Coding Standards

Naming Conventions

- Do not use Hungarian notation
- Do not use a prefix for member variables (_, m_, s_, etc.). If you want to distinguish between local and member variables you should use "this." in C# and "Me." in VB.NET.
- Do use camelCasing for member variables
- Do use camelCasing for parameters
- Do use camelCasing for local variables
- Do use PascalCasing for function, property, event, and class names
- Do prefix interfaces names with "I"
- Do not prefix enums, classes, or delegates with any letter

Spacing

- Do use a single space after a comma between function arguments.
  Right: Console.In.Read(myChar, 0, 1);
  Wrong: Console.In.Read(myChar,0,1);
- Do not use a space after the parenthesis and function arguments
  Right: CreateFoo(myChar, 0, 1)
  Wrong: CreateFoo(myChar, 0, 1)
- Do not use spaces between a function name and parenthesis.
  Right: CreateFoo()
  Wrong: CreateFoo ()
- Do not use spaces inside brackets.
  Right: x = dataArray[index];
  Wrong: x = dataArray[index];
- Do use a single space before flow control statements
  Right: while (x == y)
  Wrong: while(x==y)
- Do use a single space before and after comparison operators
  Right: if (x == y)
  Wrong: if (x==y)

Layout Conventions

Good layout uses formatting to emphasize the structure of your code and to make the code easier to read. Use the default Code Editor settings (smart indenting, four-character indents, tabs saved as spaces).

- Write only one statement per line.
- Write only one declaration per line.
- If continuation lines are not indented automatically, indent them one tab stop (four spaces).
- Add at least one blank line between method definitions and property definitions.

- Use parentheses to make clauses in an expression apparent, as shown in the following code.

Comment Conventions

- The // (two slashes) style of comment tags should be used in most situations. Where ever possible, place comments above the code instead of beside it
- All methods should use XML doc comments. For internal dev comments, the <devdoc> tag should be used

Class and Method Design

- Source files should contain only one public type, although multiple internal classes are allowed
- Source files should be given the name of the public class in the file
- Create one class for one purpose only

Error and Exception Handling

- Use a try-catch statement for most exception handling
- Throw exceptions rather than returning some kind of status value

Maintainability

- Use variables for constants and numbers wherever possible
- Make all members private and types internal by default
- Initialize variables at the point of declaration
- Don’t change a loop variable inside a for or foreach loop

Framework Guidelines

- Use C# type aliases instead of the types from the System namespace. For instance, use object instead of Object, string instead of String, and int instead of Int32
- Don’t hardcode strings that change based on the deployment