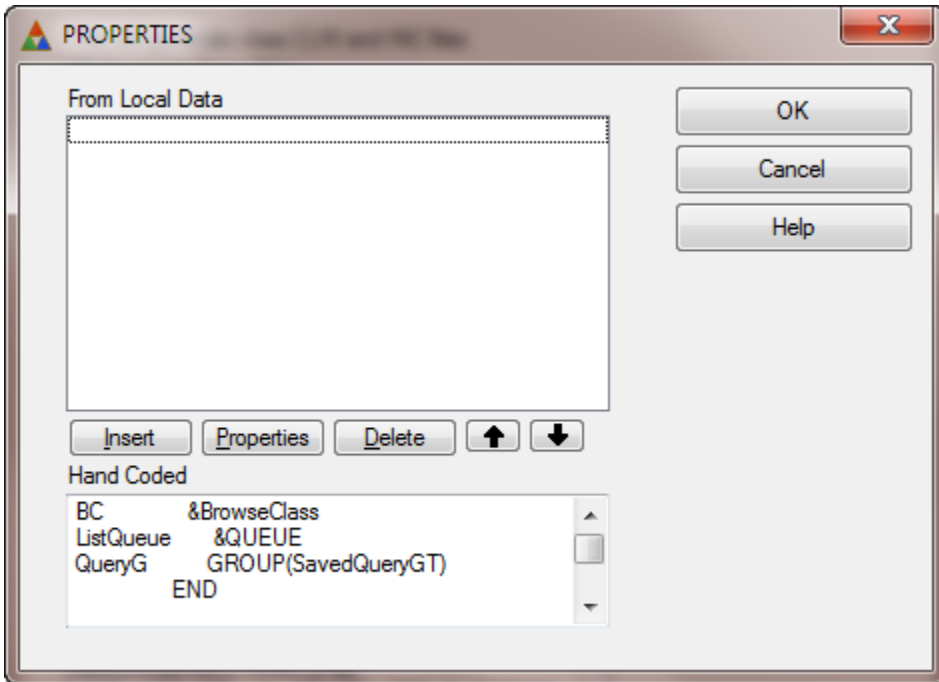
Interface Method. The method label for the interface. Note that the template will add the class label and interface label to the method.

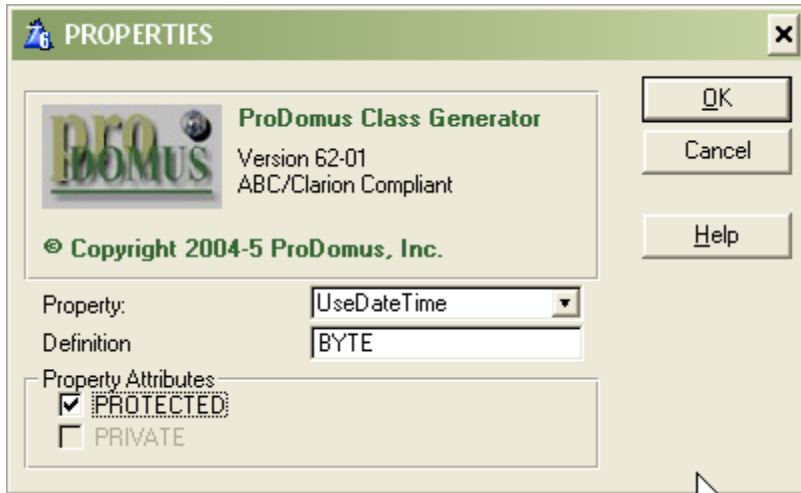
Interface Method Prototype. The method prototype including surrounding parentheses.

Data for Interface Method. Data section code for the interface method.

Code for Interface Method. The code for the interface method.

