Naming Conventions for Actuate Basic Report Designs

By Chris Geiss
chris_geiss@yahoo.com
http://www.chrisgeiss.com

Naming Conventions for Actuate Basic Report Designs

By Chris Geiss
6/2/02

Table of Contents

About the Author

Chris Geiss is an independent consultant and software developer with 17 years of software development and IT industry experience. He has been working with the Actuate reporting products for over 4 years and specializes in consulting, development, training and mentoring services for the Actuate e.Reporting Suite. Chris can be contacted at chris_geiss@yahoo.com. See his web site at http://www.chrisgeiss.com.

The Mailing List and How to Report Errors and Omissions

If you find any errors or omissions in this document, or have any suggestions, please email them to chris_geiss@yahoo.com so I can update this document accordingly. Also, if you email your contact information to me, I will include you on the mailing list for future updates to this document.

Do You Have Suggestions for a New Article?

If there is a particular Actuate related topic that you would like more information on, please email me and let me know. If the topic seems of interest to the general Actuate developer and user communities, I may base a future article on it.

Sample Report Design for this Article

I created a sample report design that illustrates the naming conventions discussed in this document and which I used for the screen shots. If you would like to receive a copy of this sample report design, please email your contact information to chris_geiss@yahoo.com.

Introduction

The subject of Naming Conventions is one of those topics that can lead to very heated discussions. That is because everyone has an opinion as to what constitutes a good naming convention (from their own point of view), yet, this is a subject with no right or wrong answers. It is simply a matter of preference and style. On the subject of naming conventions, it is more important that you use a standard naming method than it is that you use any one particular method.

In this article I describe the coding standards that I have developed over the years that I have been developing Actuate Basic reports. I encourage you to use these as a starting point. If the conventions I describe meet your needs, then by all means use them as is. However, if you have your own ideas about a good naming convention, then by all means use them to modify what I propose here.

The main objective is to select a convention and stick to it! It will make it much easier for you to maintain your report designs and will make life easier for anyone who maintains them in the future.

"Tone" of the Article

In this article I use phrases like "you should", or "you must." That is because this article was written primarily for use in my own development projects. Therefore, on the projects that I lead, the terms "should" and "must" apply because I am the one dictating the standards. If you are using this article for your own projects, read these phrases as "you could", or "it is suggested that." I am not trying to tell you what to do!

Identifier Names

The term “identifier” refers to any of the names assigned to various program constructs such as components (classes), constants, variables, methods, functions, and subroutines.

Identifiers should be made up of combinations of words that meaningfully describe the purpose and usage of the item. Words in the name should be distinguished by capitalizing the first letter of each word. Standard acronyms should be kept in all capital letters (e.g. IRS, SEC, RAM, etc.). Use underscores if it assists the reader in understanding the name of the item, but use underscores sparingly.

Examples:

CustomerName
DollarAmount
SECFee
WithholdingForIRS
MyCustomFunction()
CG_REQAPI_Interface

“Report Component” vs. “Class”

The term “Report Component” refers to any of the items that you see in the Structure Pane (the left hand portion of the Designer window) or the Library, Project and Globals Browsers in e.Report Designer Professional. Examples include database connections, datasources, frames, controls, etc. In object-oriented terms, these “Report Components” are actually class definitions for the items in your report design. If you examine the Actuate Basic source code that is generated for your report design, you will see that each component results in a CLASS … END CLASS definition. When your report executes, the Actuate factory process uses these class definitions to create the objects which perform the actual report processing. You can examine these objects using the Variables Browser when you are in the debug mode. Many of these report objects are saved in the .roi document that gets generated by the report executable.

In this article I use the terms “report component” and “class” interchangeably.

