Beautifier?	Resharper rule	Resharper rule description	Similar rule in FxCop
Yes	Expression ? can be rewritten as ??		
	Empty general catch clause	A catch clause that catches System.Exception and has an empty body.	`DoNotCatchGeneralExceptionTypes` *Resharper has more narrow definition of the rule*
	Exception rethrow possibly intended	A 'throw' statement inside a catch clause which throws the exception caught. In most of cases a 'throw' statement with no argument is to be used.	`DoNotRaiseExceptionsInExceptionClauses` `RethrowToPreserveStackDetails`
	Function never returns	Function does not reach its end or a 'return' statement by any of possible execution paths.	
	Local variable hides member	Local variable has the same name as a field and hides it.	
	'Object.ReferenceEquals' is always false because it is called with value type	'Object.ReferenceEquals' is always false because it is called with value type	
	Parameter hides member	Method parameter has the same name as a field and hides it.	`ParameterNamesShouldNotMatchMemberNames`
	Possible compare of value type with 'null'	Generic type has no value or class constraint, the condition could be always 'false'	
	Problems in format string	Detects missing parameters in parameter list, invalid specifiers in format string etc.	
	'value' parameter is not used	The setter of a property or indexer does not use its 'value' parameter. Also applies to adders and removers of events.	

Beautifier?	Resharper rule	Resharper rule description	Similar rule in FxCop
	Virtual member call in constructor	When a virtual method is called, the actual type that executes the method is not selected until run time. When a constructor calls a virtual method, it is possible that the constructor for the instance that invokes the method has not executed.	`DoNotCallOverridableMethodsInConstructors`
Yes	Anonymous method signature is not necessary	Specifying signature in an anonymous method is not necessary because none of its parameters are used in the body.	
	Assignment is not used	Value assigned to a local variable or parameter is not used in any execution path.	`C# compiler warning` `warning CS0219: The variable 'XXX' is assigned but its value is never used`
Yes	Empty constructor	Empty public constructor declaration with no parameters is redundant. The compiler generates the same by default.	
Yes	Empty namespace declaration	Empty namespace declaration is redundant.	`AvoidNamespacesWithFewTypes`
Yes	Explicit delegate creation expression is redundant		
	Field can be made readonly		
Yes	Iteration variable can be declared with a more specific type	Type of iteration variable declared in 'foreach' statement is less specific than that which can be inferred from the collection type being iterated	
Yes	Local variable has too wide declaration scope	Local variable is declared in a wider scope than the scope of its actual use.	

Beautifier?	Resharper rule	Resharper rule description	Similar rule in FxCop
	Member can be made static	A non-virtual instance member does not use 'this' object (neither implicitly nor explicitly) and can be made static.	**MarkMembersAsStatic**
	Method return value is never used	For private members only	**DoNotIgnoreMethodResults**
Yes	Parameter can be declared with base type		**ConsiderPassingBaseTypesAsParameters**
Yes	Parentheses are redundant if attribute has no arguments	Parentheses are redundant if attribute has no arguments	
Yes	Redundant base constructor call	Explicit call to the base class constructor with no arguments. Is generated by the compiler by default and can be omitted.	
Yes	Redundant 'base.' Qualifier	'base.' qualifier is redundant and can be safely removed without changing code semantics.	
Yes	Redundant boolean comparison	Comparison of a boolean value with 'true' or 'false' constant.	
	Redundant cast	Type cast can be safely removed.	**DoNotCastUnnecessarily**
Yes	Redundant catch clause	Catch clause with single 'throw' statement is redundant.	
Yes	Redundant empty finally block	Empty 'finally' block is redundant.	
	Redundant extends list entry		
Yes	Redundant member override	The override of a virtual member is redundant because it consists of only a call to the base.	
Yes	Redundant name qualifier	Redundant use of qualifier for a type name or static member usage	
Yes	Redundant 'object.ToString()' call	Use of ToString() call in a context	

Beautifier?	Resharper rule	Resharper rule description	Similar rule in FxCop
		where it would be generated by the compiler automatically. For example, in a concatenation with a string or as an argument of a string.Format() call.	
Yes	Redundant 'partial' modifier	Class is declared as 'partial', but has only single part	
	Redundant 'string.ToCahrArray()' call		
Yes	Redundant 'this.' Qualifier	'this.' qualifier is redundant and can be safely removed without changing code semantics.	
	Redundant type arguments of method	Specification of method type arguments is redundant because they are inferred from argument types.	
	Redundant using directive	Using directive is not required by the code and can be safely removed.	
Yes	Sealed member in sealed class	'sealed' modifier for member in a sealed class is redundant.	
Yes	'true' is redundant as 'for'-statement condition	'true' is redundant as 'for'-statement condition, and thus could safely be omitted	
Yes	Underlying type of enum is 'int'	'int' is default underlying type of enum, so it is not necessary to specify it explicitly	**EnumStorageShouldBeInt32**
	Unsafe context declaration is redundant	Unsafe context declaration is redundant because it is declared in unsafe context, or it doesn't contain unsafe constructs	
	Unused member in private type	Non-private member in a private	

Beautifier?	Resharper rule	Resharper rule description	Similar rule in FxCop
		nested type is never used.	
	Unused private member	Private member is never used.	**AvoidUnusedPrivateFields** **AvoidUncalledPrivateCode** **AvoidUninstantiatedInternalClasses**
	Unused type parameter	Type parameter is never used.	
	Expression is always 'true' or always 'false'	Value of a boolean expression is always the same at this point.	
	Possible 'null' assignment to entity marked with 'Value cannot be null' attribute	An expression which can have 'null' value is assigned to an entity marked with 'Value cannot be null' attribute. In particular, this can happen when passing such value to a method whose parameter is marked with 'Value cannot be null' attribute.	
	Possible 'System.NullReferenceException'	Dereferencing an expression which can have 'null' value. This warning is detected either when there is a comparison with 'null' earlier in the code or when this value is returned by a member marked with 'Value can be null' attribute.	
	Namespace does not correspond to file location	Namespace in file does not have a form of project Default Namespace plus folder names in the path to file. You can configure which folders participate in namespace building process on the folder's properties page	