Properties





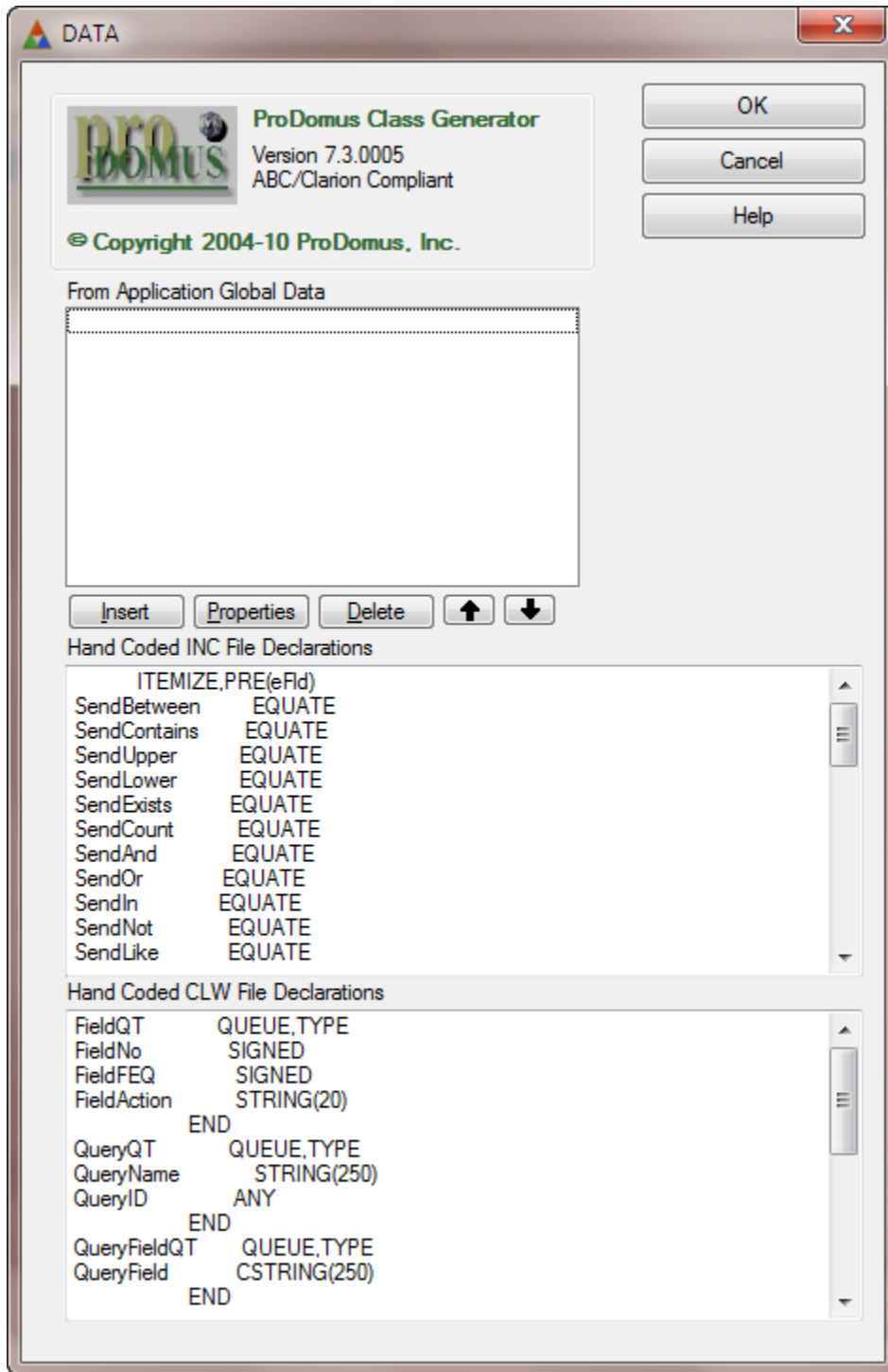
Class properties entry form.