Report Component (Class) Names

Report component names should be comprised of a type prefix and an identifier. It can be argued that a prefix is not necessary when working in the structure pane of the Developer Workbench, since the icons depict the basic class type. However, the advantage of type prefixes becomes clear when you refer to classes within Actuate Basic code, since the prefix of a variable or object will make it clear to what basic class type you are refering. For example, the code:

```
Dim objMyObject As frmDetailData
```

makes it clear that you are declaring a variable that references a frame object. The code:

```
Dim objMyObject As DetailData
```

makes this less clear. “DetailData” could be a data control. Also, using prefixes on all your components forces you to think about the component names and you are more likely to choose names that would be helpful for someone else who is looking at your report design.

Report Component Type Prefixes

The following report component type prefixes should be used when naming classes or object reference variables to help the developer recognize the basic class type that the class involved derives from.

Databound Controls and Label Components

Prefix	Meaning	Component Derives From
cur	Currency data control	AcCurrencyControl
dbl	Double data control	AcDoubleControl
dt	Datetime data control	AcDatetimeControl
int	Integer data control	AcIntegerControl
lbl	Label control	AcLabelControl
txt	Text data control	AcTextControl

Database Components

Prefix	Meaning	Component Derives From
conn	Database connection	AcDatabaseConnection
odbc	ODBC connection	AcODBCConnection
mssql	Microsoft SQL Server connection	AcMSSQLConnection
db2	DB2 connection	AcDB2Connection
ora	Oracle connection	AcOracleConnection
syb	Sybase connection	AcSybaseConnection
sp	Stored procedure	AcStoredProcedureSource
sql	SQL graphical query (Graphical Query Editor component)	AcSQLQuerySource
sqltxt	SQL text query (Textual Query Editor component)	AcTextQuerySource

