

Namespace Bodu

Classes

[ThrowHelper](#)

Provides centralized guard clause methods for argument validation using resource-based exception messages.

[XorShiftRandom](#)

Represents a high-performance, non-cryptographic pseudo-random number generator based on the XOR-shift algorithm.

Interfaces

[IRandomGenerator](#)

Defines a pluggable interface for random number generation.

Interface IRandomGenerator

Namespace: [Bodu](#)

Assembly: Bodu.CoreLib.dll

Defines a pluggable interface for random number generation.

```
public interface IRandomGenerator
```

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#)

Methods

Next(int)

Returns a non-negative random integer that is less than the specified maximum.

```
int Next(int maxValue)
```

Parameters

maxValue [int](#)

The exclusive upper bound.

Returns

[int](#)

A random integer in the range [0, maxValue].

Class ThrowHelper

Namespace: [Bodu](#)

Assembly: Bodu.CoreLib.dll

Provides centralized guard clause methods for argument validation using resource-based exception messages.

```
public static class ThrowHelper
```

Inheritance

[object](#) ← ThrowHelper

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

ThrowIfArrayContainsNonNumeric(Array, string?)

Throws an [ArgumentException](#) if the array contains any non-numeric items.

```
public static void ThrowIfArrayContainsNonNumeric(Array array, string? paramName = null)
```

Parameters

array [Array](#)

The array to validate.

paramName [string](#)

The name of the parameter.

Remarks

Validates that each element is of a numeric type (e.g., int, double, float, etc.).

Exceptions

[ArgumentException](#)

Thrown when any element in the array is not a numeric type. Message: "The array contains non-numeric values."

ThrowIfArrayIsNotSingleDimension(Array, string?)

Throws an [ArgumentException](#) if the specified array is not single-dimensional.

```
public static void ThrowIfArrayIsNotSingleDimension(Array array, string? paramName = null)
```

Parameters

array [Array](#)

The array to validate.

paramName [string](#)

The name of the parameter, automatically inferred.

Remarks

Checks that `array.Rank == 1`.

Exceptions

[ArgumentException](#)

Thrown when `array` has a rank other than 1. Message: "Only single dimension arrays are supported here."

ThrowIfArrayIsNotZeroBased(Array, string?)

Throws an [ArgumentException](#) if the array does not have a zero lower bound.

```
public static void ThrowIfArrayIsNotZeroBased(Array array, string? paramName = null)
```

Parameters

array [Array](#)

The array to validate.

paramName [string](#)

The name of the array parameter.

Remarks

Checks that the array is zero-based (i.e., index starts at 0).

Exceptions

[ArgumentException](#)

Thrown when `array.GetLowerBound(0) != 0`. Message: "The lower bound of target array must be zero."

ThrowIfArrayLengthIsInsufficient(Array, int, int, string?)

Throws an [ArgumentException](#) if the array is too short for the given index and required length.

```
public static void ThrowIfArrayLengthIsInsufficient(Array array, int index, int requiredLength, string? paramName = null)
```

Parameters

array [Array](#)

The array to check.

index [int](#)

The starting index.

requiredLength [int](#)

The required length from index onward.

paramName [string](#)

The name of the array parameter.

Exceptions

[ArgumentException](#)

Thrown when `array.Length - index < requiredLength`. Message: "Array is too short. Required minimum is {0} from a specified index."

ThrowIfArrayLengthIsZero(Array, string?)

Throws an [ArgumentException](#) if the array has zero length.

```
public static void ThrowIfArrayLengthIsZero(Array array, string? paramName = null)
```

Parameters

`array` [Array](#)

The array to check.

`paramName` [string](#)

The name of the array parameter.

Exceptions

[ArgumentException](#)

Thrown when `array.Length == 0`. Message: "The provided array has zero length. Length must be greater than zero."

ThrowIfArrayLengthNotPositiveMultipleOf(Array, int, string?)

Throws an [ArgumentException](#) if the array length is not a positive multiple of a given divisor.

```
public static void ThrowIfArrayLengthNotPositiveMultipleOf(Array array, int divisor, string? paramName = null)
```

Parameters

array [Array](#)

The array to check.

divisor [int](#)

The required positive divisor.

paramName [string](#)

The name of the array parameter.

Exceptions

[ArgumentException](#)

Thrown when the array length is 0 or not divisible by **divisor**. Message: "Length of the array must be a multiple of {0}."

ThrowIfArrayOffsetOrCountInvalid(Array, int, int, string?, string?, string?)

Throws an exception if the specified **index** and **count** define a segment that exceeds the bounds of the **array**.

```
public static void ThrowIfArrayOffsetOrCountInvalid(Array array, int index, int count,
    string? paramArrayName = null, string? paramIndexName = null, string? paramCountName = null)
```

Parameters

array [Array](#)

The array to validate.

index [int](#)

The zero-based starting index within the array.

count [int](#)

The number of elements to access from **index**.

[paramArrayName](#) [string](#)

The name of the array parameter to include in exception messages.

[paramIndexName](#) [string](#)

The name of the index parameter to include in exception messages.

[paramCountName](#) [string](#)

The name of the count parameter to include in exception messages.

Remarks

This method validates that the segment of the array specified by `index` and `count` is within valid bounds. It ensures that no out-of-range access occurs when operating on a subrange.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown when `index` is negative or greater than `array.Length`, or when `count` is negative or greater than `array.Length`. Message: "The value must zero or positive, and less than the number of elements in the array."

[ArgumentException](#)

Thrown when the sum of `index` and `count` exceeds `array.Length`. Message: "The sum of index and count exceeds the number of elements in the array."

ThrowIfArrayTypeIsNotCompatible<TExpected>(Array, string?)

Throws an [ArgumentException](#) if the array is not assignable to the specified type `TExpected`.

```
public static void ThrowIfArrayTypeIsNotCompatible<TExpected>(Array array, string? paramName = null)
```

Parameters

[array](#) [Array](#)

The array to check.

`paramName` [string](#)

The parameter name of the array.

Type Parameters

`TExpected`

The expected array element type.

Remarks

Validates the runtime type of the array using pattern matching and avoids invalid casts.

Exceptions

[ArgumentException](#)

Thrown when the array is not of type `TExpected`[]. Message: "Target array type is not compatible with the type of items in the collection."

ThrowIfCollectionTooSmall<T>(ICollection<T>, int, string?)

Throws an [ArgumentException](#) if the collection has fewer elements than the specified minimum.

```
public static void ThrowIfCollectionTooSmall<T>(ICollection<T> collection, int minCount,  
string? paramName = null)
```

Parameters

`collection` [ICollection](#)<T>

The collection to validate.

`minCount` [int](#)

The minimum number of required elements.

`paramName` [string](#)

The name of the parameter.

Type Parameters

T

The element type.

Remarks

Performs a minimum count check on the collection.

Exceptions

[ArgumentException](#)

Thrown if `collection` has fewer than `minCount` elements. Message: "Collection contains insufficient elements."

ThrowIfCountExceedsAvailable(int, int, string?)

Throws an [ArgumentOutOfRangeException](#) if the specified `count` is less than zero or greater than the number of `available` items.

```
public static void ThrowIfCountExceedsAvailable(int count, int available, string? paramName  
= null)
```

Parameters

`count` [int](#)

The count value to validate.

`available` [int](#)

The number of available items.

`paramName` [string](#)

The name of the parameter being validated (usually "count").

Remarks

Use this method when validating that a subset operation will not exceed the size of the source.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown when `count` is negative or greater than `available`. Message: "Count must be non-negative and not exceed the number of available items."

ThrowIfEnumValueIsUndefined<TEnum>(TEnum, string?)

Throws an [ArgumentOutOfRangeException](#) if the provided enum value is not defined for `TEnum`.

```
public static void ThrowIfEnumValueIsUndefined<TEnum>(TEnum value, string? paramName = null)  
where TEnum : struct, Enum
```

Parameters

`value` TEnum

The value to validate.

`paramName` [string](#)

The parameter name, inferred from the caller.

Type Parameters

`TEnum`

The enum type.

Remarks

Uses [IsDefined\(Type, object\)](#) to check that the enum value is valid.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown when `value` is not a defined enum member. Message: "The value is not a valid member of the enumeration {0}."

ThrowIfGreaterThanOrEqualOther<T>(T, T, string?, string?)

Throws an [ArgumentException](#) if the value is greater than or equal to another parameter's value.

```
public static void ThrowIfGreaterThanOrEqualOther<T>(T value, T other, string? paramName = null, string? otherName = null) where T : IComparable<T>
```

Parameters

value **T**

The value being validated.

other **T**

The comparison reference value.

paramName [string](#)

Name of the value parameter.

otherName [string](#)

Name of the comparison parameter.

Type Parameters

T

A comparable type.

Exceptions

[ArgumentException](#)

Thrown if **value** >= **other**. Message: "The value must not be greater than or equal to the value of {0}."

ThrowIfGreaterThanOrEqual<T>(T, T, string?)

Throws an [ArgumentOutOfRangeException](#) if the value is greater than or equal to the specified maximum.

```
public static void ThrowIfGreaterThanOrEqual<T>(T value, T max, string? paramName = null)
where T : IComparable<T>
```

Parameters

value **T**

The value to compare.

max **T**

The exclusive upper bound.

paramName [string](#)

The parameter name of the value.

Type Parameters

T

A comparable type.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if **value** >= **max**. Message: "The value must be less than {0}."

ThrowIfGreaterThanOther<T>(T, T, string?, string?)

Throws an [ArgumentException](#) if the value is greater than another parameter's value.

```
public static void ThrowIfGreaterThanOther<T>(T value, T other, string? paramName = null,
string? otherName = null) where T : IComparable<T>
```

Parameters

value **T**

The current value being validated.

other T

The comparison reference value.

paramName [string](#)

Name of the value parameter.

otherName [string](#)

Name of the comparison parameter.

Type Parameters

T

A comparable type.

Exceptions

[ArgumentException](#)

Thrown if **value** > **other**. Message: "The value must not be greater than the value of {0}."

ThrowIfGreaterThan<T>(T, T, string?)

Throws an [ArgumentOutOfRangeException](#) if the value is greater than the specified maximum.

```
public static void ThrowIfGreaterThan<T>(T value, T max, string? paramName = null) where T : IComparable<T>
```

Parameters

value T

The value to compare.

max T

The upper bound (inclusive).

paramName [string](#)

The parameter name of the value.

Type Parameters

T

A comparable type.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `value > max`. Message: "The value must be less than or equal to {0}."

ThrowIfIndexOutOfRange<T>(int, int, string?)

Throws an [ArgumentOutOfRangeException](#) if the index is outside the valid range of a collection.

```
public static void ThrowIfIndexOutOfRange<T>(int index, int size, string? paramName = null)  
where T : IComparable<T>
```

Parameters

`index` [int](#)

The index to validate.

`size` [int](#)

The valid size of the collection.

`paramName` [string](#)

The parameter name for the index.

Type Parameters

T

Type of items in the collection (used for annotation).

Exceptions

[ArgumentException](#)

Thrown if `index` is not in [0, `size`). Message: "The index must be non-negative and less than the size of {0}."

ThrowIfInvalidStringComparison(StringComparison, string?)

Throws an [ArgumentException](#) if the string comparison option is invalid or unsupported.

```
public static void ThrowIfInvalidStringComparison(StringComparison comparison, string?  
paramName = null)
```

Parameters

`comparison` [StringComparison](#)

The [StringComparison](#) value to validate.

`paramName` [string](#)

The name of the parameter.

Remarks

Useful for guarding API input that relies on specific string comparison modes.

Exceptions

[ArgumentException](#)

Thrown if `comparison` is not a valid [StringComparison](#) enum value. Message: "The string comparison type is not supported."

ThrowIfLessThanOrEqualOther<T>(T, T, string?, string?)

Throws an [ArgumentException](#) if the value is less than or equal to another parameter's value.

```
public static void ThrowIfLessThanOrEqualOther<T>(T value, T other, string? paramName =  
null, string? otherName = null) where T : IComparable<T>
```

Parameters

value T

The value being validated.

other T

The comparison reference value.

paramName [string](#)

Name of the value parameter.

otherName [string](#)

Name of the comparison parameter.

Type Parameters

T

A comparable type.

Exceptions

[ArgumentException](#)

Thrown if **value** >= **other**. Message: "The value must not be less than or equal to the value of {0}."

ThrowIfLessThanOrEqual<T>(T, T, string?)

Throws an [ArgumentOutOfRangeException](#) if the value is less than or equal to the specified minimum.

```
public static void ThrowIfLessThanOrEqual<T>(T value, T min, string? paramName = null) where  
T : IComparable<T>
```

Parameters

value T

The value to check.

`min` T

The minimum exclusive bound.

`paramName` [string](#)

The parameter name for the value.

Type Parameters

T

A comparable type.

Exceptions

[ArgumentException](#)

Thrown if `value` <= `min`. Message: "The value must be greater than {0}."

ThrowIfLessThanOther<T>(T, T, string?, string?)

Throws an [ArgumentException](#) if the value is less than another parameter's value.

```
public static void ThrowIfLessThanOther<T>(T value, T other, string? paramName = null,  
    string? otherName = null) where T : IComparable<T>
```

Parameters

`value` T

The current value being validated.

`other` T

The comparison reference value.

`paramName` [string](#)

Name of the value parameter.

`otherName` [string](#)

Name of the comparison parameter.

Type Parameters

T

A comparable type.

Exceptions

[ArgumentException](#)

Thrown if `value > other`. Message: "The value must not be less than the value of {0}."

ThrowIfLessThan<T>(T?, T, bool, string?)

Throws an [ArgumentOutOfRangeException](#) if the value is less than the specified minimum. Optionally throws [ArgumentNullException](#) if the value is null.

```
public static void ThrowIfLessThan<T>(T? value, T min, bool throwIfNull = false, string?  
paramName = null) where T : struct, IComparable<T>
```

Parameters

`value` T?

The value to validate (nullable).

`min` T

The minimum value allowed.

`throwIfNull` bool

Whether to throw if `value` is null. Default is false.

`paramName` string

The name of the parameter being validated.

Type Parameters

T

A comparable value type.

Exceptions

[ArgumentNullException](#)

Thrown when `value` is null and `throwIfNull` is true.

[ArgumentOutOfRangeException](#)

Thrown if `value` is non-null and less than `min`. Message: "The value must be greater than or equal to {0}."