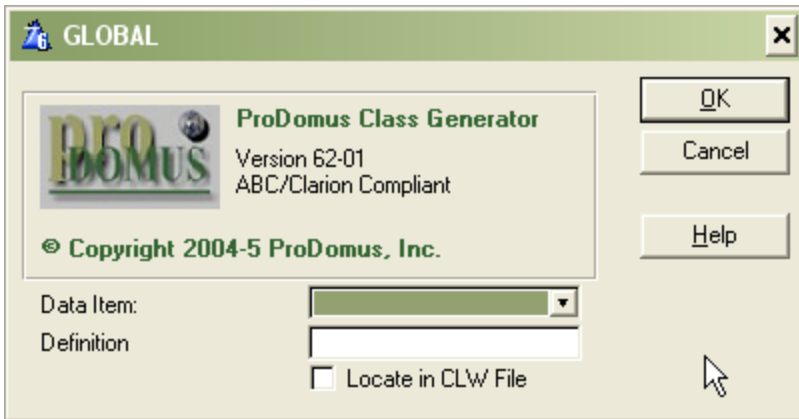
Property. The drop list includes data that has been added to procedure. When a data item is selected, its definition will be entered as the default definition in the Definition entry.

Definition. The definition of the property.

Property Attributes. By default, the property will be public. Check whether it should be protected or private.

Data





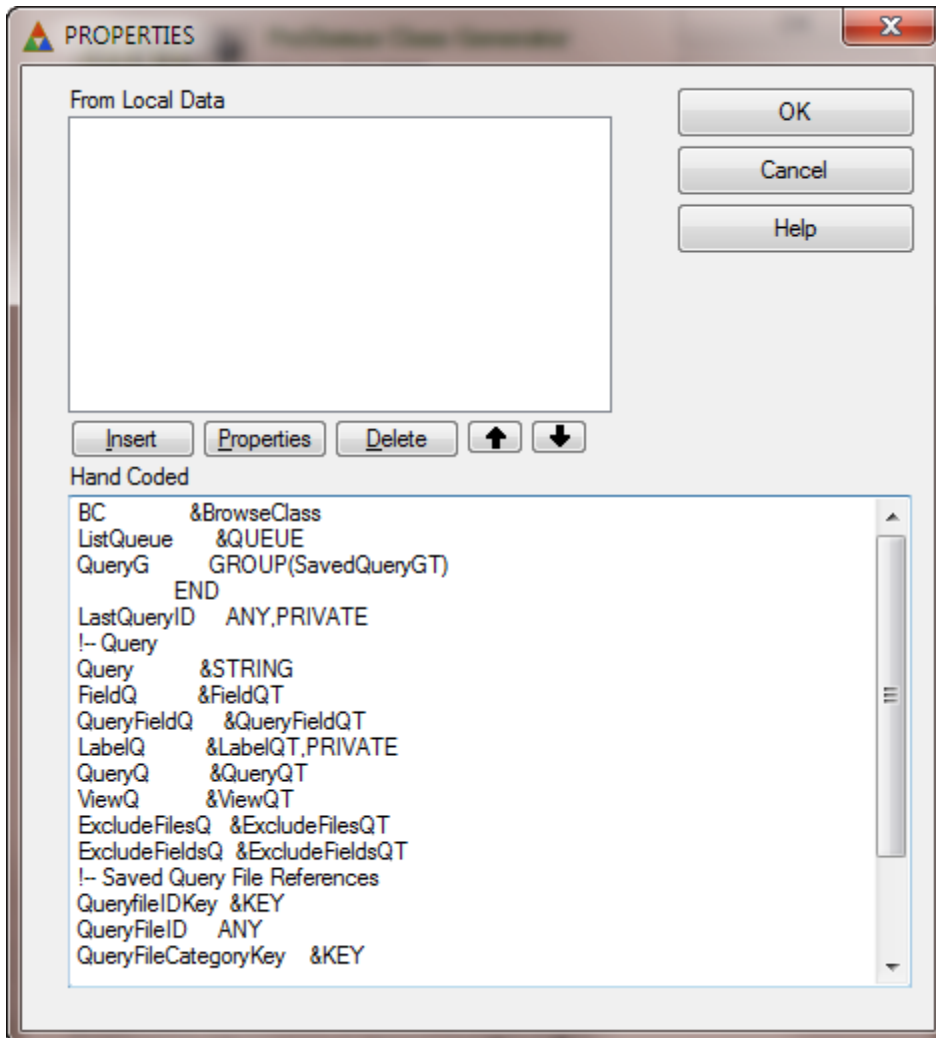
The form displays a drop list of data declared as application global data. The data and its declaration may be included in the INC file (the default) or the class CLW file.