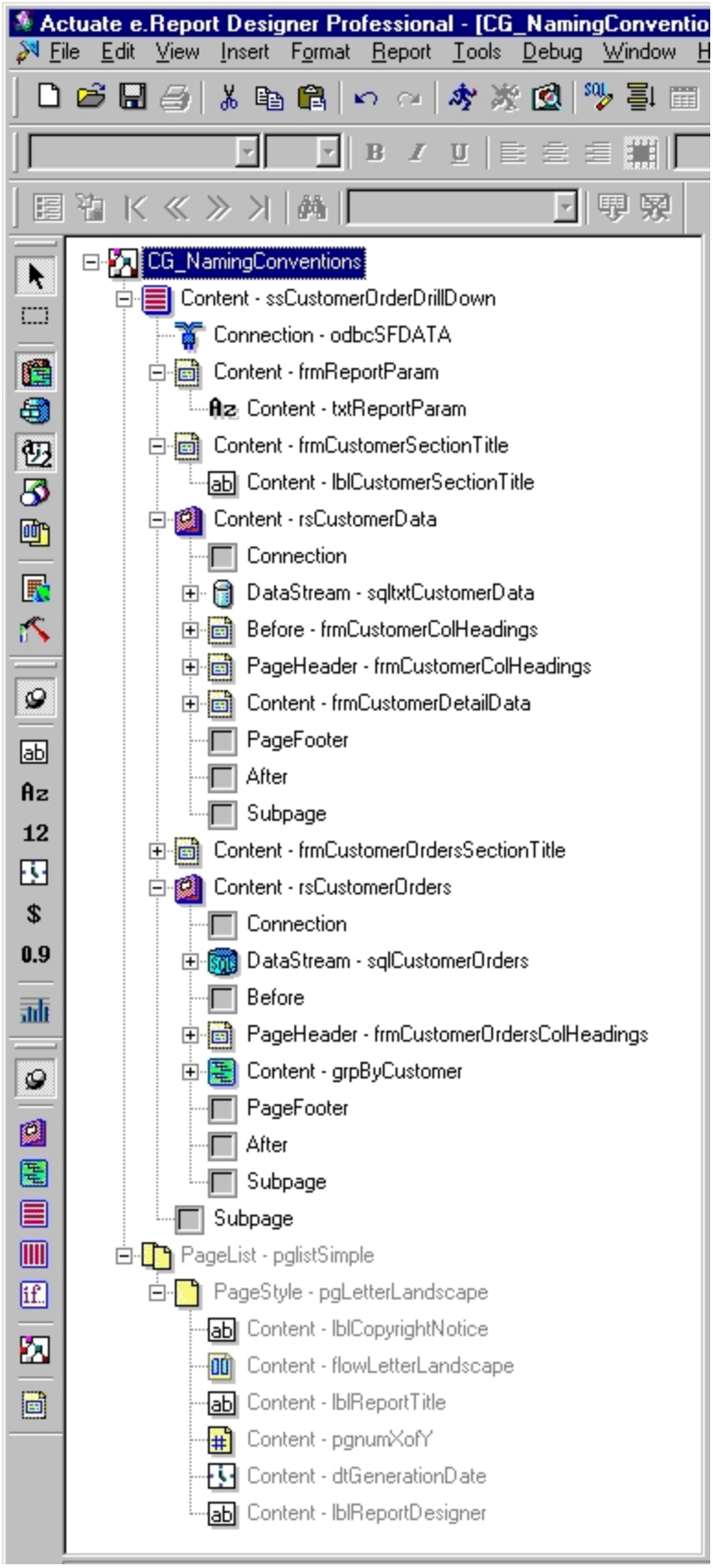
Report Structure Components

Prefix	Meaning	Component Derives From
frm	Frame	AcFrame
grp	Group section	AcGroupSection
if	Conditional section	AcConditionalSection
ps	Parallel section	AcParallelSection
rpt	Report	AcReport
rs	Report section	AcReportSection
ss	Sequential section	AcSequentialSection

Examples:

odbcSFDATA
frmReportParam
lblCustomerSectionTitle
sqltxtCustomerData
grpByCustomer
intOrderID
dtForecastOrderDate

The following screen shot shows the sample report design which illustrates the use of many of the component prefixes described above.



Constant Names

Constants are created using the Const keword. Constant names should follow the rules for naming identifiers, but all letters should be in upper case to help them standout.

Example:

```
Declare

     Const CG_LOGTYPE_MSGBOX = 1
     Const CG_LOGTYPE_FACTORYSTATUS = 2
     Const CG_LOGTYPE_FILE = 3
     Const CG_LOGTYPE_STRING = 4

End Declare
```

Variable Names

Variables names should be comprised of a scope prefix (if applicable), a type prefix and an identifier.

Variable Scope Prefixes

In this context, the term “scope” refers to the parts of the report design to which the variable is visible. If a variable has “global” scope, then it means that the variable is visible to any code in the report design. If a variable has “instance” scope, then it is defined as an instance variable, and is only accessible to the code for the class to which it belongs, or to code which has a reference to an object created from that class.

The following variable scope prefixes should be used to help the developer recognize the scope in which the variable is accessible. This makes it easier to know where to look to find the declaration and documentation for the variable. For example, if a variable has global scope (g_ prefix), then it must either be defined in an Actuate Basic source file or as a report parameter.

Scope Prefix	Meaning
no prefix	Local variables do not have a scope prefix
g_	Global variable; g_ usually means that the global variable was created specifically for the report design in which it is declared, and not for use with other report designs; see XXX_ below
XXX_	A global variable belonging to a reusable code library designated as XXX. For example if there is an ACCT_ accounting function library, then a global variable in that library might be named ACCT_sClientName
i_	Developer defined instance variable. Used when extending a predefined Actuate class, to help distinguish Actuate supplied vs. developer supplied instance variables. Using this prefix is optional if you are creating a class from scratch.
s_	Developer defined static (class) variable. Used when extending a predefined Actuate class, to help distinguish Actuate supplied vs. developer supplied static variables. When used in a class created from scratch, this will help distinguish static variables from instance variables.
ov_	Object variable (used in code in frames to easily reference controls contained within the frame); note, object variables defined in report designs actually result in Actuate generating a class method with that name, so an "object variable" is technically not a "variable"; however, it is used like a variable and so variable naming rules apply
pi_	A Function or Sub parameter that is used to input a value
po_	A Function or Sub parameter that is used to output a value
pio_	A Function or Sub parameter that is used to input and output a value
rp_	Signifies that the variable is defined as a report parameter (using the View | Parameters… menu choice); note, a hidden report parameter which is used exclusively as a global variable should use the g_ prefix; any such global variables should be included under the parameter group "Global Variables"

Variable Type Prefixes

The following variable type prefixes should be used when naming variables to help the developer recognize the data type assigned to the variable.