ThrowIfLessThan<T>(T, T, string?)

Throws an [ArgumentOutOfRangeException](#) if the value is less than the specified minimum.

```
public static void ThrowIfLessThan<T>(T value, T min, string? paramName = null) where T : IComparable<T>
```

Parameters

`value` T

The value to check.

`min` T

The minimum value allowed.

`paramName` [string](#)

The parameter name for the value.

Type Parameters

T

A comparable type.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `value < min`. Message: "The value must be greater than or equal to {0}."

ThrowIfNegative<T>(T, string?)

Throws an [ArgumentOutOfRangeException](#) if the value is less than zero.

```
public static void ThrowIfNegative<T>(T value, string? paramName = null) where T : IComparable<T>
```

Parameters

`value T`

The value to check.

`paramName string`

The name of the value parameter.

Type Parameters

`T`

A comparable numeric type.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `value < 0`. Message: "The value must be zero or positive."

ThrowIfNotBetweenExclusive<T>(T, T, T, string?)

Throws an [ArgumentOutOfRangeException](#) if the value is not between min and max (exclusive).

```
public static void ThrowIfNotBetweenExclusive<T>(T value, T min, T max, string? paramName =
```

```
null) where T : IComparable<T>
```

Parameters

value T

The value to validate.

min T

The lower exclusive bound.

max T

The upper exclusive bound.

paramName [string](#)

The name of the value parameter.

Type Parameters

T

A comparable type.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if **value** <= **min** or >= **max**. Message: "The value must be greater than {0} and less than {1}."

ThrowIfNotBetweenInclusive<T>(T, T, T, string?)

Throws an [ArgumentOutOfRangeException](#) if the value is not between min and max (inclusive).

```
public static void ThrowIfNotBetweenInclusive<T>(T value, T min, T max, string? paramName = null) where T : IComparable<T>
```

Parameters

value T

The value to validate.

`min` `T`

The lower inclusive bound.

`max` `T`

The upper inclusive bound.

`paramName` [string](#)

The name of the value parameter.

Type Parameters

`T`

A comparable type.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `value < min` or `> max`. Message: "The value must be between {0} and {1}, inclusive."

ThrowIfNotOfType<T>(object?, string?)

Throws an [ArgumentException](#) if the object is not assignable to type `T`.

```
public static void ThrowIfNotOfType<T>(object? value, string? paramName = null)
```

Parameters

`value` [object](#)

The value to check.

`paramName` [string](#)

The name of the value parameter.

Type Parameters

T

The target type to validate against.

Remarks

Null is only allowed if T is a reference or nullable type.

Exceptions

[ArgumentException](#)

Thrown if value is not null and not of type T, or if it is null and T is a non-nullable value type. Message: "Object must be of type {0}."

ThrowIfNotPositiveMultipleOf(int, int, string?)

Throws an [ArgumentOutOfRangeException](#) if the value is not a positive multiple of the specified divisor.

```
public static void ThrowIfNotPositiveMultipleOf(int value, int divisor, string? paramName  
= null)
```

Parameters

value [int](#)

The value to validate.

divisor [int](#)

The required positive divisor.

paramName [string](#)

The name of the parameter.

Remarks

Useful for validating aligned buffer sizes, memory boundaries, or block-aligned lengths.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `value` is not greater than zero or not divisible by `divisor`. Message: "The value must be a positive number and a multiple of {0}."

ThrowIfNotZero<T>(T, string?)

Throws an [ArgumentOutOfRangeException](#) if the value is not equal to zero.

```
public static void ThrowIfNotZero<T>(T value, string? paramName = null) where T : IEquatable<T>
```

Parameters

`value` `T`

The value to validate.

`paramName` [string](#)

The name of the parameter.

Type Parameters

`T`

A type that supports equality comparison.

Remarks

Ensures a value is exactly zero — commonly used for flags, counters, or reset validation.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `value` is not zero. Message: "The value must be zero."

ThrowIfNullOrEmpty(string, string?)

Throws an [ArgumentNullException](#) if the string is `null`, or an [ArgumentException](#) if it is empty.

```
public static void ThrowIfNullOrEmpty(string value, string? paramName = null)
```

Parameters

`value` [string](#)

The string value to validate.

`paramName` [string](#)

The name of the parameter.

Exceptions

[ArgumentNullException](#)

Thrown if `value` is `null`.

[ArgumentException](#)

Thrown if `value` is an empty string. Message: "The string was either null or empty."

ThrowIfNull<T>(T, string?)

Throws an [ArgumentNullException](#) if the provided value is `null`.

```
public static void ThrowIfNull<T>(T value, string? paramName = null)
```

Parameters

`value` T

The value to check.

`paramName` [string](#)

The name of the parameter being validated.

Type Parameters

T

The type of the object.

Exceptions

[ArgumentNullException](#)

Thrown if `value` is `null`. Message: "Value cannot be null."

ThrowIfNull<T>(T, string, string?)

Throws an [ArgumentNullException](#) if the value is `null`, with a custom message.

```
public static void ThrowIfNull<T>(T value, string message, string? paramName = null)
```

Parameters

`value` T

The value to validate.

`message` [string](#)

A custom error message for the exception.

`paramName` [string](#)

The parameter name.

Type Parameters

T

The type of the value.

Exceptions

[ArgumentNullException](#)

Thrown if `value` is `null`.

ThrowIfPositive<T>(T, string?)

Throws an [ArgumentOutOfRangeException](#) if the value is positive.

```
public static void ThrowIfPositive<T>(T value, string? paramName = null) where T : IComparable<T>
```

Parameters

`value` `T`

The value to check.

`paramName` [string](#)

The name of the value parameter.

Type Parameters

`T`

A comparable numeric type.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `value` > 0. Message: "The value must be zero or negative."

ThrowIfSequenceRangeOverflows(int, int, string?)

Throws an [ArgumentOutOfRangeException](#) if the calculated sequence would exceed the maximum value for [int](#).

```
public static void ThrowIfSequenceRangeOverflows(int start, int count, string? paramName = null)
```

Parameters

`start` [int](#)

The starting value of the sequence.

`count` [int](#)

The number of values to generate.

`paramName` [string](#)

The name of the parameter representing `count`.

Remarks

This check prevents arithmetic overflow when generating sequences like ranges.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown when `start + count - 1` would exceed [MaxValue](#).

ThrowIfSequenceRangeOverflows(long, int, string?)

Throws an [ArgumentOutOfRangeException](#) if the calculated sequence would exceed the maximum value for [long](#).

```
public static void ThrowIfSequenceRangeOverflows(long start, int count, string? paramName = null)
```

Parameters

`start` [long](#)

The starting value of the sequence.

`count` [int](#)

The number of values to generate.

`paramName` [string](#)

The name of the parameter representing `count`.

Remarks

This check prevents arithmetic overflow when generating long-based numeric sequences.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown when `start + count - 1` would exceed [MaxValue](#).

ThrowIfSpanLengthIsInsufficient<T>(Span<T>, int, int, string?)

Throws an [ArgumentException](#) if the remaining length of the span from the given index is less than required.

```
public static void ThrowIfSpanLengthIsInsufficient<T>(Span<T> span, int index, int requiredLength, string? paramName = null)
```

Parameters

`span` [Span](#)<T>

The span to check.

`index` [int](#)

The index from which to measure the remaining length.

`requiredLength` [int](#)

The required number of elements.

`paramName` [string](#)

The name of the span parameter.

Type Parameters

`T`

The element type of the span.

Exceptions

[ArgumentException](#)

Thrown when `span.Length - index < requiredLength`. Message: "Span is too short. Required minimum is {0} from a specified index."

ThrowIfSpanLengthNotPositiveMultipleOf<T> (`ReadOnlySpan<T>`, `int`, `Func<string, Exception>?`, `string?`)

Throws an [ArgumentException](#) if the span length is not a positive multiple of a given divisor.

```
public static void ThrowIfSpanLengthNotPositiveMultipleOf<T>(ReadOnlySpan<T> span, int divisor, Func<string, Exception>? func = null, string? paramName = null)
```

Parameters

`span` [ReadOnlySpan](#)<T>

The span to check.

`divisor` [int](#)

The divisor that span length must be a multiple of.

`func` [Func](#)<string, Exception>

A factory for a custom exception (unused in default implementation).

`paramName` [string](#)

The name of the span parameter.

Type Parameters

`T`

The element type of the span.

Exceptions

[ArgumentException](#)

Thrown when `span.Length == 0 || span.Length % divisor != 0`. Message: "Length of the Span must be a multiple of {0}."

ThrowIfZeroOrNegative<T>(T, string?)

Throws an [ArgumentOutOfRangeException](#) if the value is zero or negative.

```
public static void ThrowIfZeroOrNegative<T>(T value, string? paramName = null) where T : IComparable<T>
```

Parameters

`value` `T`

The value to check.

`paramName` [string](#)

The name of the value parameter.

Type Parameters

`T`

A comparable numeric type.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `value <= 0`. Message: "The value must be a positive number."

ThrowIfZeroOrPositive<T>(T, string?)

Throws an [ArgumentOutOfRangeException](#) if the value is zero or positive.

```
public static void ThrowIfZeroOrPositive<T>(T value, string? paramName = null) where T : IComparable<T>
```

Parameters

value T

The value to check.

paramName string

The name of the value parameter.

Type Parameters

T

A comparable numeric type.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if **value** >= 0. Message: "The value must be a negative number."

ThrowIfZero<T>(T, string?)

Throws an [ArgumentOutOfRangeException](#) if the value is zero.

```
public static void ThrowIfZero<T>(T value, string? paramName = null) where T : IEquatable<T>
```

Parameters

value T

The value to check.

paramName string

The name of the value parameter.

Type Parameters

T

A type that implements [IEquatable<T>](#).

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `value` equals 0. Message: "The value must not be zero."

ThrowIsNullOrEmptyWhiteSpace(string, string?)

Throws an [ArgumentNullException](#) if the value is `null`, or an [ArgumentException](#) if it is empty or whitespace.

```
public static void ThrowIsNullOrEmptyWhiteSpace(string value, string? paramName = null)
```

Parameters

`value` [string](#)

The string value to validate.

`paramName` [string](#)

The name of the parameter.

Exceptions

[ArgumentNullException](#)

Thrown if `value` is `null`.

[ArgumentException](#)

Thrown if `value` is empty or contains only whitespace. Message: "The string was either empty or contained only whitespace."

Class XorShiftRandom

Namespace: [Bodu](#)

Assembly: Bodu.CoreLib.dll

Represents a high-performance, non-cryptographic pseudo-random number generator based on the XOR-shift algorithm.

```
public sealed class XorShiftRandom : Random, IRandomGenerator
```

Inheritance

[object](#) ← [Random](#) ← XorShiftRandom

Implements

[IRandomGenerator](#)

Inherited Members

[Random.NextInt64\(\)](#) , [Random.NextInt64\(long\)](#) , [Random.NextInt64\(long, long\)](#) ,
[Random.NextBytes\(Span<byte>\)](#) , [Random.NextSingle\(\)](#) , [Random.Shared](#) , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#).

Remarks

This class is designed as a drop-in replacement for [Random](#) with better performance characteristics. It is not suitable for cryptographic purposes.

Constructors

XorShiftRandom()

Initializes a new instance of the [XorShiftRandom](#) class using a system-generated seed.

```
public XorShiftRandom()
```

Remarks

The default seed is derived from [TickCount](#) at the time of construction.

XorShiftRandom(int)

Initializes a new instance of the [XorShiftRandom](#) class with a 32-bit signed integer seed.

```
public XorShiftRandom(int seed)
```

Parameters

seed [int](#)

The seed used to initialize the random generator.

XorShiftRandom(uint)

Initializes a new instance of the [XorShiftRandom](#) class with a 32-bit unsigned seed.

```
public XorShiftRandom(uint seed)
```

Parameters

seed [uint](#)

The seed used to initialize the random generator.

Methods

Next()

Returns a non-negative random integer.

```
public override int Next()
```

Returns

[int](#)

A 32-bit signed integer that is greater than or equal to 0 and less than [MaxValue](#).

Next(int)

Returns a non-negative random integer that is less than the specified maximum.

```
public override int Next(int maxValue)
```

Parameters

`maxValue` [int](#)

The exclusive upper bound of the random number to be generated. `maxValue` must be greater than or equal to 0.

Returns

[int](#)

A 32-bit signed integer that is greater than or equal to 0, and less than `maxValue`; that is, the range of return values ordinarily includes 0 but not `maxValue`. However, if `maxValue` equals 0, `maxValue` is returned.

Exceptions

[ArgumentOutOfRangeException](#)

`maxValue` is less than 0.

Next(int, int)

Returns a random integer that is within a specified range.

```
public override int Next(int minValue, int maxValue)
```

Parameters

`minValue` [int](#)

The inclusive lower bound of the random number returned.

maxValue [int ↗](#)

The exclusive upper bound of the random number returned. **maxValue** must be greater than or equal to **minValue**.

Returns

[int ↗](#)

A 32-bit signed integer greater than or equal to **minValue** and less than **maxValue**; that is, the range of return values includes **minValue** but not **maxValue**. If **minValue** equals **maxValue**, **minValue** is returned.

Exceptions

[ArgumentOutOfRangeException ↗](#)

minValue is greater than **maxValue**.

NextBytes(byte[])

Fills the elements of a specified array of bytes with random numbers.

```
public override void NextBytes(byte[] buffer)
```

Parameters

buffer [byte\[\] ↗](#)

The array to be filled with random numbers.

Exceptions

[ArgumentNullException ↗](#)

buffer is [null ↗](#).

NextDouble()

Returns a random floating-point number that is greater than or equal to 0.0, and less than 1.0.

```
public override double NextDouble()
```

Returns

double ↗

A double-precision floating point number that is greater than or equal to 0.0, and less than 1.0.

Namespace Bodu.Buffers

Classes

[PooledBufferBuilder<T>](#)

Provides an efficient way to copy items from a sequence into a pooled buffer, automatically resizing as needed and supporting collection fast-paths.

Class PooledBufferBuilder<T>

Namespace: [Bodu.Buffers](#)

Assembly: Bodu.CoreLib.dll

Provides an efficient way to copy items from a sequence into a pooled buffer, automatically resizing as needed and supporting collection fast-paths.

```
public sealed class PooledBufferBuilder<T> : IDisposable
```

Type Parameters

T

The type of item.

