# C# Coding Standards and Naming Conventions

Below are our **C# coding standards**, naming conventions, and best practices.
Use these in your own projects and/or adjust these to your own needs.

---

use **PascalCasing** for class names and method names.

```
1. publicclassClientActivity
2. publicClearStatistics
3. //...
4. publicCalculateStatistics
5. //...
```

**Why**: consistent with the Microsoft's .NET Framework and easy to read.

---

use **camelCasing** for method arguments and local variables.

```
1. publicclassUserLog
2. publicLogEvent logEvent
3.  itemCount  logEventItemsCount
4. // ...
```

**Why**: consistent with the Microsoft's .NET Framework and easy to read.

---

do notuse **Hungarian** notation or any other type identification in identifiers

```
1. // Correct
2.  counter
3. string
4. // Avoid
```

```
5.  iCounter
6. string strName
```

**Why**: consistent with the Microsoft's .NET Framework and Visual Studio IDE makes determining types very easy (via tooltips). In general you want to avoid type indicators in any identifier.

---

do notuse **Screaming Caps** for constants or readonly variables

```
1. // Correct
2. publicstaticconststringShippingType"DropShip"
3. // Avoid
4. publicstaticconststring SHIPPINGTYPE "DropShip"
```

**Why**: consistent with the Microsoft's .NET Framework. Caps grap too much attention.

---

avoidusing **Abbreviations**. Exceptions: abbreviations commonly used as names, such as **Id, Xml, Ftp, Uri**

```
1. // Correct
2. UserGroup userGroup
3. Assignment employeeAssignment
4. // Avoid
5. UserGroup usrGrp
6. Assignment empAssignment
7. // Exceptions
8. CustomerId customerId
9. XmlDocument xmlDocument
10. FtpHelper ftpHelper
11. UriPart uriPart
```

**Why**: consistent with the Microsoft's .NET Framework and prevents inconsistent abbreviations.

---

use **PascalCasing** for abbreviations 3 characters or more (2 chars are both uppercase)

```
1. HtmlHelper htmlHelper
2. FtpTransfer ftpTransfer
3. UIControl uiControl
```

**Why**: consistent with the Microsoft's .NET Framework. Caps would grap visually too much attention.

---

do notuse **Underscores** in identifiers. Exception: you can prefix private static variables with an underscore.

```
1. // Correct
2. publicDateTime clientAppointment
3. publicTimeSpan timeLeft
4. // Avoid
5. publicDateTime client_Appointment
6. publicTimeSpan time_Left
7. // Exception
8. privateDateTime _registrationDate
```

**Why**: consistent with the Microsoft's .NET Framework and makes code more natural to read (without 'slur'). Also avoids underline stress (inability to see underline).

---

use **predefined type names** instead of system type names like Int16, Single, UInt64, etc

```
1. // Correct
2. string firstName
3.  lastIndex
4.  isSaved
5. // Avoid
6. String firstName
7. Int32 lastIndex
8. Boolean isSaved
```

**Why**: consistent with the Microsoft's .NET Framework and makes code more natural to read.

---

use implicit type **var** for local variable declarations. Exception: primitive types (int, string, double, etc) use predefined names.

```
1.  stream Create
2.  customers Dictionary
3. // Exceptions
4.  index
5. string timeSheet
6.  isCompleted
```

**Why**: removes clutter, particularly with complex generic types. Type is easily detected with Visual Studio tooltips.

---

use noun or noun phrases to name a class.

```
1. publicclassEmployee
2. publicclassBusinessLocation
3. publicclassDocumentCollection
```

**Why**: consistent with the Microsoft's .NET Framework and easy to remember.

---

prefix interfaces with the letter .  Interface names are noun (phrases) or adjectives.

```
1. publicinterfaceIShape
2. publicinterfaceIShapeCollection
3. publicinterfaceIGroupable
```

**Why**: consistent with the Microsoft's .NET Framework.

---

name source files according to their main classes. Exception: file names with partial classes reflect their source or purpose, e.g. designer, generated, etc.

```
1. // Located in Task.cs
```

```
2. publicpartialclass
3. //...
```

```
1. // Located in Task.generated.cs
2. publicpartialclass
3. //...
```

**Why**: consistent with the Microsoft practices. Files are alphabetically sorted and partial classes remain adjacent.

---

organize namespaces with a clearly defined structure

```
1. // Examples
2. namespaceCompanyProductModuleSubModule
3. namespaceProductModuleComponent
4. namespaceProductLayerModuleGroup
```

**Why**: consistent with the Microsoft's .NET Framework. Maintains good organization of your code base.

---

vertically align curly brackets.

```
1. // Correct
2. classProgram
3. staticstring
```

**Why**: Microsoft has a different standard, but developers have overwhelmingly preferred vertically aligned brackets.

---

declare all member variables at the top of a class, with static variables at the very top.

```
1. // Correct
2. publicclassAccount
3. publicstaticstringBankName
4. publicstaticdecimalReserves
```

```
5. publicstringNumber
6. publicDateTimeDateOpened
7. publicDateTimeDateClosed
8. publicdecimalBalance
9. // Constructor
10. publicAccount
11. // ...
```

**Why**: generally accepted practice that prevents the need to hunt for variable declarations.

---

use singular names for enums. Exception: bit field enums.

```
1. // Correct
2. publicColor
3. Green
4. Yellow
5. Magenta
6. // Exception
7. Flags
8. publicDockings
9. Right
10. Bottom
```

**Why**: consistent with the Microsoft's .NET Framework and makes the code more natural to read. Plural flags because enum can hold multiple values (using bitwise 'OR').

---

do notexplicitly specify a type of an enum or values of enums (except bit fields)

```
1. // Don't
2. publicDirection
3. North
4. South
5. // Correct
6. publicDirection
7. North
8. South
```

**Why**: can create confusion when relying on actual types and values.

---

do notsuffix enum names with Enum

```
1. // Don't
2. publicCoinEnum
3. Penny
4. Nickel
5. Quarter
6. Dollar
7. // Correct
8. public
9. Penny
10. Nickel
11. Quarter
12. Dollar
```

**Why**: consistent with the Microsoft's .NET Framework and consistent with prior rule of no type indicators in identifiers.

---

Links