Prefix	Meaning
s	String
i	Integer
l	Long (note, in Actuate 5.x Integer and Long are the same storage size of 4 bytes)
b	Boolean
dbl	Double
dt	Datetime
cur	Currency
var	Variant
obj	Reference to an object

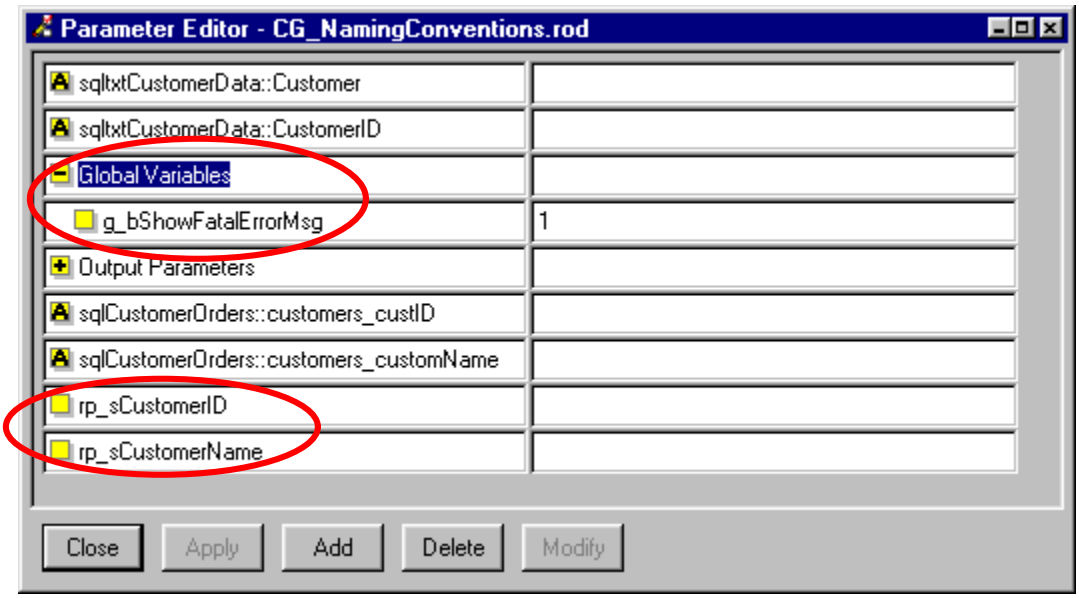
Variable Name Examples

The following table lists examples of variables formed using the prefixes defined above.

Sample Variable Name	Information Obtained from the Prefixes
iEmployeeCount	Local integer variable
g_dblTotalDollars	Global double variable defined for a specific report design (declared in a basic source file associated with just one report design)
IRS_dtDueDate	Global datetime variable defined in the IRS code library
i_sCustPhone	String variable which has been added to an Actuate supplied class as an instance variable

Sample Variable Name	Information Obtained from the Prefixes
s_dtLastPayDate	Datetime variable which has been added to an Actuate supplied class as a static variable
pio_dtOfLastUpdate	A method parameter that is used to input a value as well as provide an output value
rp_dtProcessDate	A datetime report parameter
ov_txtCustomerName	An object variable for referencing the txtCustomerName component
g_objDataFrame	A global variable providing a reference to a frame object

The following screen shot shows report parameters and a global variable that use the naming conventions outlined above. Note, the global variable (a hidden report parameter) is included under the "Global Variables" group. The other parameters are adhoc parameters that have been hidden in this report design.



