**C# Coding Guidelines**

**1. General Principles**

* Write clear, readable, and maintainable code.
* Prefer clarity over cleverness.
* Use consistent naming and formatting throughout the codebase.
* Keep methods and classes focused on a single responsibility.

**2. Naming and Structure**

* **Namespaces**: Use meaningful, hierarchical namespaces (e.g., Company.Project.Module). Avoid generic names like Utils or Helpers.
* **Classes/Interfaces**: Name with nouns or noun phrases. Prefix interfaces with I (e.g., IProductExporter).
* **Methods**: Use verbs or verb phrases (e.g., ExportProducts, ParseResponse).
* **Variables/Parameters**: Use camelCase for locals and parameters. Use PascalCase for properties and fields.

**3. Formatting and Layout**

* Indentation: 4 spaces per level. Never use tabs.
* Braces: Place opening braces on a new line for types and members.
* Line Length: Limit lines to 120 characters.
* Use blank lines to separate logical code blocks.

**4. Code Style**

* Use var only when the type is obvious (e.g., var customer = new Customer();).
* Use explicit types when not obvious (e.g., int count = 0;).
* Prefer object and collection initializers for clarity.
* Avoid Hungarian notation (no strName, intCount, etc.).

**5. Error Handling**

* Throw the most specific exception available (e.g., ArgumentNullException).
* Use TryParse for conversions that might fail, rather than exceptions.
* Never swallow exceptions; always log or handle them meaningfully.

**6. Method Design**

* Each method should do one thing (Single Responsibility).
* Keep methods short (ideally ≤15 lines).
* Avoid more than 3 parameters; use objects for grouping if needed.
* Use optional parameters to reduce overloads, but not in interfaces.
* Avoid boolean flags in method signatures.

**7. Immutability and State**

* Prefer readonly and immutable types where possible.
* Properties should not depend on the order of setting other properties.

**8. Async and Await**

* Use async/await for asynchronous operations.
* Avoid .Result and .Wait() on tasks.
* Suffix async methods with Async (e.g., ExportAsync).

**9. Documentation**

* Use XML comments for all public classes, methods, and properties.
* Use <summary>, <param>, and <returns> tags for clarity.
* Add comments to explain complex logic, but avoid redundant comments.

**10. Testing and Mocking**

* Use interfaces to decouple logic and enable mocking.
* Use dependency injection for substituting real and mock implementations.
* Keep mock data and logic separate from core business logic.

**11. Logging**

* Centralize logging in a dedicated class.
* Include timestamps and severity in log entries.
* Log both actions and errors.

**12. Formatting and Style**

* Use consistent indentation (4 spaces).
* Place braces on a new line for classes, methods, and properties.
* Use object and collection initializers where appropriate.
* Avoid field prefixes like \_, m\_, or g\_.

**13. Configuration Management**

* Store file paths and settings in a configuration file or as constants.
* Do not hard-code deployment-specific strings.

**14. Other Best Practices**

* Use named constants or enums instead of magic numbers.
* Prefer LINQ for collection manipulation, but avoid overly complex queries.
* Implement IDisposable where needed and use using statements.

**15. Sample Good Practices**

**Namespace Example:**

C#

namespace AdmLodPrototype.Export

{

// ...

}

Show more lines

**Var Usage:**

C#

var customer = new Customer(); // Type is evident

Show more lines

**TryParse Pattern:**

C#

bool success = int.TryParse(text, out int number);

Show more lines

**Specific Exception:**

C#

public void ProcessData(string data)

{

if (data == null)

throw new ArgumentNullException(nameof(data), "Input data cannot be null.");

// ...

}

Show more lines

**Interface Decoupling:**

C#

public interface IDataProvider

{

IEnumerable<Product> GetProducts();

}

Show more lines

**References**

* https://github.com/dotnet/csharpstandard
* https://context7.com/dennisdoomen/csharpguidelines
* https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions