DecisionPro Code Review

Reviewer: David A.

## Scope

The scope of this code review is to do an overall assessment of the current code, based on the *RefactorObjects* branch in source control. I will cover the most important aspects I see that need to be improved, but not the entire program. I will try to review this in such a way that you can take the feedback and apply it to the rest of the program, and future programs you write.

##### Variable names inside methods

Variable names inside a method should always use camel casing (ex. string *myVariableName = “”;*). Pascal casing for class members, fields, or method variables is invalid usage (ex. String MyVariableName = “”;).

##### Formatting dates and times

Program.cs - Lines 26 to 27

When formatting dates and times, you can actually do this through the *ToString* method, by specifying a format string. This functionality is documented on msdn (<http://msdn.microsoft.com/en-us/library/8kb3ddd4(v=vs.100).aspx)>. You can also do this with *TimeSpan* structures.

Example:

DateTime now = DateTime.Now;  
string formattedTime = now.ToString(“MM/dd/yyyy H:mm:ss”) // 06/10/11 15:24:16

This is the cleanest way to format dates and times, and is the best practice. It uses the proper culture information from the platform the application is running on and always ensures accuracy. An example of combining this into a file name would be:

DateTime startTime = DateTime.Now;  
string filename = String.Format(“ApplicationProcessorLog\_{0}.log”,  
 startTime.ToString(“MM/dd/yyyy H:mm:ss”));  
  
You can also just do this with one less line of code:

string filename = String.Format(“ApplicationProcessorLog\_{0}.log”,  
 DateTime.Now.ToString(“MM/dd/yyyy H:mm:ss”));

Regions  
The usage of regions is an indication that the source code needs to be broken up into more granular or logical classes and methods. In addition, if utilized, region names should use proper English and grammar structure for readability, since the feature is meant to be used as such.

Example:

#region Final Processes

…

#endregion

String Concatenation and Formatting  
Any source code.

Always use string formatting (eg. String.Format()) over basic string concatenation (eg. “stringA” + “stringB”). Using string formatting enhances readability and clarity, and ensures that the proper culture and globalization information from the underlying platform is used.

Comparing Strings  
Any code.

When comparing strings, avoid use String.ToLower or String.ToUpper for comparison. Instead, use String.Equals with the StringComparison Enumeration which allows you to do case sensitive or case insensitive string comparison, while considering or not considering culture information, or doing a fast ordinal comparison instead (for performance).

Example:

string x = “a”;  
string y = “b”;  
  
if (x.Equals(y, StringComparison.OrdinalIgnoreCase))  
{  
}

In this case, it does not matter whether x or y are upper or lower case, and we are using ordinal string comparison for fast performance.

Initializing Classes with Property Values  
You can use object initializers to write cleaner initialization code when instantiating a class. The first example will show the current method being utilized, and the second example will show how to use object initializers to accomplish the same task with less code and overhead.

Example 1 (Current method):

Process importer = new Process();  
importer.StartInfo.FileName = Path.Combine(Configuration.ImporterPath, "accuaccount.importer.exe");  
importer.StartInfo.WorkingDirectory = Configuration.ImporterPath;  
importer.Start();  
importer.WaitForExit();  
LogWriter.LogMessage("Importer Complete");

Example 2 (Object Initializer, Preferred method):

Process importer = new Process  
{  
 StartInfo =  
 {  
 FileName = Path.Combine(Configuration.ImporterPath, "accuaccount.importer.exe"),  
 WorkingDirectory = Configuration.ImporterPath  
 }  
};

importer.Start();  
importer.WaitForExit();

Object initializers cannot always be used, but the majority of the time they can. Object initializers can primarily be used when you are instantiating a class and setting properties. You cannot use object initializers to call methods.