Inheritance

[object](#) ← PooledBufferBuilder<T>

Implements

[IDisposable](#)

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#)

Constructors

PooledBufferBuilder(int)

Initializes a new instance of the [PooledBufferBuilder<T>](#) class.

```
public PooledBufferBuilder(int initialCapacity = 256)
```

Parameters

`initialCapacity` [int](#)

Optional initial buffer size.

Properties

Count

The number of valid elements in the buffer.

```
public int Count { get; }
```

Property Value

[int](#)

Methods

AppendRange(IEnumerable<T>)

Appends items from an [IEnumerable<T>](#), expanding the pooled buffer if needed.

```
public void AppendRange(IEnumerable<T> source)
```

Parameters

`source` [IEnumerable](#)<T>

The input sequence.

AsArray()

Returns the buffered elements as a span of valid items.

```
public T[] AsArray()
```

Returns

T[]

AsSpan()

Returns the buffered elements as a span of valid items.

```
public Span<T> AsSpan()
```

Returns

[Span](#)<T>

Dispose()

Returns the underlying buffer to the pool and resets internal state.

```
public void Dispose()
```

TryCopyFrom(IReadOnlyCollection<T>)

Attempts to fast-copy the source if it supports [ICollection<T>](#).

```
public bool TryCopyFrom(IReadOnlyCollection<T> source)
```

Parameters

source [IReadOnlyCollection](#)<T>

The input sequence.

Returns

[bool](#)

true if copied using [CopyTo](#) otherwise, use [AppendRange\(IEnumerable<T>\)](#).

Namespace Bodu.Collections.Extensions

Classes

[IEnumerableExtensions](#)

[SequenceGenerator](#)

Provides a collection of static helper methods for generating sequences of values.

Enums

[RecursiveSelectControl](#)

Represents predefined behaviors for how an element is processed during recursive selection.

Class IEnumerableExtensions

Namespace: [Bodu.Collections.Extensions](#)

Assembly: Bodu.CoreLib.dll

```
public static class IEnumerableExtensions
```

Inheritance

[object](#) ← IEnumerableExtensions

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

**RecursiveSelect(IEnumerable, Func<object,
IEnumerable<object>>, Func<object, int, int, object>)**

Recursively flattens and projects a hierarchical structure, providing index and depth information to the selector.

```
public static IEnumerable RecursiveSelect(this IEnumerable source, Func<object,  
IEnumerable<object>> childSelector, Func<object, int, int, object> selector)
```

Parameters

source [IEnumerable](#)

The root sequence to traverse recursively.

childSelector [Func<object, IEnumerable<object>>](#)

A delegate that returns child elements for a given item.

selector [Func<object, int, int, object>](#)

A projection that receives the element, its index, and its recursion depth.

Returns

[IEnumerable](#)

A flattened sequence of projected elements with access to structural context (index, depth).

Examples

```
var result = nodes.RecursiveSelect(  
    n => n.Children,  
    (n, i, depth) => new { n.Name, Index = i, IndentLevel = depth });
```

Remarks

Use when formatting output that depends on the depth of the node in the hierarchy (e.g., indentation, styling).

Exceptions

[ArgumentNullException](#)

Thrown if `source`, `childSelector`, or `selector` is [null](#).

RecursiveSelect(IEnumerable, Func<object, IEnumerable<object>>, Func<object, int, int, object>, Func<object, RecursiveSelectControl>)

Recursively flattens and projects a hierarchical structure, allowing control over whether to recurse into children.

```
public static IEnumerable RecursiveSelect(this IEnumerable source, Func<object,  
    IEnumerable<object>> childSelector, Func<object, int, int, object> selector, Func<object,  
    RecursiveSelectControl> recursionControl)
```

Parameters

`source` [IEnumerable](#)

The root sequence to traverse recursively.

`childSelector` [Func<object, IEnumerable<object>>](#)

A delegate that returns child elements for a given item.

`selector` [Func<object, int, object>](#)

A projection that receives the element, its index, and its recursion depth.

`recursionControl` [Func<object, RecursiveSelectControl>](#)

A delegate that determines how each element in the sequence should be handled during recursion. It returns a [RecursiveSelectControl](#) value indicating whether to yield the element, recurse into its children, skip it, or terminate the traversal entirely.

Returns

[IEnumerable](#)

A sequence of projected elements, where children are included only if `recursionControl` returns [true](#).

Examples

```
var filtered = nodes.RecursiveSelect(  
    n => n.Children,  
    (n, i, d) => n.Name,  
    n => n.Children?.Count() > 2  
    ? RecursiveSelectControl.Recurse  
    : RecursiveSelectControl.YieldOnly);
```

Remarks

Useful when pruning the recursion tree — e.g., limiting depth, skipping inactive branches, or filtering by condition.

Exceptions

[ArgumentNullException](#)

Thrown if `source`, `childSelector`, `selector`, or `recursionControl` is [null](#).

`RecursiveSelect(IEnumerable, Func<object, IEnumerable<object>>, Func<object, int, object>)`

Recursively flattens and projects a hierarchical structure, passing the index of each element to the projection.

```
public static IEnumerable RecursiveSelect(this IEnumerable source, Func<object, IEnumerable<object>> childSelector, Func<object, int, object> selector)
```

Parameters

source [IEnumerable](#)

The root sequence to traverse recursively.

childSelector [Func<object, IEnumerable<object>>](#)

A delegate that returns child elements for a given item.

selector [Func<object, int, object>](#)

A projection that receives each element and its zero-based index.

Returns

[IEnumerable](#)

A depth-first, recursively flattened sequence with projected values using the element and its index.

Examples

```
var labeled = nodes.RecursiveSelect(n => n.Children, (n, i) => $"{i}: {n.Name}");
```

Remarks

Use this when you need both element content and positional context during recursion.

Exceptions

[ArgumentNullException](#)

Thrown if **source**, **childSelector**, or **selector** is [null](#).

RecursiveSelect(IEnumerable, Func<object, IEnumerable<object>>, Func<object, object>)

Recursively flattens and projects a hierarchical structure into a linear sequence using a projection selector.

```
public static IEnumerable RecursiveSelect(this IEnumerable source, Func<object, IEnumerable<object>> childSelector, Func<object, object> selector)
```

Parameters

source [IEnumerable](#)

The root sequence to traverse recursively.

childSelector [Func<object, IEnumerable<object>>](#)

A delegate that returns child elements for a given item.

selector [Func<object, object>](#)

A projection applied to each element.

Returns

[IEnumerable](#)

A depth-first, recursively flattened sequence of projected elements.

Examples

```
var names = nodes.RecursiveSelect(n => n.Children, n => n.Name);
```

Remarks

This overload is useful when you want to flatten and transform the hierarchy into a different shape or type.

Exceptions

[ArgumentNullException](#)

Thrown if `source`, `childSelector`, or `selector` is [null](#).

RecursiveSelect(IEnumerable, Func<object, IEnumerable>)

Recursively flattens a hierarchical or tree-like [IEnumerable](#) into a single linear sequence.

```
public static IEnumerable RecursiveSelect(this IEnumerable source, Func<object, IEnumerable> childSelector)
```

Parameters

`source` [IEnumerable](#)

The root sequence to traverse recursively. Each element is expected to potentially contain child elements retrievable via the `childSelector` delegate.

`childSelector` [Func](#)<[object](#), [IEnumerable](#)>

A delegate that returns the child elements for a given item. This should return an [IEnumerable](#) representing the recursive children of the current element, or [null](#) if none.

Returns

[IEnumerable](#)

A flattened [IEnumerable](#) that yields all elements in depth-first order, including their recursive children.

Examples

```
class Category { public string Name; public List<Category> Children; }
var root = new List<Category> { ... };
var all = root.RecursiveSelect(c => c.Children).Cast<Category>();
```

Remarks

This method is useful for navigating recursive structures like directory trees, category hierarchies, or comment threads.

Execution is deferred and will only begin when the resulting sequence is enumerated.

All elements are treated as [object](#) and may need casting to their actual types.

Exceptions

[ArgumentNullException](#)

Thrown if `source` or `childSelector` is [null](#).

Enum RecursiveSelectControl

Namespace: [Bodu.Collections.Extensions](#)

Assembly: Bodu.CoreLib.dll

Represents predefined behaviors for how an element is processed during recursive selection.

```
public enum RecursiveSelectControl
```

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#)

Fields

RecurseOnly = 2

Recurse into children without yielding the current element.

SkipAndBreak = 12

Skip yielding the current element and stop traversal at the current level.

SkipAndExit = 20

Skip yielding the current element and stop traversal completely across all levels.

SkipAndRecurse = 6

Skip yielding the current element, but still recurse into its children.

SkipOnly = 4

Skip yielding the current element and do not recurse.

YieldAndBreak = 9

Yield the current element, then stop traversal at the current level (like a break).

YieldAndExit = 17

Yield the current element, then stop traversal completely across all levels.

YieldAndRecurse = 3

Yield the current element and recurse into its children. This is the default behavior.

`YieldOnly = 1`

Yield the current element only, without recursing into children.

Remarks

This enum defines composable control flags that determine whether an element should be yielded, whether its children should be traversed, and whether recursion should stop after the current element. These values are intended to be treated as bitmasks using [int](#) values (not [Flags]).

Typical usage includes returning one of these values from a `Func<T, RecursiveSelectControl>` to control behavior dynamically within a recursive traversal method.

Behavior Matrix

Combination	Yield?	Recurse?	Break Level?	Exit All?
SkipOnly	✗	✗	✗	✗
YieldOnly	✓	✗	✗	✗
YieldAndRecurse	✓	✓	✗	✗
YieldAndBreak	✓	✗	✓	✗
YieldAndExit	✓	✗	✗	✓
RecurseOnly	✗	✓	✗	✗
SkipAndRecurse	✗	✓	✗	✗
SkipAndBreak	✗	✗	✓	✗
SkipAndExit	✗	✗	✗	✓

Class SequenceGenerator

Namespace: [Bodu.Collections.Extensions](#)

Assembly: Bodu.CoreLib.dll

Provides a collection of static helper methods for generating sequences of values.

```
public static class SequenceGenerator
```

Inheritance

[object](#) ← SequenceGenerator

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

This partial class is intended to contain reusable sequence generators such as numeric ranges, repeated patterns, and combinatoric sequences.

Methods

FareySequence(int)

Generates the Farey sequence of a given order n, consisting of reduced fractions between 0 and 1 inclusive.

```
public static IEnumerable<(int Numerator, int Denominator)> FareySequence(int order)
```

Parameters

order [int](#)

The order of the Farey sequence. Must be positive.

Returns

[IEnumerable](#) <(int Numerator, int Denominator)>

A sequence of tuples representing simplified fractions (numerator, denominator) in ascending order.

Remarks

The Farey sequence of order n includes all unique fractions a/b such that: $0 \leq a \leq b \leq n$, $\text{GCD}(a, b) = 1$.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `order` is less than 1.

Fibonacci(long, long)

Generates the Fibonacci sequence values within the specified inclusive range.

```
public static IEnumerable<long> Fibonacci(long min, long max)
```

Parameters

`min` [long](#)

The inclusive minimum value for the sequence.

`max` [long](#)

The exclusive maximum value for the sequence.

Returns

[IEnumerable](#) <[long](#)>

An enumerable of Fibonacci numbers between `min` and `max`.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if either `min` or `max` is negative.

[ArgumentException](#)

Thrown if `min` is greater than `max`.

Leibniz(double, double)

Generates terms from the **Leibniz series**: $F(n) = (-1)^n / (2n + 1)$, which converges to $\pi/4$ as $n \rightarrow \infty$.

```
public static IEnumerable<double> Leibniz(double min, double max)
```

Parameters

`min` [double](#)

The minimum absolute value (inclusive) a term must have to be included in the output. Must be greater than or equal to 0.

`max` [double](#)

The maximum absolute value (exclusive) a term may have before generation stops. Must be greater than or equal to 0 and not less than `min`.

Returns

[IEnumerable](#) <[double](#)>

A lazily-evaluated sequence of [double](#) values from the Leibniz series, each within the specified range of absolute values.

Remarks

The Leibniz series is an infinite alternating series defined as:

$$F(n) = (-1)^n / (2n + 1)$$

It converges to $\pi/4$ as n approaches infinity.

This generator emits only terms where `abs(term) ≥ min` and `abs(term) < max`.

This method is suitable for illustrating convergence, alternating series behavior, or computing partial approximations of π via summation and multiplying the result by 4.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown when `min` or `max` is less than 0.

[ArgumentException](#)

Thrown when `min` is greater than `max`.

LookAndSay(int)

Generates terms in the Look-and-Say sequence starting from "1".

```
public static IEnumerable<string> LookAndSay(int count)
```

Parameters

`count` [int](#)

The number of terms to generate. Must be positive.

Returns

[IEnumerable](#)<[string](#)>

A sequence of strings, each representing a term in the Look-and-Say sequence.

Remarks

Each term describes the digits of the previous one, e.g.: 1, 11, 21, 1211, 111221, ...

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `count` is less than 1.

ThueMorse(int)

Generates the Thue–Morse sequence as a binary sequence of 0s and 1s.

```
public static IEnumerable<int> ThueMorse(int count)
```

Parameters

count [int](#)

The number of terms to generate. Must be non-negative.

Returns

[IEnumerable](#) <[int](#)>

A sequence of [int](#) values where each term is 0 or 1, representing the Thue–Morse sequence.

Remarks

The n-th value is the parity of the number of 1s in the binary representation of n: $T(n) = \text{bitcount}(n) \% 2$

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if **count** is negative.

Namespace Bodu.Collections.Generic

Classes

[CircularBuffer<T>](#)

Represents a first-in, first-out collection (FirstInFirstOut) of objects using a fixed buffer and automatic overwrite support.

[EvictingDictionary< TKey, TValue >](#)

Represents a fixed-capacity dictionary that automatically removes entries based on a chosen eviction policy, such as First-In-First-Out (FirstInFirstOut), Least Recently Used (LeastRecentlyUsed), or Least Frequently Used (LeastFrequentlyUsed).

[SequenceGenerator](#)

[ShuffleHelpers](#)

Provides shared utilities for in-place and yield-based randomization using Fisher–Yates shuffle.