Data Item. List of fields declared as global data.

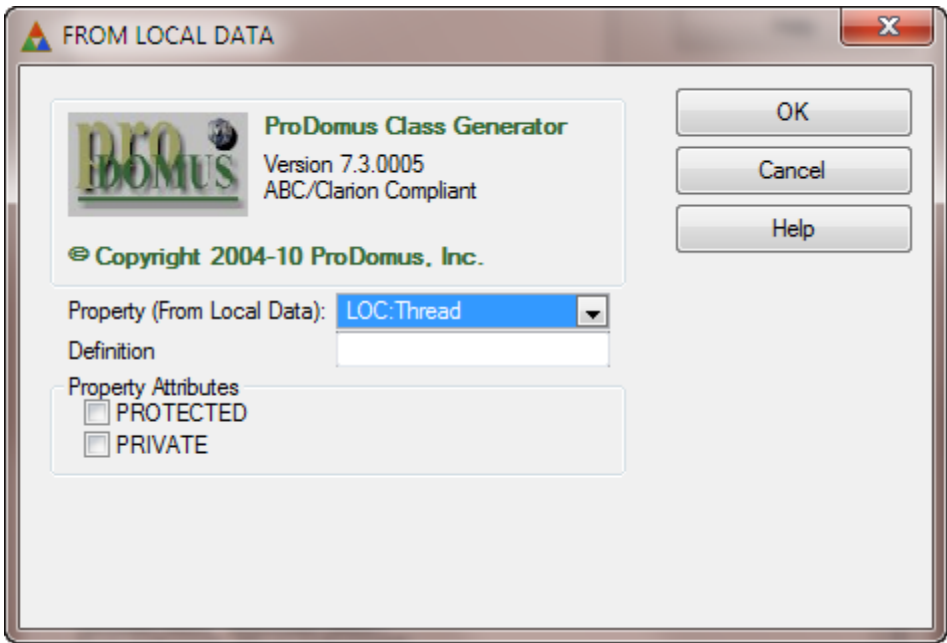
Definition. The data type declaration.

Locate in CLW File. Check if the declaration should appear in the CLW file.

Properties



Instance properties are entered on the properties window. Properties may be copied from the Procedures Local Data or hand coded.



Properties from local data.

Methods

Methods

ProDomus Class Generator
Version 7.3.0006
ABC/Clarion Compliant

© Copyright 2004-10 ProDomus, Inc.

Method:

Prototype:

Attributes

- VIRTUAL
- PROTECTED
- PRIVATE
- PROC
- Show in Embed Tree
- Hide from Embed Tree
- REPLACE Base Method

DATA

CODE - Please indent or line will be in column 1. This allows lines such as ? ASSERT(expression,Description)

```
SELF.BC &= pBC
SELF.ListQueue &= pListQueue
SELF.Query &= pQuery
SELF.QueryG = pQueryG
SELF.AddQueryFields
SELF.GetQueryActions
SELF.GetQueryFileInfo
SELF.LoadQueryQ()
```

Template Embeds

OK
Cancel
Help

Entry form for generating class methods. Each method and its prototype will be included in the class definition INC file. The code will be generated in the class implementation CLW file.

Method. The method label.

Prototype. The method prototype including parentheses.