Here is a code fragment that uses these report parameters:

```
' Setup data sources added to this report design

If rp_sCustomerName <> "" Then
      sqltxtCustomerData::Customer = rp_sCustomerName
      sqlCustomerOrders::customers_customName = rp_sCustomerName
End If

If rp_sCustomerID <> "" Then
      sqltxtCustomerData::CustomerID = rp_sCustomerID
      sqlCustomerOrders::customers_custID = rp_sCustomerID
End If
```

Here is another code fragment that declares a global variable:

```
Declare

      ' Variable to hold and display report parameter values
      Global g_sRptParamText As String

End Declare
```

Method Names

Note, in this context the term "Method" is used as a generic term to refer to both functions and subroutines that are defined for a class. Method names should be comprised of a scope prefix, a type prefix (if applicable) and an identifier. The scope prefix for methods is "m_".

Method Prefixes

The following prefixes should be used when naming methods that are being defined for a class that inherits from a class provided by Actuate. They are optional for methods of classes created from scratch. These prefixes will help the developer recognize custom versus Actuate provided methods.

Prefix	Meaning
m_	A custom method that returns no value (defined using Sub…End Sub)
m_s	A custom method that returns a string value (defined using Function…End Function)
m_i	A custom method that returns an integer value (defined using Function…End Function)
m_*x*	A custom method that returns a different data type than what was mentioned above. See the table in the "Variable Type Prefixes" section for other appropriate prefixes.

Method Name Examples

Sample Method Name	Information Obtained from the Prefixes
m_UpdateAccount()	Custom method that does not return a value
m_iGetEmployeeCount()	Custom method that returns an integer value
m_dblTotalDollars()	Custom method that returns a double value

Here is an example of a custom method for the AcReport component in the sample report design:

```
Sub m_SetDataSourceParams( )

      Super::m_SetDataSourceParams( )

      ' Setup data sources added to this report design

      If rp_sCustomerName <> "" Then
            sqltxtCustomerData::Customer = rp_sCustomerName
            sqlCustomerOrders::customers_customName = rp_sCustomerName
      End If

      If rp_sCustomerID <> "" Then
            sqltxtCustomerData::CustomerID = rp_sCustomerID
            sqlCustomerOrders::customers_custID = rp_sCustomerID
      End If

End Sub
```

Global Function Names

Note, in this context the term "Function" is used as a generic term to refer to global functions and subroutines. Global Function names should be comprised of a scope prefix, a type prefix and an identifier. Global Subroutine names should be comprised of a scope prefix and an identifier.

Global Function Prefixes

The following scope/return type prefixes should be used when creating global functions. These prefixes will help the developer recognize where the function is defined. These prefixes apply to Sub/End Sub as well as Function/End Function type constructs.

Prefix	Meaning
g_	Global subroutine
XXX_	A global subroutine belonging to a code library designated as XXX. For example, if there is an ACCT_ accounting library, then a global subroutine in that library might be named ACCT_SetClientName()
g_*x*	Global function that returns a value of type x. See the table in the "Variable Type Prefixes" section for the appropriate type prefixes.
XXX_*x*	A function belonging to a code library designated as XXX, that returns a value of type x. See the table in the "Variable Type Prefixes" section for the appropriate type prefixes.

Global Function Name Examples

Sample Function Name	Information Obtained from the Prefixes
g_UpdateAccount()	Global subroutine that does not return a value
g_iGetEmployeeCount()	Global function that returns an integer value
g_dblTotalDollars()	Global function that returns a double value
ACCT_DisplayStatus()	Global subroutine (does not return a value) from the ACCT code library.
IRS_sGetEIN()	Global function from the IRS code library that returns a string value

Here is an example of a global function designed specifically for the sample report design:

```
Function g_AddRptParamText(sText As String) As String

      ' Add the specified text to the existing report parameter text

      If Trim$(g_sRptParamText) <> "" Then
            g_sRptParamText = g_sRptParamText & ", "
      End If

      g_sRptParamText = g_sRptParamText & sText

End Function
```