Structs

[CircularBuffer<T>.Enumerator](#)

Enumerates the elements of a [CircularBuffer<T>](#).

[EvictingDictionary< TKey, TValue >.DictionaryEnumerator](#)

Enumerates the elements of a [EvictingDictionary< TKey, TValue >](#).

Enums

[EvictionPolicy](#)

Specifies the eviction strategy used by a [EvictingDictionary< TKey, TValue >](#) when its capacity is exceeded.

Class CircularBuffer<T>

Namespace: [Bodu.Collections.Generic](#)

Assembly: Bodu.CoreLib.dll

Represents a first-in, first-out collection (FirstInFirstOut) of objects using a fixed buffer and automatic overwrite support.

```
[Serializable]
public class CircularBuffer<T> : ICollection, IReadOnlyCollection<T>,
IEnumerable<T>, IEnumerable
```

Type Parameters

T

Specifies the type of elements in the collection.

Inheritance

[object](#) ← CircularBuffer<T>

Implements

[ICollection](#), [IReadOnlyCollection](#)<T>, [IEnumerable](#)<T>, [IEnumerable](#)

Inherited Members

[object.Equals\(object\)](#), [object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),
[object.MemberwiseClone\(\)](#), [object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#), [IEnumerableExtensions.Randomize<T>\(IEnumerable<T>\)](#),
[IEnumerableExtensions.Randomize<T>\(IEnumerable<T>, RandomizationMode, IRandomGenerator, int?\)](#),

[IEnumerableExtensions.RecursiveSelect<TSource>\(IEnumerable<TSource>, Func<TSource, IEnumerable<TSource>>\)](#),
[IEnumerableExtensions.RecursiveSelect<TSource, TResult>\(IEnumerable<TSource>, Func<TSource, IEnumerable<TSource>>, Func<TSource, int, int, TResult>\)](#),
[IEnumerableExtensions.RecursiveSelect<TSource, TResult>\(IEnumerable<TSource>, Func<TSource, IEnumerable<TSource>>, Func<TSource, int, int, TResult>, Func<TSource, RecursiveSelectControl>\)](#),
[IEnumerableExtensions.RecursiveSelect<TSource, TResult>\(IEnumerable<TSource>, Func<TSource, IEnumerable<TSource>>, Func<TSource, int, TResult>\)](#),

[IEnumerableExtensions.RecursiveSelect<TSource, TResult>\(IEnumerable<TSource>, Func<TSource, IEnumerable<TSource>, Func<TSource, TResult>\)](#),
[IEnumerableExtensions.RecursiveSelect\(IEnumerable, Func<object, IEnumerable<object>>, Func<object, int, int, object>\)](#),
[IEnumerableExtensions.RecursiveSelect\(IEnumerable, Func<object, IEnumerable<object>>, Func<object, int, int, object>, Func<object, RecursiveSelectControl>\)](#),
[IEnumerableExtensions.RecursiveSelect\(IEnumerable, Func<object, IEnumerable<object>>, Func<object, int, object>\)](#),
[IEnumerableExtensions.RecursiveSelect\(IEnumerable, Func<object, IEnumerable<object>>, Func<object, object>\)](#),
[IEnumerableExtensions.RecursiveSelect\(IEnumerable, Func<object, IEnumerable>\)](#).

Remarks

This class implements a generic a circular buffer of a fixed capacity. Objects stored in a [CircularBuffer<T>](#) are inserted at one end and removed from the other. Use [CircularBuffer<T>](#) if you need to limited the number of elements in the collection and to access the information from oldest to newest in the collection. Use [Queue<T>](#) if you do not need to restrict the number of elements in the collection.

Three main operations can be performed on a [CircularBuffer<T>](#) and its elements:

- [Enqueue\(T\)](#) adds an element to the end of the [CircularBuffer<T>](#).
- [Dequeue\(\)](#) removes the oldest element from the start of the [CircularBuffer<T>](#).
- [Peek\(\)](#) returns the oldest element that is at the start of the [CircularBuffer<T>](#) but does not remove it from the [CircularBuffer<T>](#).

The [Capacity](#) of a [CircularBuffer<T>](#) is the maximum number of elements the [CircularBuffer<T>](#) can hold. As elements are added to a full [CircularBuffer<T>](#) the oldest elements are overwritten by default. Setting the [AllowOverwrite](#) to [false](#) will throw an [InvalidOperationException](#) if an attempt is made to add an element to a [CircularBuffer<T>](#) where the [Count](#) equals [Capacity](#).

[CircularBuffer<T>](#) accepts [null](#) (Nothing in Visual Basic) as a valid value for reference types and allows duplicate elements.

Constructors

[CircularBuffer\(\)](#)

Initializes a new instance of the [CircularBuffer<T>](#) class with the default capacity and allows overwriting by default.

```
public CircularBuffer()
```

Remarks

The default capacity is defined by the [DefaultCapacity](#) constant. When the buffer is full, adding a new item will overwrite the oldest element.

CircularBuffer(IEnumerable<T>)

Initializes a new [CircularBuffer<T>](#) by copying elements from the specified collection. Uses Bodu.Collections.Generic.CircularBuffer<T>.DefaultCapacity as the capacity. Oldest items are overwritten if necessary.

```
public CircularBuffer(IEnumerable<T> collection)
```

Parameters

[collection](#) [IEnumerable](#)<T>

The collection from which elements are copied. Must not be null.

Remarks

If the number of elements in [collection](#) exceeds Bodu.Collections.Generic.CircularBuffer<T>.DefaultCapacity, only the most recent items are retained.

Exceptions

[ArgumentNullException](#)

Thrown when [collection](#) is [null](#).

CircularBuffer(IEnumerable<T>, int)

Initializes a new [CircularBuffer<T>](#) by copying elements from the specified collection and applying the given capacity.

```
public CircularBuffer(IEnumerable<T> collection, int capacity)
```

Parameters

collection [IEnumerable<T>](#)

The collection from which elements are copied. Must not be null.

capacity [int](#)

The maximum number of elements the buffer can contain. Must be non-negative.

Remarks

If the number of elements in **collection** exceeds **capacity**, the most recent items are retained.

Exceptions

[ArgumentNullException](#)

Thrown when **collection** is [null](#).

[ArgumentOutOfRangeException](#)

Thrown when **capacity** is less than zero.

CircularBuffer(IEnumerable<T>, int, bool)

Initializes a new [CircularBuffer<T>](#) by copying elements from the specified collection, and using the specified capacity and overwrite behavior.

```
public CircularBuffer(IEnumerable<T> collection, int capacity, bool allowOverwrite)
```

Parameters

collection [IEnumerable<T>](#)

The collection from which elements are copied. Must not be null.

capacity [int](#)

The maximum number of elements the buffer can contain. Must be greater than zero.

allowOverwrite [bool](#)

If `true`, the most recent elements from the collection are retained if its size exceeds capacity. If `false`, the collection size must not exceed capacity, or an exception is thrown.

Exceptions

[ArgumentNullException](#)

Thrown when `collection` is `null`.

[ArgumentOutOfRangeException](#)

Thrown when `capacity` is less than zero.

[InvalidOperationException](#)

Thrown when `allowOverwrite` is `false` and the collection size exceeds `capacity`.

CircularBuffer(int)

Initializes an empty [`CircularBuffer<T>`](#) with the specified capacity. Oldest elements are automatically overwritten when capacity is exceeded.

```
public CircularBuffer(int capacity)
```

Parameters

`capacity` [int](#)

The maximum number of elements the buffer can contain. Must be non-negative.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown when `capacity` is less than zero.

CircularBuffer(int, bool)

Initializes an empty [`CircularBuffer<T>`](#) with the specified capacity and overwrite behavior.

```
public CircularBuffer(int capacity, bool allowOverwrite)
```

Parameters

capacity [int](#)

The maximum number of elements the buffer can contain. Must be greater than 0.

allowOverwrite [bool](#)

If [true](#), oldest elements are overwritten when capacity is reached; otherwise, adding beyond capacity will throw an exception.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown when [capacity](#) is less than zero.

Properties

AllowOverwrite

Gets or sets a value indicating whether the [CircularBuffer<T>](#) will automatically overwrite the oldest element when capacity is reached.

```
public bool AllowOverwrite { get; set; }
```

Property Value

[bool](#)

[true](#) to automatically overwrite the oldest element when adding an element to a [CircularBuffer<T>](#) where the [Count](#) equals [Capacity](#) otherwise, [false](#).

Remarks

Setting the [AllowOverwrite](#) to [false](#) will throw an [InvalidOperationException](#) if an attempt is made to add an element to a [CircularBuffer<T>](#) where the [Count](#) equals [Capacity](#). Otherwise the oldest element is overwritten.

Capacity

Gets the maximum number of elements that the [CircularBuffer<T>](#) can contain.

```
public int Capacity { get; }
```

Property Value

[int](#)

The maximum number of elements that the current [CircularBuffer<T>](#) can contain.

Count

Gets the number of elements contained in the [ICollection](#).

```
public int Count { get; }
```

Property Value

[int](#)

The number of elements contained in the [ICollection](#).

this[int]

Gets the element at the specified zero-based index from the oldest to the newest element.

```
public T this[int index] { get; }
```

Parameters

[index](#) [int](#)

The zero-based index (0 is the oldest element).

Property Value

T

The element at the specified position.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown when `index` is negative or greater than or equal to [Count](#).

Methods

Clear()

Removes all elements from the [CircularBuffer<T>](#), resetting its state.

```
public void Clear()
```

Remarks

The buffer's capacity remains unchanged after clearing.

Contains(T)

Determines whether the buffer contains a specific element.

```
public bool Contains(T item)
```

Parameters

`item` T

The element to locate in the buffer.

Returns

[bool](#)

[true](#) if `item` is found; otherwise, [false](#).

CopyTo(T[], int)

Copies elements from the buffer to an existing array, starting at the specified array index.

```
public void CopyTo(T[] array, int index)
```

Parameters

array T[]

The target array for the copied elements.

index int

The zero-based index at which copying begins.

Exceptions

[ArgumentNullException](#)

Thrown if **array** is null.

[ArgumentOutOfRangeException](#)

Thrown if **index** is negative.

[ArgumentException](#)

Thrown if insufficient space in the target array.

Dequeue()

Removes and returns the oldest element from the buffer.

```
public T Dequeue()
```

Returns

T

The oldest element in the buffer.

Exceptions

[InvalidOperationException](#)

Thrown if the buffer is empty.

Enqueue(T)

Adds an element to the end of the buffer.

```
public void Enqueue(T item)
```

Parameters

item T

The element to add. Can be null for reference types.

Exceptions

[InvalidOperationException](#)

Thrown if the buffer reaches capacity and overwriting is disabled.

GetEnumerator()

Returns an enumerator that iterates through the [CircularBuffer<T>](#).

```
public CircularBuffer<T>.Enumerator GetEnumerator()
```

Returns

[CircularBuffer<T>.Enumerator](#)

An [CircularBuffer<T>.Enumerator](#) that can be used to iterate through the buffer.

Remarks

The [foreach](#) statement in C# (**For Each** in Visual Basic) hides the complexity of using enumerators. Therefore, using [foreach](#) is recommended over directly manipulating the enumerator.

The enumerator provides read-only access to the collection. It cannot be used to modify the underlying buffer.

Initially, the enumerator is positioned before the first element. At this position, [Current](#) is undefined. You must call [MoveNext\(\)](#) to advance to the first element before reading [Current](#).

[Current](#) returns the same value until [MoveNext\(\)](#) is called. Each call to [MoveNext\(\)](#) advances the enumerator to the next element.

If [MoveNext\(\)](#) passes the end of the buffer, the enumerator is positioned after the last element and returns [false](#). At this point, [Current](#) is undefined, and you must create a new enumerator instance to iterate again.

The enumerator is invalidated if the buffer is modified. Subsequent calls to [MoveNext\(\)](#) or [Reset\(\)](#) will throw an [InvalidOperationException](#).

Enumeration is not thread-safe. To guarantee thread safety, lock the buffer during enumeration. [CircularBuffer<T>](#) is not synchronized by default.

GetSegments()

Provides direct read-only access to the internal buffer storage. Returns two array segments due to potential buffer wrap-around.

```
public (ArraySegment<T> FirstSegment, ArraySegment<T> SecondSegment) GetSegments()
```

Returns

[\(ArraySegment<T> FirstSegment, ArraySegment<T> SecondSegment\)](#)

A tuple containing two array segments (FirstSegment, SecondSegment). Use both segments sequentially to read all elements.

Peek()

Returns the oldest element without removing it from the buffer.

```
public T Peek()
```

Returns

T

The oldest element in the buffer.

Exceptions

[InvalidOperationException](#)

Thrown if the buffer is empty.

ToArray()

Copies the buffer elements to a new array.

```
public T[] ToArray()
```

Returns

T[]

A new array containing the buffer's elements from oldest to newest.

TrimExcess()

Reduces the buffer capacity to match the current count, freeing unused memory.

```
public void TrimExcess()
```

Remarks

Useful if the buffer previously had a large capacity but now requires less memory.

If the buffer is empty ([Count](#) is 0), the capacity is reduced to a minimal internal size (typically 1) to ensure the buffer remains usable.

TryDequeue(out T)

Attempts to remove and return the oldest item without throwing exceptions.

```
public bool TryDequeue(out T item)
```

Parameters

item **T**

The item removed from the buffer.

Returns

[bool](#)

[true](#) if an item was dequeued successfully; [false](#) if the buffer was empty.

TryEnqueue(T)

Attempts to add an item to the buffer without throwing exceptions.

```
public bool TryEnqueue(T item)
```

Parameters

item **T**

The item to enqueue.

Returns

[bool](#)

[true](#) if the item was enqueued successfully; [false](#) if the buffer is full and overwriting is disabled.

TryPeek(out T)

Attempts to retrieve the oldest element without removing it from the buffer.

```
public bool TryPeek(out T item)
```

Parameters

item T

The oldest element in the buffer.

Returns

[bool](#)

[true](#) if an item was retrieved successfully; [false](#) if the buffer was empty.