Attributes. Standard method attributes. By default, the method is public. Other methods may be hidden from the embed tree by checking "Hide from Embed Tree."

Show in Embed Tree. Display a public method in the embed tree. Adds "!,EXTENDS."

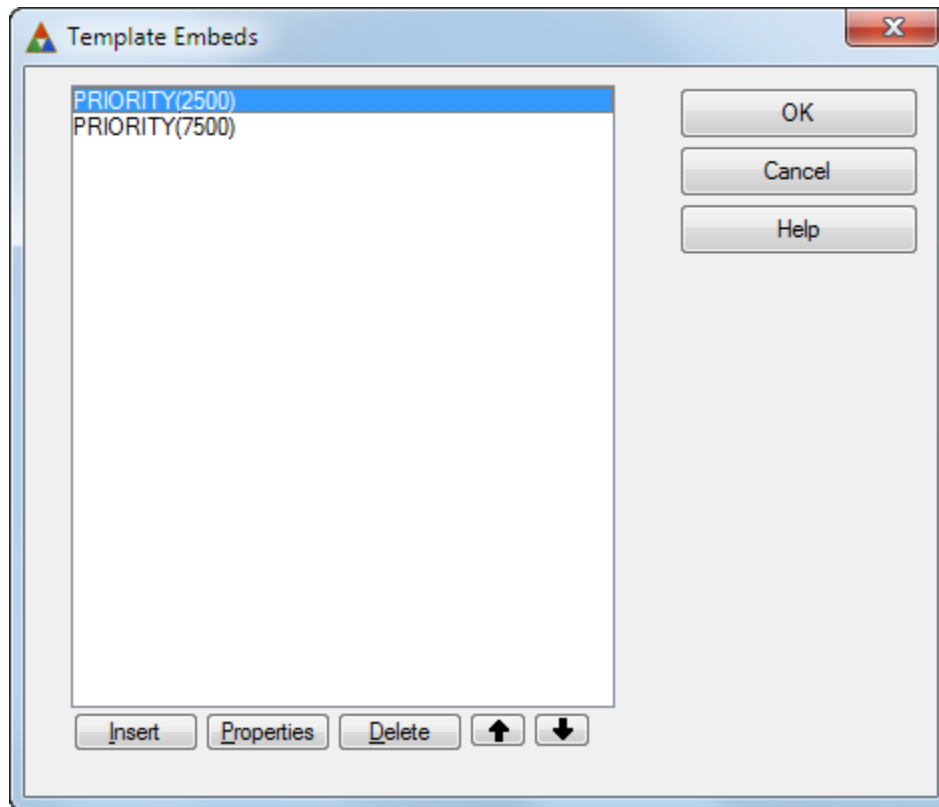
Hide from Embed Tree. Hide Protected or Virtual method from embed tree. Adds "!,FINAL"

REPLACE Base Method. Specifies the "Construct" or "Destruct" PROCEDURE in the derived CLASS completely replaces the constructor or destructor of its parent CLASS.

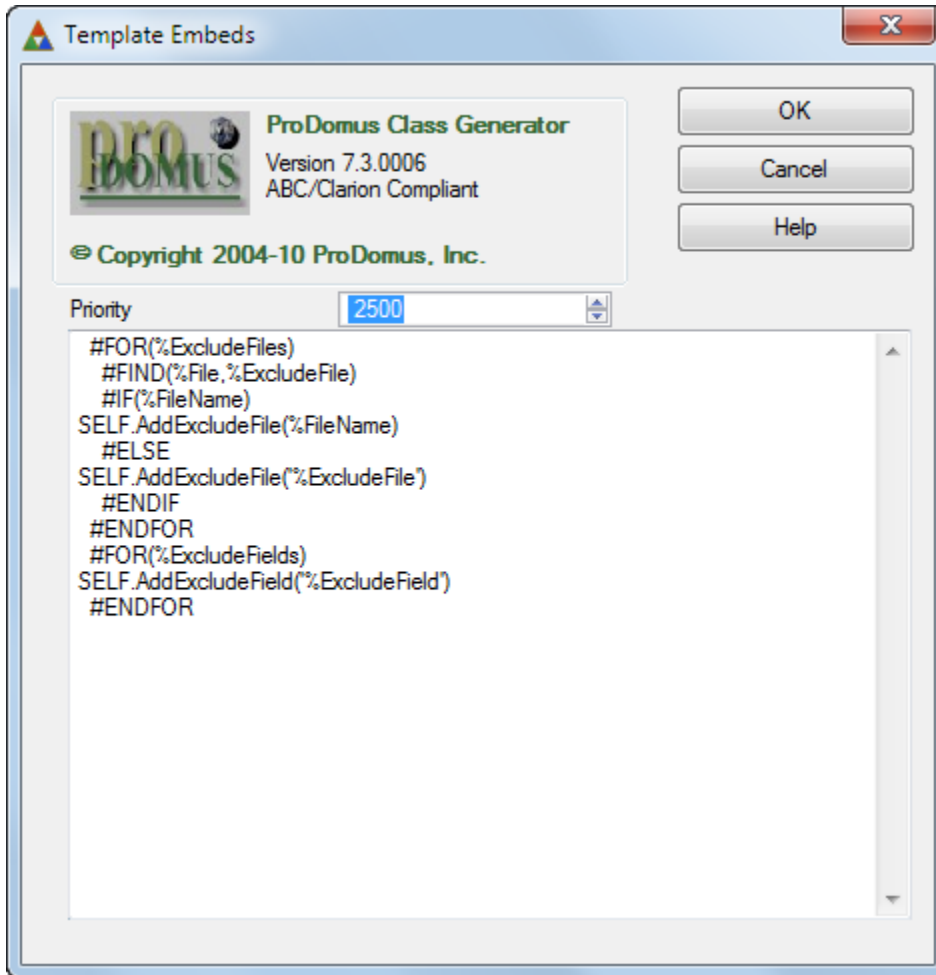
Data. Declare any data that should appear in the Method Data Section.

Code. Method Code Section code. Note that code will be generated in column 1.

Template Method Embeds



Template embeds place template code in the virtual method for the selected procedure. This facilitates putting additional template code into the specific method and its prototype.



You can specify the PRIORITY and the specific code. Note that a priority less than 5000 will put the code before the Parent call. Greater than 5000 will be after the parent call. This generates template code similar to the following:

```
#!-----
#AT(%PDSQLFILTERMethodCodeSection,%ActiveTemplateInstance),PRIORITY(7500),%|
WHERE(%pClassMethod='Init' AND &|
%pClassMethodPrototype='(BrowseClass pBC,*STRING pQuery,*QUEUE
pListQueue,SavedQueryGT pQueryG)')
#!-----
#FOR(%FilterField)
SELF.AddField(%FilterFieldNo,%FilterField)
#ENDFOR
#FOR(%Joins)
SELF.AddJoin(%JoinFile,'%JoinExpression')
#ENDFOR
#ENDAT
```

This saves your having to code these complex #AT statements.

Examples

PD SQL Query Template and Class

The screenshot shows a graphical user interface for creating SQL queries. It includes a 'Field:' dropdown menu, an 'Action:' dropdown menu, and a 'Text:' input field. The 'Text:' field contains a long string of dollar signs. Below these are several rows of buttons for SQL operators: 'BETWEEN', 'CONTAINS', 'UPPER', 'LOWER', 'EXISTS', 'COUNT', 'T', 'All', 'AND', 'OR', 'IN', 'NOT', 'LIKE', 'MIN', 'MAX', 'AVG', 'SUM', 'ALL', '=', '<<>', '<<', '>', '<=>', '>=', '(', ')', '.', and '-'. At the bottom right are buttons for 'Load', 'Save', 'Reset Query', and 'Execute Query'.