Events

ItemEvicted

Occurs immediately **after** an item has been evicted from the [CircularBuffer<T>](#) due to capacity limits.

```
public event Action<T>? ItemEvicted
```

Event Type

[Action](#)<T>

Examples

```
var buffer = new CircularBuffer<string>(capacity: 2, allowOverwrite: true);
buffer.ItemEvicted += item => Console.WriteLine($"Evicted: {item}");

buffer.Enqueue("A");
buffer.Enqueue("B");
buffer.Enqueue("C"); // Triggers ItemEvicted for "A"
```

Remarks

This event is raised only when the buffer reaches capacity and [AllowOverwrite](#) is [true](#).

The event provides the evicted item, which is no longer present in the buffer. It can be used to record history, notify observers, or synchronize with external systems.

Important: Exceptions thrown from event handlers will propagate to the caller of [Enqueue\(T\)](#) or [TryEnqueue\(T\)](#). Consumers should handle exceptions appropriately.

ItemEvicting

Occurs immediately **before** an item is evicted from the [CircularBuffer<T>](#) due to capacity limits.

```
public event Action<T>? ItemEvicting
```

Event Type

[Action](#) <T>

Examples

```
var buffer = new CircularBuffer<string>(capacity: 2, allowOverwrite: true);
buffer.ItemEvicting += item => Console.WriteLine($"Evicting: {item}");

buffer.Enqueue("A");
buffer.Enqueue("B");
buffer.Enqueue("C"); // Triggers ItemEvicting for "A"
```

Remarks

This event is raised only when the buffer reaches capacity and [AllowOverwrite](#) is [true](#).

It allows consumers to inspect the item that is about to be evicted. This is useful for diagnostics, logging, or synchronizing external state.

Important: Exceptions thrown from event handlers will propagate to the caller and may prevent the item from being overwritten.

Struct CircularBuffer<T>.Enumerator

Namespace: [Bodu.Collections.Generic](#)

Assembly: Bodu.CoreLib.dll

Enumerates the elements of a [CircularBuffer<T>](#).

```
[Serializable]
public struct CircularBuffer<T>.Enumerator : IEnumarator<T>, IEnumarator, IDisposable
```

Implements

[IEnumarator](#)<T>, [IEnumarator](#), [IDisposable](#)

Inherited Members

[ValueType.Equals\(object\)](#), [ValueType.GetHashCode\(\)](#), [ValueType.ToString\(\)](#),
[object.Equals\(object, object\)](#), [object.GetType\(\)](#), [object.ReferenceEquals\(object, object\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#)

Remarks

Use the [foreach](#) statement to simplify the enumeration process instead of directly using this enumerator.

The enumerator provides read-only access to the collection's elements. Modifying the underlying collection while enumerating invalidates the enumerator.

Properties

Current

Gets the element in the collection at the current position of the enumerator.

```
public T Current { get; }
```

Property Value

T

The element in the collection at the current position of the enumerator.

Methods

Dispose()

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

```
public void Dispose()
```

MoveNext()

Advances the enumerator to the next element of the collection.

```
public bool MoveNext()
```

Returns

[bool](#)

[true](#) if the enumerator was successfully advanced to the next element; [false](#) if the enumerator has passed the end of the collection.

Exceptions

[InvalidOperationException](#)

The collection was modified after the enumerator was created.

Reset()

Sets the enumerator to its initial position, which is before the first element in the collection.

```
public void Reset()
```

Exceptions

[InvalidOperationException](#)

The collection was modified after the enumerator was created.

Class EvictingDictionary<TKey, TValue>

Namespace: [Bodu.Collections.Generic](#)

Assembly: Bodu.CoreLib.dll

Represents a fixed-capacity dictionary that automatically removes entries based on a chosen eviction policy, such as First-In-First-Out (FirstInFirstOut), Least Recently Used (LeastRecentlyUsed), or Least Frequently Used (LeastFrequentlyUsed).

```
[Serializable]
public class EvictingDictionary<TKey, TValue> : IDictionary<TKey, TValue>,
ICollection<KeyValuePair<TKey, TValue>>, IEnumerable<KeyValuePair<TKey, TValue>>,
IDictionary, ICollection, IEnumerable where TKey : notnull
```

Type Parameters

TKey

Specifies the type of keys in the dictionary.

TValue

Specifies the type of values in the dictionary.

Inheritance

[object](#) ← EvictingDictionary<TKey, TValue>

Implements

[IDictionary](#)<TKey, TValue>, [ICollection](#)<KeyValuePair<TKey, TValue>>,
[IEnumerable](#)<KeyValuePair<TKey, TValue>>, [IDictionary](#), [ICollection](#), [IEnumerable](#)

Inherited Members

[object.Equals\(object\)](#), [object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),
[object.MemberwiseClone\(\)](#), [object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#), [IEnumerableExtensions.Randomize<T>\(IEnumerable<T>\)](#),
[IEnumerableExtensions.Randomize<T>\(IEnumerable<T>, RandomizationMode, IRandomGenerator, int?\)](#),
,

[IEnumerableExtensions.RecursiveSelect<TSource>\(IEnumerable<TSource>, Func<TSource, IEnumerable<TSource>>\)](#),

[IEnumerableExtensions.RecursiveSelect<TSource, TResult>\(IEnumerable<TSource>, Func<TSource, IEnumerable<TSource>, Func<TSource, int, int, TResult>\)](#),
[IEnumerableExtensions.RecursiveSelect<TSource, TResult>\(IEnumerable<TSource>, Func<TSource, IEnumerable<TSource>, Func<TSource, int, int, TResult>, Func<TSource, RecursiveSelectControl>\)](#),
[IEnumerableExtensions.RecursiveSelect<TSource, TResult>\(IEnumerable<TSource>, Func<TSource, IEnumerable<TSource>, Func<TSource, int, TResult>\)](#),
[IEnumerableExtensions.RecursiveSelect<TSource, TResult>\(IEnumerable<TSource>, Func<TSource, IEnumerable<TSource>, Func<TSource, int, TResult>\)](#),
[IEnumerableExtensions.RecursiveSelect<TSource, TResult>\(IEnumerable<TSource>, Func<TSource, IEnumerable<TSource>, Func<TSource, int, TResult>\)](#),
[IEnumerableExtensions.RecursiveSelect\(IEnumerable, Func<object, IEnumerable<object>, Func<object, int, int, object>\)](#),
[IEnumerableExtensions.RecursiveSelect\(IEnumerable, Func<object, IEnumerable<object>, Func<object, int, int, object>, Func<object, RecursiveSelectControl>\)](#),
[IEnumerableExtensions.RecursiveSelect\(IEnumerable, Func<object, IEnumerable<object>, Func<object, int, object>\)](#),
[IEnumerableExtensions.RecursiveSelect\(IEnumerable, Func<object, IEnumerable<object>, Func<object, object>\)](#),
[IEnumerableExtensions.RecursiveSelect\(IEnumerable, Func<object, IEnumerable>\)](#).

Remarks

[EvictingDictionary< TKey, TValue >](#) maintains a maximum number of key-value pairs and automatically evicts items when capacity is exceeded. Eviction is determined by a specified [EvictionPolicy](#), allowing this dictionary to behave like a queue, an access-order cache, or a frequency-based cache.

[EvictingDictionary< TKey, TValue >](#) allows [null](#) keys and values (for reference types) and supports custom key equality via [IEqualityComparer<T>](#).

Constructors

EvictingDictionary()

Initializes a new empty [EvictingDictionary< TKey, TValue >](#) with the default capacity and eviction policy.

```
public EvictingDictionary()
```

Remarks

The default capacity is 16, and the default eviction policy is [LeastRecentlyUsed](#).

EvictingDictionary(IEnumerable<KeyValuePair< TKey, TValue >>)

Initializes a new [EvictingDictionary< TKey, TValue >](#) by copying entries from the specified dictionary, using default capacity and eviction policy.

```
public EvictingDictionary(IEnumerable<KeyValuePair< TKey, TValue >> source)
```

Parameters

source [IEnumerable<KeyValuePair< TKey, TValue >>](#)

The enumerable collection of key-value pairs to copy. Must not be null.

Remarks

If the number of entries in **source** exceeds the default capacity (16), only the most recent entries are retained according to the default eviction policy ([LeastRecentlyUsed](#)).

Exceptions

[ArgumentNullException](#)

Thrown when **source** is [null](#).

EvictingDictionary(IEnumerable<KeyValuePair< TKey, TValue >>, EvictionPolicy)

Initializes a new [EvictingDictionary< TKey, TValue >](#) by copying entries from the specified dictionary, using the specified eviction policy and default capacity.

```
public EvictingDictionary(IEnumerable<KeyValuePair< TKey, TValue >> source, EvictionPolicy policy)
```

Parameters

source [IEnumerable<KeyValuePair< TKey, TValue >>](#)

The enumerable collection of key-value pairs to copy. Must not be null.

policy [EvictionPolicy](#)

The eviction policy to use when the dictionary exceeds its capacity.

Exceptions

[ArgumentNullException](#)

Thrown when `source` is [null](#).

EvictingDictionary(int)

Initializes a new empty [EvictingDictionary<TKey, TValue>](#) with the specified capacity and the default eviction policy.

```
public EvictingDictionary(int capacity)
```

Parameters

`capacity` [int](#)

The maximum number of key-value pairs the dictionary can contain. Must be positive.

Remarks

The default eviction policy is [LeastRecentlyUsed](#).

Exceptions

[ArgumentOutOfRangeException](#)

Thrown when `capacity` is less than or equal to zero.

EvictingDictionary(int, EvictionPolicy)

Initializes a new empty [EvictingDictionary<TKey, TValue>](#) with the specified capacity and eviction policy.

```
public EvictingDictionary(int capacity, EvictionPolicy policy)
```

Parameters

capacity [int](#)

The maximum number of key-value pairs the dictionary can contain. Must be positive.

policy [EvictionPolicy](#)

The eviction policy to use when the dictionary exceeds its capacity.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown when `capacity` is less than or equal to zero.

EvictingDictionary(int, EvictionPolicy, IEqualityComparer<TKey>?)

Initializes a new empty [EvictingDictionary<TKey, TValue>](#) with the specified capacity, eviction policy, and key comparer.

```
public EvictingDictionary(int capacity, EvictionPolicy policy, IEqualityComparer<TKey>? comparer)
```

Parameters

capacity [int](#)

The maximum number of key-value pairs the dictionary can contain. Must be positive.

policy [EvictionPolicy](#)

The eviction policy to use when the dictionary exceeds its capacity.

comparer [IEqualityComparer](#)<TKey>

An equality comparer to use for comparing keys, or [null](#) to use the default comparer.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown when `capacity` is less than or equal to zero. A non-zero capacity is required to allow storing items.

EvictingDictionary(int, IEnumerable<KeyValuePair<TKey, TValue>>)

Initializes a new [EvictingDictionary<TKey, TValue>](#) by copying entries from the specified enumerable source, using the specified capacity and the default eviction policy ([LeastRecentlyUsed](#)).

```
public EvictingDictionary(int capacity, IEnumerable<KeyValuePair<TKey, TValue>> source)
```

Parameters

`capacity` [int](#)

The maximum number of key-value pairs the dictionary can contain. Must be a positive integer.

`source` [IEnumerable](#)<[KeyValuePair](#)<TKey, TValue>>

The enumerable collection of key-value pairs to copy into the dictionary. Must not be [null](#).

Remarks

If the number of elements in `source` exceeds `capacity`, only the most recently added entries will be retained based on the default eviction policy ([LeastRecentlyUsed](#)).

Exceptions

[ArgumentNullException](#)

Thrown when `source` is [null](#).

[ArgumentOutOfRangeException](#)

/// Thrown when `capacity` is less than or equal to zero.

EvictingDictionary(int, IEnumerable<KeyValuePair<TKey, TValue>>, EvictionPolicy)

Initializes a new [EvictingDictionary<TKey, TValue>](#) by copying entries from the specified dictionary, using the specified capacity and eviction policy.

```
public EvictingDictionary(int capacity, IEnumerable<KeyValuePair<TKey, TValue>> source,  
EvictionPolicy policy)
```

Parameters

capacity [int](#)

The maximum number of key-value pairs the dictionary can contain. Must be positive.

source [IEnumerable<KeyValuePair<TKey, TValue>>](#)

The enumerable collection of key-value pairs to copy. Must not be null.

policy [EvictionPolicy](#)

The eviction policy to use when the dictionary exceeds its capacity.

Exceptions

[ArgumentNullException](#)

Thrown when **source** is [null](#).

[ArgumentOutOfRangeException](#)

Thrown when **capacity** is less than or equal to zero.

EvictingDictionary(int, IEnumerable<KeyValuePair<TKey, TValue>>, EvictionPolicy, IEqualityComparer<TKey>?)

Initializes a new [EvictingDictionary<TKey, TValue>](#) by copying entries from the specified dictionary, using the specified capacity, eviction policy, and key comparer.

```
public EvictingDictionary(int capacity, IEnumerable<KeyValuePair<TKey, TValue>> source,  
EvictionPolicy policy, IEqualityComparer<TKey>? comparer)
```

Parameters

capacity [int](#)

The maximum number of key-value pairs the dictionary can contain. Must be positive.

source [IEnumerable](#)<[KeyValuePair](#)<TKey, TValue>>

The enumerable collection of key-value pairs to copy. Must not be null.

policy [EvictionPolicy](#)

The eviction policy to use when the dictionary exceeds its capacity.

comparer [IEqualityComparer](#)<TKey>

An equality comparer to use for comparing keys, or [null](#) to use the default comparer.

Exceptions

[ArgumentNullException](#)

Thrown when `source` is [null](#).

[ArgumentOutOfRangeException](#)

Thrown when `capacity` is less than or equal to zero.

EvictingDictionary(int, IEqualityComparer<TKey>?)

Initializes a new empty [EvictingDictionary](#)<TKey, TValue> with the specified capacity and key comparer.

```
public EvictingDictionary(int capacity, IEqualityComparer<TKey>? comparer)
```

Parameters

capacity [int](#)

The maximum number of key-value pairs the dictionary can contain. Must be positive.

comparer [IEqualityComparer](#)<TKey>

An equality comparer to use for comparing keys, or [null](#) to use the default comparer.

Remarks

The default eviction policy is [LeastRecentlyUsed](#).

Exceptions

[ArgumentOutOfRangeException](#)

Thrown when `capacity` is less than or equal to zero.

Properties

Capacity

Gets the maximum number of items that can be stored in the dictionary before eviction occurs.

```
public int Capacity { get; }
```

Property Value

[int](#)

Count

Gets the number of elements contained in the [ICollection<T>](#).

```
public int Count { get; }
```

Property Value

[int](#)

The number of elements contained in the [ICollection<T>](#).

EvictionCount

Gets the total number of items evicted from the dictionary since creation.

```
public long EvictionCount { get; }
```

Property Value

[long](#)

IsReadOnly

Gets a value indicating whether the [ICollection<T>](#) is read-only.

```
public bool IsReadOnly { get; }
```

Property Value

[bool](#)

[true](#) if the [ICollection<T>](#) is read-only; otherwise, [false](#).

this[TKey]

Gets or sets the element with the specified key.

```
public TValue this[TKey key] { get; set; }
```

Parameters

key TKey

The key of the element to get or set.

Property Value

TValue

The element with the specified key.

Exceptions

[ArgumentNullException](#)

key is [null](#).

[KeyNotFoundException](#)

The property is retrieved and **key** is not found.

[NotSupportedException](#)

The property is set and the [IDictionary<TKey, TValue>](#) is read-only.

Keys

Gets an [ICollection<T>](#) containing the keys of the [IDictionary<TKey, TValue>](#).

```
public ICollection<TKey> Keys { get; }
```

Property Value

[ICollection<TKey>](#)

An [ICollection<T>](#) containing the keys of the object that implements [IDictionary<TKey, TValue>](#).

Policy

Gets the eviction policy configured for this dictionary.

```
public EvictionPolicy Policy { get; }
```

Property Value

[EvictionPolicy](#)

TotalTouches

Gets the total number of times any key has been accessed or touched.

```
public long TotalTouches { get; }
```

Property Value

[long](#)

Values

Gets an [ICollection<T>](#) containing the values in the [IDictionary< TKey, TValue >](#).

```
public ICollection<TValue> Values { get; }
```

Property Value

[ICollection](#)<TValue>

An [ICollection<T>](#) containing the values in the object that implements [IDictionary< TKey, TValue >](#).

Methods

Add(KeyValuePair< TKey, TValue >)

Adds the specified key/value pair to the dictionary. If the dictionary has reached its capacity, an existing entry will be evicted according to the configured [EvictionPolicy](#).

```
public void Add(KeyValuePair< TKey, TValue > item)
```

Parameters

item [KeyValuePair](#)<TKey, TValue>

The key/value pair to add to the dictionary.

Exceptions

[ArgumentNullException](#)

If **item.Key** is [null](#) and the key type is a reference type.

Add(TKey, TValue)

Adds the specified key and value to the dictionary. If the dictionary has reached its capacity, an existing entry will be evicted according to the configured [EvictionPolicy](#).

```
public void Add(TKey key, TValue value)
```

Parameters

key TKey

The key of the element to add.

value TValue

The value of the element to add.

Exceptions

[ArgumentNullException](#)

If **key** is [null](#) and the key type is a reference type.

Clear()

Removes all entries from the dictionary and resets internal tracking counters.

```
public void Clear()
```

Remarks

This operation clears the dictionary and resets all internal eviction metadata, including access order (for LeastRecentlyUsed), frequency tracking (for LeastFrequentlyUsed), and counters such as [TotalTouches](#) and [EvictionCount](#).

Contains(KeyValuePair<TKey, TValue>)

Determines whether the [ICollection<T>](#) contains a specific value.

```
public bool Contains(KeyValuePair<TKey, TValue> item)
```

Parameters

item [KeyValuePair<TKey, TValue>](#)

The object to locate in the [ICollection<T>](#).

Returns

[bool](#)

[true](#) if **item** is found in the [ICollection<T>](#); otherwise, [false](#).

ContainsKey(TKey)

Determines whether the [IDictionary<TKey, TValue>](#) contains an element with the specified key.

`public bool ContainsKey(TKey key)`

Parameters

key TKey

The key to locate in the [IDictionary<TKey, TValue>](#).

Returns

[bool](#)

[true](#) if the [IDictionary<TKey, TValue>](#) contains an element with the key; otherwise, [false](#).

Exceptions

[ArgumentNullException](#)

key is [null](#).

CopyTo(KeyValuePair<TKey, TValue>[], int)

Copies the elements of the [ICollection<T>](#) to an [Array](#), starting at a particular [Array](#) index.

```
public void CopyTo(KeyValuePair<TKey, TValue>[] array, int index)
```

Parameters

array [KeyValuePair](#)<TKey, TValue>[]

The one-dimensional [Array](#) that is the destination of the elements copied from [ICollection<T>](#). The [Array](#) must have zero-based indexing.

index [int](#)

Exceptions

[ArgumentNullException](#)

array is [null](#).

[ArgumentOutOfRangeException](#)

arrayIndex is less than 0.

[ArgumentException](#)

The number of elements in the source [ICollection<T>](#) is greater than the available space from arrayIndex to the end of the destination array.

GetEnumerator()

Returns an enumerator that iterates through the collection.

```
public I IEnumerator<KeyValuePair<TKey, TValue>> GetEnumerator()
```

Returns

[IEnumerator](#)<[KeyValuePair](#)<TKey, TValue>>

An enumerator that can be used to iterate through the collection.

PeekEvictionCandidate()

Returns the key that would be evicted next based on the current eviction policy and internal state.

```
public TKey? PeekEvictionCandidate()
```

Returns

TKey

The key that is next in line for eviction.

Remarks

The eviction candidate depends on the selected [EvictionPolicy](#):

- **FirstInFirstOut**: returns the oldest inserted key.
- **LeastRecentlyUsed**: returns the least recently accessed key.
- **MostRecentlyUsed**: returns the most recently accessed key.
- **LeastFrequentlyUsed**: returns the key with the fewest total accesses.
- **RandomReplacement**: returns an arbitrary key from the dictionary.
- **SecondChance**: returns the first key that has not been accessed recently; falls back to FIFO if all have second chances.

Remove(KeyValuePair<TKey, TValue>)

Removes the first occurrence of a specific object from the [ICollection<T>](#).

```
public bool Remove(KeyValuePair<TKey, TValue> item)
```

Parameters

item [KeyValuePair](#)<TKey, TValue>

The object to remove from the [ICollection<T>](#).

Returns

[bool](#)

[true](#) if **item** was successfully removed from the [ICollection<T>](#); otherwise, [false](#). This method also returns [false](#) if **item** is not found in the original [ICollection<T>](#).

Exceptions

[NotSupportedException](#)

The [ICollection<T>](#) is read-only.

Remove(TKey)

Removes the element with the specified key from the [IDictionary< TKey, TValue >](#).

```
public bool Remove(TKey key)
```

Parameters

key TKey

The key of the element to remove.

Returns

[bool](#)

[true](#) if the element is successfully removed; otherwise, [false](#). This method also returns [false](#) if [key](#) was not found in the original [IDictionary< TKey, TValue >](#).

Exceptions

[ArgumentNullException](#)

[key](#) is [null](#).

[NotSupportedException](#)

The [IDictionary< TKey, TValue >](#) is read-only.

Touch(TKey)

Marks the specified key as recently accessed without retrieving its value. If the eviction policy involves usage tracking, this updates the internal usage metadata.

```
public bool Touch(TKey key)
```

Parameters

key TKey

The key to touch.

Returns

[bool](#)

[true](#) if the key exists and was marked as accessed; otherwise, [false](#).

TouchOrThrow(TKey)

Marks the specified key as recently accessed without retrieving its value, and throws an exception if the key does not exist in the dictionary.

```
public void TouchOrThrow(TKey key)
```

Parameters

key TKey

The key to touch.

Remarks

If the eviction policy is [LeastRecentlyUsed](#) or [LeastFrequentlyUsed](#), this updates the internal usage metadata.

Exceptions

[KeyNotFoundException](#)

Thrown when the specified key does not exist in the dictionary.

TryGetValue(TKey, out TValue)

Attempts to retrieve the value associated with the specified key.

```
public bool TryGetValue(TKey key, out TValue value)
```

Parameters

key TKey

The key of the value to retrieve.

value TValue

When this method returns, contains the value associated with the specified key, if the key is found; otherwise, the default value for the type of the value parameter.

Returns

[bool](#)

[true](#) if the dictionary contains an element with the specified key; otherwise, [false](#).

Remarks

If the eviction policy is [LeastRecentlyUsed](#) or [LeastFrequentlyUsed](#), accessing a key through this method will update its usage metadata.

Events

ItemEvicted

Occurs immediately **after** an item is evicted from the [EvictingDictionary<TKey, TValue>](#) due to capacity limits.

```
public event Action<TKey, TValue>? ItemEvicted
```

Event Type

[Action](#)<TKey, TValue>

Examples

```

var cache = new EvictingDictionary<string, int>(capacity: 2, policy:
EvictionPolicy.FirstInFirstOut);
cache.ItemEvicted += (key, value) =>
{
Console.WriteLine($"[AfterEvict] {key} = {value}");
};

cache.Add("A", 1);
cache.Add("B", 2);
cache.Add("C"); // Triggers ItemEvicted for "A"

```

Remarks

This event is raised after the item has been removed from the collection, based on the configured [EvictionPolicy](#) (e.g., FirstInFirstOut, LeastRecentlyUsed, or LeastFrequentlyUsed).

Consumers can use this event to record historical data, notify observers, or synchronize external caches. The key and value provided are no longer present in the dictionary.

ItemEvicting

Occurs immediately **before** an item is evicted from the [EvictingDictionary<TKey, TValue>](#) due to capacity limits.

```
public event Action<TKey, TValue>? ItemEvicting
```

Event Type

[Action](#)<TKey, TValue>

Examples

```

var cache = new EvictingDictionary<string, int>(capacity: 2, policy:
EvictionPolicy.FirstInFirstOut);
cache.ItemEvicting += (key, value) =>
{
Console.WriteLine($"[BeforeEvict] {key} = {value}");
};

cache.Add("A", 1);

```

```
cache.Add("B", 2);
cache.Add("C", 3); // Triggers ItemEvicting for "A"
```

Remarks

This event is raised before the item is removed from the collection, allowing consumers to inspect the key and value before eviction occurs.

Common use cases include diagnostics, logging, cache warm-up, or state mirroring. This event is informational and cannot cancel or delay eviction.

Struct EvictingDictionary< TKey, TValue>.DictionaryEnumerator

Namespace: [Bodu.Collections.Generic](#)

Assembly: Bodu.CoreLib.dll

Enumerates the elements of a [EvictingDictionary< TKey, TValue>](#).

```
public struct EvictingDictionary< TKey, TValue>.DictionaryEnumerator :  
IDictionaryEnumerator, IEnumera
```

Implements

[IDictionaryEnumerator](#), [IEnumerator](#)

Inherited Members

[ValueType.Equals\(object\)](#), [ValueType.GetHashCode\(\)](#), [ValueType.ToString\(\)](#),
[object.Equals\(object, object\)](#), [object.GetType\(\)](#), [object.ReferenceEquals\(object, object\)](#)

Extension Methods

[NumericExtensions.GetBytes< T >\(T, bool\)](#)

Remarks

Use the [foreach](#) statement to simplify the enumeration process instead of directly using this enumerator.

The enumerator provides read-only access to the dictionary's elements. Modifying the underlying dictionary while enumerating invalidates the enumerator.

Constructors

DictionaryEnumerator(EvictingDictionary< TKey, TValue>)

Initializes a new instance of the [EvictingDictionary< TKey, TValue>.DictionaryEnumerator](#) struct.

```
public DictionaryEnumerator(EvictingDictionary< TKey, TValue> dictionary)
```

Parameters

[dictionary](#) [EvictingDictionary](#)<TKey, TValue>

The dictionary to enumerate.

Properties

Current

Gets the element in the collection at the current position of the enumerator.

```
public object Current { get; }
```

Property Value

[object](#)

The element in the collection at the current position of the enumerator.

Entry

Gets both the key and the value of the current dictionary entry.

```
public DictionaryEntry Entry { get; }
```

Property Value

[DictionaryEntry](#)

A [DictionaryEntry](#) containing both the key and the value of the current dictionary entry.

Exceptions

[InvalidOperationException](#)

The [IDictionaryEnumerator](#) is positioned before the first entry of the dictionary or after the last entry.

Key

Gets the key of the current dictionary entry.

```
public object Key { get; }
```

Property Value

[object](#)

The key of the current element of the enumeration.

Exceptions

[InvalidOperationException](#)

The [IDictionaryEnumerator](#) is positioned before the first entry of the dictionary or after the last entry.

Value

Gets the value of the current dictionary entry.

```
public object? Value { get; }
```

Property Value

[object](#)

The value of the current element of the enumeration.

Exceptions

[InvalidOperationException](#)

The [IDictionaryEnumerator](#) is positioned before the first entry of the dictionary or after the last entry.

Methods

MoveNext()

Advances the enumerator to the next element of the collection.

```
public bool MoveNext()
```

Returns

[bool](#)

[true](#) if the enumerator was successfully advanced to the next element; [false](#) if the enumerator has passed the end of the collection.

Exceptions

[InvalidOperationException](#)

The collection was modified after the enumerator was created.

Reset()

Sets the enumerator to its initial position, which is before the first element in the collection.

```
public void Reset()
```

Exceptions

[InvalidOperationException](#)

The collection was modified after the enumerator was created.

Enum EvictionPolicy

Namespace: [Bodu.Collections.Generic](#)

Assembly: Bodu.CoreLib.dll

Specifies the eviction strategy used by a [EvictingDictionary<TKey, TValue>](#) when its capacity is exceeded.

```
public enum EvictionPolicy
```

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#)

Fields

FirstInFirstOut = 0

First-In-First-Out: the oldest item added to the dictionary is removed first, regardless of usage.

LeastFrequentlyUsed = 2

Least Frequently Used: the item with the fewest total accesses is removed first.

LeastRecentlyUsed = 1

Least Recently Used: the item that has not been accessed for the longest time is removed first.

MostRecentlyUsed = 3

Most Recently Used: the most recently accessed item is evicted first.

RandomReplacement = 4

Random Replacement: a randomly selected item is removed.

SecondChance = 5

Second Chance: items are given a second chance before being evicted, based on a reference flag.

Remarks

The [EvictionPolicy](#) enumeration defines how items are selected for removal when the dictionary reaches its maximum capacity. This allows the dictionary to behave like a queue, cache, or frequency-based store, depending on the selected strategy.