This template was derived from a Procedure developed by Robert Huff. It essentially creates an SQL Filter from fields in a view file using SQL syntax. Each time the filter is modified, it is tested to be sure that it is a valid SQL filter statement. Queries may be saved and reloaded. Fields that can be used in the query can be dragged from a listbox or from a list of file fields. The “Execute Query” button will have green or red text depending on whether the current query is valid or not. The template is designed for use with ABC applications only. Further details can be found in the example folder.

PD Window Manager Class Examples

The Window Manager Class maintains a queue of open windows. A record is added every time a window is opened and deleted when it is closed. The Queue includes the procedure name, a reference to the window, its thread, and whether it is modal or not. The methods do the following:

- Post messages and notifications by using the procedure name, i.e. PDWinC.
Post('BrowseNames',EVENT:MyEvent) or PDWinC.Notify('ClassTree',NOTIFY:ForceReset)
- Determine whether there is a modal window open, i.e. PDWinC.IsModal().
- Relocate a window if it gets lost outside the client area of the main frame. Window positions are checked any time a window opens and when the application frame is resized.
- Change window properties consistently across an application, i.e. change all &Insert button text to &Add.
- Perform thread limiting.

The example has two implementations to illustrate two features of the Class Generation Template.

1. **PDWinQ.** This example creates a Procedure template that instantiates the class at the procedure level. It automatically cascades the procedure template to all procedures by adding the procedure template class name and chain to the APPLICATION attribute of the global extension.

2. **PDWin2Q.** The example is essentially the same class, but the class object is instantiated in each procedure without creating a procedure template for each. The downside of this approach is that it will not generate virtual class embed code in each procedure.

In both examples, procedures to be thread limited are listed in the global extension. The both use the SCHOOL example. Note that relocating windows is Font sensitive; in this case fonts must be MS Sans Serif size 8.

The installation program installs generated templates and source files into template and libsrc directories.

ClarionLive Example

This is a simple time zone class to get and display Universal Time. The demonstration is available on ClarionLive. It illustrates the process of creating class libraries and template, in this case the creation of a code template with a window that is put into a translation file, and the creation of a translate(Window pWin) method. It does not include API calls to get TimeZone information from Windows. The demonstration link is <http://nicetouch.adobeconnect.com/p63146263/?launcher=false&fcsContent=true&pbMode=normal>.

PDClGen.app

This application illustrates population of a control template. There is no underlying class that will compile.

History

NOTE: The first two version number indicate the latest version of Clarion in which the template has been tested.

Version 7.3.0006 January 20, 2011

- Update C7 SQL Query control template example.
- Added multiple entry of method specific template code that automatically generates “#AT” statements for virtual methods. See **Template Method Embeds**.
- Fixed generation of PRIVATE attribute.

Version 7.3.0005 January 4, 2011

- Add required parent template for control templates.
- Add multiple sections and optional locations in translation files.
- Blank x and y positions in first field of control templates.
- Add automatic insertion of Translator into virtual embeds for Translate(TheWindow) and TranslateString(TheString)
- Increased text control sizes make data entry easier.
- New SQL Query Control Template example.
- Misc. minor bug fixes.

Version 7.2.0004 December 14, 2010

- Enhancement: Option to change logo generated.

- Advanced Button for Application Extensions. Contains options for MULTI instances of extension and control templates and option to create object instances in each window procedure without creating procedure extensions for each procedure. Note that in this case, class virtuals will not be generated in for each virtual.
- Advanced Button for Procedure Extensions. Option to add APPLICATION(Child(Cain)) to the global extension. This assigns a designated procedure extension to each procedure.
- Template ATSTART Embed: An entry has been created to add template code to the ATSTART embed in the template.
- Translate Method Check Box. A check box has been added to add a Translate(WINDOW pWin) virtual method to the class methods.
- TranslateString Check Box. A check box has been added to add a TranslateString(STRING pStr) method to the class generated.
- Fix. The global template embed code was being generated in the wrong location.
- New Window Management class added