The following eviction strategies are available:

- [FirstInFirstOut](#) - removes the oldest inserted item first, regardless of access pattern.
- [LeastRecentlyUsed](#) - removes the item that has not been accessed for the longest period of time.
- [LeastFrequentlyUsed](#) - removes the item with the lowest number of access operations over its lifetime.
- [MostRecentlyUsed](#) - removes the most recently accessed item first.
- [RandomReplacement](#) - evicts an item chosen at random.
- [SecondChance](#) - gives items a second chance before eviction, based on a reference bit.

Class SequenceGenerator

Namespace: [Bodu.Collections.Generic](#)

Assembly: Bodu.CoreLib.dll

```
public static class SequenceGenerator
```

Inheritance

[object](#) ← SequenceGenerator

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

Factory<TResult>(Func<IEnumerator<TResult>>)

Creates a sequence from a custom enumerator factory.

```
public static IEnumerable<TResult> Factory<TResult>(Func<IEnumerator<TResult>>  
enumeratorFactory)
```

Parameters

enumeratorFactory [Func](#)<[IEnumerator](#)<TResult>>

A delegate that produces a new [IEnumerator<T>](#) each time the sequence is iterated.

Returns

[IEnumerable](#)<TResult>

An enumerable sequence generated from the custom enumerator.

Type Parameters

TResult

The type of elements in the generated sequence.

Remarks

This method is useful when integrating external or imperative enumerators into a deferred LINQ-style pipeline.

Exceptions

[ArgumentNullException](#)

Thrown if `enumeratorFactory` is `null`.

NextWhile<TResult>(TResult, Func<TResult, bool>, Func<TResult, int, TResult>)

Generates a sequence by repeatedly transforming a value while a condition remains true, providing the current index to each transformation.

```
public static IEnumerable<TResult> NextWhile<TResult>(TResult initialValue, Func<TResult, bool> conditionHandler, Func<TResult, int, TResult> resultSelector)
```

Parameters

`initialValue` TResult

The initial value of the sequence.

`conditionHandler` [Func](#)<TResult, bool>

A function that determines whether the sequence should continue based on the current value.

`resultSelector` [Func](#)<TResult, int, TResult>

A function that computes the next value based on the current value and its zero-based index in the sequence.

Returns

[IEnumerable](#)<TResult>

An enumerable sequence generated by applying the transformation while the condition is met.

Type Parameters

TResult

The type of elements in the generated sequence.

Remarks

This overload allows the transformation to depend on the iteration index.

Exceptions

[ArgumentNullException](#)

Thrown if `conditionHandler` or `resultSelector` is `null`.

NextWhile<TResult>(TResult, Func<TResult, bool>, Func<TResult, TResult>)

Generates a sequence by repeatedly transforming a value while a condition remains true.

```
public static IEnumerable<TResult> NextWhile<TResult>(TResult initialValue, Func<TResult, bool> conditionHandler, Func<TResult, TResult> resultSelector)
```

Parameters

`initialValue` TResult

The starting value of the sequence.

`conditionHandler` [Func](#)<TResult, bool>

A function that determines whether the sequence should continue, based on the current value.

`resultSelector` [Func](#)<TResult, TResult>

A function that computes the next value in the sequence from the current value.

Returns

[IEnumerable](#)<TResult>

An enumerable sequence generated by applying the transformation while the condition is met.

Type Parameters

TResult

The type of elements in the generated sequence.

Remarks

This method yields a lazy sequence where the next value is generated by applying `resultSelector` to the current one, provided that `conditionHandler` evaluates to true.

Exceptions

[ArgumentNullException](#)

Thrown if `conditionHandler` or `resultSelector` is `null`.

NextWhile<TState, TResult>(TState, Func<TState, bool>, Func<TState, TState>, Func<TState, TResult>)

Generates a sequence by transforming a custom state object while a condition remains true.

```
public static IEnumerable<TResult> NextWhile<TState, TResult>(TState initialState, Func<TState, bool> conditionHandler, Func<TState, TState> iterateFunction, Func<TState, TResult> resultSelector)
```

Parameters

`initialState` TState

The initial state used to generate the sequence.

`conditionHandler` [Func](#)<TState, bool>

A predicate that determines whether the sequence should continue based on the current state.

`iterateFunction` [Func](#)<TState, TState>

A function that computes the next state from the current state.

`resultSelector` [Func](#)<TState, TResult>

A function that projects the current state into a sequence value.

Returns

[IEnumerable](#)<TResult>

An enumerable sequence generated by iterating the state transformation while the condition is met.

Type Parameters

TState

The type of the internal state object.

TResult

The type of the elements in the generated sequence.

Remarks

Use this overload to generate sequences where multiple values must be tracked (e.g., multiple counters, accumulators, etc.).

Exceptions

[ArgumentNullException](#)

Thrown if `conditionHandler`, `iterateFunction`, or `resultSelector` is `null`.

Range(int, int)

Generates a sequence of integers from `start` to `stop`, inclusive.

```
public static IEnumerable<int> Range(int start, int stop)
```

Parameters

start [int](#)

The first value in the sequence.

stop [int](#)

The final value in the sequence, included in the result.

Returns

[IEnumerable](#)<[int](#)>

A sequence of consecutive integers.

Remarks

If **start** is greater than **stop**, the sequence will decrement.

Range(int, int, int)

Generates a sequence of integers from **start** to **stop** using a specified step value.

```
public static IEnumerable<int> Range(int start, int stop, int step)
```

Parameters

start [int](#)

The initial value in the sequence.

stop [int](#)

The endpoint of the sequence, included if reached by stepping.

step [int](#)

The amount to increment or decrement per step. If zero, an infinite sequence of **start** is returned.

Returns

[IEnumerable](#)<[int](#)>

A sequence of integers spaced by **step** units.

Remarks

This method supports ascending, descending, and constant sequences depending on the sign of `step`. If `step` is zero, the sequence yields `start` indefinitely. The method safely handles edge cases near [MaxValue](#) and [MinValue](#) by terminating if overflow would occur.

Range(long, int)

Generates a fixed-length sequence of 64-bit integers starting at `start`.

```
public static IEnumerable<long> Range(long start, int count)
```

Parameters

`start` [long](#)

The initial value in the sequence.

`count` [int](#)

The number of elements to produce.

Returns

[IEnumerable](#) <[long](#)>

A sequence of `count` long values beginning at `start`.

Remarks

This method avoids producing values beyond the limits of [long](#) to prevent overflow.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown when `count` is negative, or when the range would overflow a 64-bit signed integer.

Repeat<T>(T)

Generates an infinite sequence repeating the specified value.

```
public static IEnumerable<T> Repeat<T>(T value)
```

Parameters

value **T**

The value to repeat in the sequence.

Returns

[IEnumerable](#) <T>

An infinite sequence of **value**.

Type Parameters

T

The type of the repeated value.

Remarks

This method yields the same value forever and is lazily evaluated.

Repeat<T>(T, int)

Generates a finite sequence repeating the specified value a given number of times.

```
public static IEnumerable<T> Repeat<T>(T value, int count)
```

Parameters

value **T**

The value to repeat.

count [int](#)

The number of repetitions.

Returns

[IEnumerable](#) <T>

A sequence of **value** repeated **count** times.

Type Parameters

T

The type of the repeated value.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown when **count** is negative.

Class ShuffleHelpers

Namespace: [Bodu.Collections.Generic](#)

Assembly: Bodu.CoreLib.dll

Provides shared utilities for in-place and yield-based randomization using Fisher–Yates shuffle.

```
public static class ShuffleHelpers
```

Inheritance

[object](#) ← ShuffleHelpers

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

ShuffleAndYield<T>(IEnumerable<T>, IRandomGenerator, int)

Lazily yields a randomized subset of an [IEnumerable<T>](#) using a partial Fisher–Yates shuffle.

```
public static IEnumerable<T> ShuffleAndYield<T>(IEnumerable<T> source, IRandomGenerator rng,  
int count)
```

Parameters

source [IEnumerable](#)<T>

The source sequence to shuffle.

rng [IRandomGenerator](#)

The random number generator used to select shuffled items.

count [int](#)

The number of elements to yield from the shuffled source.

Returns

[IEnumerable](#) <T>

A lazily-evaluated sequence of randomly selected items from the source.

Type Parameters

T

The type of elements in the source sequence.

Remarks

Enumeration of `source` and shuffling are deferred until the result is first iterated. The entire source is buffered in memory before shuffling begins. The original sequence is not modified.

Exceptions

[ArgumentNullException](#)

Thrown when `source` or `rng` is `null`.

[ArgumentOutOfRangeException](#)

Thrown immediately if `count` is negative.

[ArgumentOutOfRangeException](#)

May be thrown upon first enumeration if `count` exceeds the total number of elements in `source`.

ShuffleAndYield<T>(Memory<T>, IRandomGenerator, int)

Yields a randomized subset of a [Memory](#)<T> block using a copied buffer and array shuffle.

```
public static IEnumerable<T> ShuffleAndYield<T>(Memory<T> memory, IRandomGenerator rng, int count)
```

Parameters

memory [Memory](#)<T>

The memory block to shuffle.

`rng` [IRandomGenerator](#)

The random number generator to use.

`count` [int](#)

The number of elements to yield.

Returns

[IEnumerable](#) <T>

A sequence of shuffled items from the memory block.

Type Parameters

T

The type of elements in memory.

Remarks

This method eagerly copies `memory` into a new array, which is then shuffled.

ShuffleAndYield<T>(ReadOnlySpan<T>, IRandomGenerator, int)

Yields a randomized subset of a [ReadOnlySpan<T>](#) using a copied buffer and array shuffle.

```
public static IEnumerable<T> ShuffleAndYield<T>(ReadOnlySpan<T> span, IRandomGenerator rng, int count)
```

Parameters

`span` [ReadOnlySpan](#) <T>

The span to shuffle.

`rng` [IRandomGenerator](#)

The random number generator to use.

`count` [int](#)

The number of elements to yield.

Returns

[IEnumerable](#)<T>

A sequence of shuffled items from the span.

Type Parameters

T

The type of elements in the span.

Remarks

This method eagerly copies `span` into a new array, which is then shuffled.

ShuffleAndYield<T>(T[], IRandomGenerator, int)

Yields a randomized subset of the specified array by copying and shuffling it using a partial Fisher–Yates algorithm.

```
public static IEnumerable<T> ShuffleAndYield<T>(T[] array, IRandomGenerator rng, int count)
```

Parameters

`array` T[]

The source array to shuffle. The original array is not modified.

`rng` [IRandomGenerator](#)

The random number generator used for shuffling.

`count` [int](#)

The number of elements to yield.

Returns

[IEnumerable](#)<T>

A sequence of randomly selected elements from the array.

Type Parameters

T

The type of elements in the array.

Remarks

The input array is copied before shuffling to ensure immutability of the original. Use this method when working with arrays and requiring source preservation.

Exceptions

[ArgumentNullException](#)

Thrown when `array` or `rng` is `null`.

[ArgumentOutOfRangeException](#)

Thrown if `count` is negative or exceeds the array length.

Shuffle<T>(Memory<T>, IRandomGenerator)

Performs an in-place Fisher–Yates shuffle over a memory region.

```
public static void Shuffle<T>(Memory<T> memory, IRandomGenerator rng)
```

Parameters

`memory` [Memory](#)<T>

The memory region to shuffle.

`rng` [IRandomGenerator](#)

The random number generator used to shuffle elements.

Type Parameters

T

The type of the elements in the memory block.

Remarks

This method shuffles the contents of the `memory` region in-place by accessing its underlying span. The original memory region is modified.

Exceptions

[ArgumentNullException](#)

Thrown if `rng` is `null`.

Shuffle<T>(Span<T>, IRandomGenerator)

Performs an in-place Fisher–Yates shuffle over a span of elements.

```
public static void Shuffle<T>(Span<T> span, IRandomGenerator rng)
```

Parameters

`span` [Span](#)<T>

The span of elements to shuffle.

`rng` [IRandomGenerator](#)

The random number generator used to shuffle elements.

Type Parameters

T

The type of the elements in the span.

Remarks

This method modifies the span in-place using the Fisher–Yates algorithm. It is optimized for shuffling stack-allocated or pooled data, and does not allocate memory.

Exceptions

[ArgumentNullException](#)

Thrown if `rng` is `null`.

Shuffle<T>(T[], IRandomGenerator)

Performs an in-place Fisher–Yates shuffle over the provided array.

```
public static void Shuffle<T>(T[] array, IRandomGenerator rng)
```

Parameters

array T[]

The array of elements to shuffle.

rng [IRandomGenerator](#)

The random number generator used to shuffle elements.

Type Parameters

T

The type of the elements in the array.

Remarks

This method modifies the original array using the Fisher–Yates algorithm. Each element has an equal probability of ending up in any position.

Exceptions

[ArgumentNullException](#)

Thrown if `array` or `rng` is `null`.

Namespace Bodu.Collections.Generic.Extensions

Classes

[IEnumerableExtensions](#)

[SystemRandomAdapter](#)

Wraps [Random](#) as an [IRandomGenerator](#) implementation.

Enums

[RandomizationMode](#)

Specifies the available strategies for handling sequences during randomization.

Class IEnumerableExtensions

Namespace: [Bodu.Collections.Generic.Extensions](#)

Assembly: Bodu.CoreLib.dll

```
public static class IEnumerableExtensions
```

Inheritance

[object](#) ← IEnumerableExtensions

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

Randomize<T>(IEnumerable<T>)

Randomizes the source sequence using a specified strategy and generator.

```
public static IEnumerable<T> Randomize<T>(this IEnumerable<T> source)
```

Parameters

source [IEnumerable](#)<T>

The sequence to randomize.

Returns

[IEnumerable](#)<T>

A randomized sequence of T.

Type Parameters

T

The element type.

Exceptions

[ArgumentNullException](#)

If `source` or `rng` is null.

[ArgumentOutOfRangeException](#)

If `count` is negative or too large.

[ArgumentException](#)

If `count` is required but not provided.

Randomize<T>(IEnumerable<T>, RandomizationMode, IRandomGenerator, int?)

Randomizes the source sequence using a specified strategy and generator.

```
public static IEnumerable<T> Randomize<T>(this IEnumerable<T> source, RandomizationMode mode, IRandomGenerator rng, int? count = null)
```

Parameters

`source` [IEnumerable](#)<T>

The sequence to randomize.

`mode` [RandomizationMode](#)

The randomization strategy.

`rng` [IRandomGenerator](#)

The random number generator.

`count` [int](#)?

The number of items to return; all if null.

Returns

[IEnumerable](#)<T>

A randomized sequence of `T`.

Type Parameters

`T`

The element type.

Exceptions

[ArgumentNullException](#)

If `source` or `rng` is null.

[ArgumentOutOfRangeException](#)

If `count` is negative or too large.

[ArgumentException](#)

If `count` is required but not provided.

RecursiveSelect<TSource>(IEnumerable<TSource>, Func<TSource, IEnumerable<TSource>>)

Recursively flattens a hierarchical sequence using the provided child selector.

```
public static IEnumerable<TSource> RecursiveSelect<TSource>(this IEnumerable<TSource> source, Func<TSource, IEnumerable<TSource>> childSelector)
```

Parameters

`source` [IEnumerable](#)<TSource>

The root sequence to begin recursion from.

`childSelector` [Func](#)<TSource, [IEnumerable](#)<TSource>>

A function that returns child elements for a given element.

Returns

[IEnumerable](#) <TSource>

A flattened sequence of all elements including their children.

Type Parameters

TSource

The type of the elements in the sequence.

Examples

```
var allNodes = rootNodes.RecursiveSelect(node => node.Children);
```

Exceptions

[ArgumentNullException](#)

Thrown if `source` or `childSelector` is null.

RecursiveSelect<TSource, TResult>(IEnumerable<TSource>, Func<TSource, IEnumerable<TSource>>, Func<TSource, int, int, TResult>)

Recursively flattens and transforms a hierarchical sequence using child selector and a selector that receives index and depth.

```
public static IEnumerable<TResult> RecursiveSelect<TSource, TResult>(this  
IEnumerable<TSource> source, Func<TSource, IEnumerable<TSource>> childSelector,  
Func<TSource, int, int, TResult> selector)
```

Parameters

source [IEnumerable](#) <TSource>

The root sequence to begin recursion from.

childSelector [Func](#) <TSource, [IEnumerable](#) <TSource>>

A function that returns child elements for a given element.

selector [Func](#)<TSource, int, int, TResult>

A transform applied to each element, receiving index and depth.

Returns

[IEnumerable](#)<TResult>

A flattened and projected sequence from all elements including children.

Type Parameters

TSource

The type of the source elements.

TResult

The type of the result elements.

Examples

```
var formatted = rootNodes.RecursiveSelect(  
    node => node.Children,  
    (node, index, depth) => new { node.Name, index, depth });
```

Exceptions

[ArgumentNullException](#)

Thrown if `source`, `childSelector`, or `selector` is null.

RecursiveSelect<TSource, TResult>(IEnumerable<TSource>, Func<TSource, IEnumerable<TSource>>, Func<TSource, int, int, TResult>, Func<TSource, RecursiveSelectControl>)

Recursively flattens and transforms a hierarchical sequence with depth/index tracking and a predicate to control recursion.

```
public static IEnumerable<TResult> RecursiveSelect<TSource, TResult>(this  
    IEnumerable<TSource> source, Func<TSource, IEnumerable<TSource>> childSelector,
```

```
Func<TSource, int, TResult> selector, Func<TSource, RecursiveSelectControl>
recursionControl)
```

Parameters

source [IEnumerable](#)<TSource>

The root sequence to begin recursion from.

childSelector [Func](#)<TSource, [IEnumerable](#)<TSource>>

A function that returns child elements for a given element.

selector [Func](#)<TSource, [int](#), [int](#), TResult>

A transform function applied to each element with index and depth.

recursionControl [Func](#)<TSource, [RecursiveSelectControl](#)>

A predicate that determines whether to continue recursion for a given element.

Returns

[IEnumerable](#)<TResult>

A flattened and projected sequence of results, including only elements for which recursion is allowed.

Type Parameters

TSource

The type of the source elements.

TResult

The type of the result elements.

Examples

```
var filtered = rootNodes.RecursiveSelect(
node => node.Children,
(node, index, depth) => node.Name,
node => node.Children.Count > 0);
```

Exceptions

[ArgumentNullException](#)

Thrown if `source`, `childSelector`, `selector`, or `recursionControl` is null.

RecursiveSelect<TSource, TResult>(IEnumerable<TSource>, Func<TSource, IEnumerable<TSource>>, Func<TSource, int, TResult>)

Recursively flattens and transforms a hierarchical sequence with index using the provided child selector.

```
public static IEnumerable<TResult> RecursiveSelect<TSource, TResult>(this  
IEnumerable<TSource> source, Func<TSource, IEnumerable<TSource>> childSelector,  
Func<TSource, int, TResult> selector)
```

Parameters

`source` [IEnumerable](#)<TSource>

The root sequence to begin recursion from.

`childSelector` [Func](#)<TSource, [IEnumerable](#)<TSource>>

A function that returns child elements for a given element.

`selector` [Func](#)<TSource, [int](#), TResult>

A function applied to each element with its index.

Returns

[IEnumerable](#)<TResult>

A flattened and projected sequence of results from all elements including children.

Type Parameters

`TSource`

The type of the source elements.

TResult

The type of the result elements.

Examples

```
var indexedNames = rootNodes.RecursiveSelect(  
    node => node.Children,  
    (node, index) => ${index}: {node.Name}\");
```

Exceptions

[ArgumentNullException](#)

Thrown if `source`, `childSelector`, or `selector` is null.

RecursiveSelect<TSource, TResult>(IEnumerable<TSource>, Func<TSource, IEnumerable<TSource>>, Func<TSource, TResult>)

Recursively flattens and transforms a hierarchical sequence using the provided child selector and projection.

```
public static IEnumerable<TResult> RecursiveSelect<TSource, TResult>(this  
    IEnumerable<TSource> source, Func<TSource, IEnumerable<TSource>> childSelector,  
    Func<TSource, TResult> selector)
```

Parameters

`source` [IEnumerable](#)<TSource>

The root sequence to begin recursion from.

`childSelector` [Func](#)<TSource, [IEnumerable](#)<TSource>>

A function that returns child elements for a given element.

`selector` [Func](#)<TSource, TResult>

A transform function applied to each element.

Returns

[IEnumerable](#) <TResult>

A flattened and projected sequence of results from all elements including children.

Type Parameters

TSource

The type of the source elements.

TResult

The type of the projected result elements.

Examples

```
var names = rootNodes.RecursiveSelect(  
    node => node.Children,  
    node => node.Name);
```

Exceptions

[ArgumentNullException](#)

Thrown if `source`, `childSelector`, or `selector` is null.

Enum RandomizationMode

Namespace: [Bodu.Collections.Generic.Extensions](#)

Assembly: Bodu.CoreLib.dll

Specifies the available strategies for handling sequences during randomization.

```
public enum RandomizationMode
```

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#)

Fields

BufferAll = 0

Buffers all elements and applies in-place shuffling.

LazyShuffle = 3

Lazily builds and shuffles a subset of items.

ReservoirSample = 1

Selects elements using reservoir sampling without full buffering.

StreamWindowed = 2

Uses a sliding window to shuffle while streaming.

Remarks

Use [RandomizationMode](#) values to control how the source sequence is processed and randomized.

Class SystemRandomAdapter

Namespace: [Bodu.Collections.Generic.Extensions](#)

Assembly: Bodu.CoreLib.dll

Wraps [Random](#) as an [IRandomGenerator](#) implementation.

```
public sealed class SystemRandomAdapter : IRandomGenerator
```

Inheritance

[object](#) ← SystemRandomAdapter

Implements

[IRandomGenerator](#)

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#)

Constructors

SystemRandomAdapter()

Initializes a new adapter with a default [Random](#).

```
public SystemRandomAdapter()
```

SystemRandomAdapter(Random)

Initializes a new adapter using the specified `random` instance.

```
public SystemRandomAdapter(Random random)
```

Parameters

Methods

Next(int)

Returns a non-negative random integer that is less than the specified maximum.

```
public int Next(int maxValue)
```

Parameters

maxValue [int](#)

The exclusive upper bound.

Returns

[int](#)

A random integer in the range [0, maxValue].

Namespace Bodu.Extensions

Classes

[ArrayExtensions](#)

[DateOnlyExtensions](#)

Provides a set of [static](#) ([Shared](#) in Visual Basic) methods that extend the [DateOnly](#) class.

[DateTimeExtensions](#)

Provides a set of [static](#) ([Shared](#) in Visual Basic) methods that extend the [DateTime](#) class.

[NumericExtensions](#)

Interfaces

[IQuarterProvider](#)

Provides custom quarter calculation logic for a [DateTime](#).

[IWeekendProvider](#)

Defines a provider interface for determining whether a given [DayOfWeek](#) is considered a weekend day.

Enums

[DateTimeResolution](#)

Specifies the resolution level for a [DateTime](#) value.

[QuarterDefinition](#)

Specifies how quarters (Q1–Q4) are defined for a given calendar or financial year system.

[StandardWeekend](#)

Represents common weekend configurations used across global regions and cultures.

[WeekOfMonthOrdinal](#)

Specifies the ordinal position of a weekday within a given month. Commonly used in recurrence rules, such as "the second Tuesday of the month."

Class ArrayExtensions

Namespace: [Bodu.Extensions](#)

Assembly: Bodu.CoreLib.dll

```
public static class ArrayExtensions
```

Inheritance

[object](#) ← ArrayExtensions

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

Clear(Array)

Sets all elements in an [Array](#) to the default value of each element type.

```
public static void Clear(this Array array)
```

Parameters

array [Array](#)

The array whose elements to clear.

Remarks

This method supports only single-dimensional, zero-based arrays.

Exceptions

[ArgumentNullException](#)

array is [null](#).

[ArgumentException](#)

`array` is not a single-dimensional array.

-or-

`array` does not have a zero-based index.

Clear(Array, int)

Sets all elements in an [Array](#) to the default value of each element type, starting from the specified index to the end of the array.

```
public static void Clear(this Array array, int index)
```

Parameters

`array` [Array](#)

The array whose elements to clear.

`index` [int](#)

The zero-based index at which to begin clearing elements.

Remarks

Clears elements from `index` to the end of the array.

Exceptions

[ArgumentNullException](#)

`array` is [null](#).

[ArgumentException](#)

`array` is not a single-dimensional array.

-or-

`array` does not have a zero-based index.

[ArgumentOutOfRangeException](#)

`index` is less than 0 or greater than the length of `array`.

Clear(Array, int, int)

Sets a specified number of elements in an [Array](#) to the default value of each element type, starting from a given index.

```
public static void Clear(this Array array, int index, int count)
```

Parameters

array [Array](#)

The array whose elements to clear.

index [int](#)

The zero-based index at which to begin clearing elements.

count [int](#)

The number of elements to clear.

Remarks

Clears **count** elements starting at **index**.

Exceptions

[ArgumentNullException](#)

array is [null](#).

[ArgumentException](#)

array is not a single-dimensional array.

-or-

array does not have a zero-based index.

[ArgumentOutOfRangeException](#)

index is less than 0 or greater than the length of **array**.

-or-

count is less than 0 or extends beyond the end of the array.

Clear<T>(T[])

Sets all elements in a one-dimensional array to the default value of the element type.

```
public static void Clear<T>(this T[] array)
```

Parameters

array T[]

The one-dimensional array to clear.

Type Parameters

T

The type of the elements of the array.

Exceptions

[ArgumentNullException](#)

array is [null](#).

Clear<T>(T[], int)

Sets all elements in a one-dimensional array to the default value of the element type, starting from the specified index to the end of the array.

```
public static void Clear<T>(this T[] array, int index)
```

Parameters

array T[]

The one-dimensional array to clear.

index [int](#)

The zero-based index at which to begin clearing elements.

Type Parameters

T

The type of the elements of the array.

Remarks

This method clears elements starting from `index` to the end of the array. The number of elements cleared is `array.Length - index`.

Exceptions

[ArgumentNullException](#)

`array` is `null`.

[ArgumentOutOfRangeException](#)

`index` is less than 0 or greater than the length of `array`.

Clear<T>(T[], int, int)

Sets a specified number of elements in a one-dimensional array to the default value of the element type, starting from a given index.

```
public static void Clear<T>(this T[] array, int index, int count)
```

Parameters

`array` T[]

The one-dimensional array to clear.

`index` int

The zero-based index at which to begin clearing elements.

`count` int

The number of elements to clear.

Type Parameters

T

The type of the elements of the array.

Remarks

This method clears exactly `count` elements starting at `index`.

Exceptions

[ArgumentNullException](#)

`array` is `null`.

[ArgumentOutOfRangeException](#)

`index` is less than 0 or greater than the length of `array`.

-or-

`count` is less than 0 or extends beyond the end of the array.

Copy<T>(T[])

Creates a new array and copies all elements from the source array to the new array.

```
public static T[] Copy<T>(this T[] array)
```

Parameters

`array` T[]

The source array containing the elements to copy.

Returns

T[]

A new array containing all elements from the `array`.

Type Parameters

T

The type of the elements in the array.

Exceptions

[ArgumentNullException](#)

Thrown when `array` is [null](#).

SliceInternal<T>(T[], int, int)

Takes a slice of the specified array starting from a given index and extending for a specified number of elements. This method does not perform any validation checks (use with caution).

```
public static T[] SliceInternal<T>(this T[] array, int index, int count)
```

Parameters

array T[]

The one-dimensional array that contains the values to copy.

index int

The starting index in the `array` from which copying begins.

count int

The number of elements to copy from the `array`.

Returns

T[]

A new array containing the copied elements from the `array` starting from `index`.

Type Parameters

T

The type of the elements in the array.

Remarks

This method is optimized and does not perform validation. Ensure the inputs are valid.

`Slice<T>(T[], int)`

Takes a slice of the specified array starting from a given index and extending to the end of the array.

```
public static T[] Slice<T>(this T[] array, int index)
```

Parameters

`array` `T[]`

The source array containing the elements to slice.

`index` `int`

The starting index in the source array from which the slice begins.

Returns

`T[]`

A new array containing elements from the `array` starting from `index` to the end.

Type Parameters

`T`

The type of the elements in the array.

Exceptions

[ArgumentNullException](#)

Thrown when `array` is `null`.

[ArgumentOutOfRangeException](#)

Thrown when `index` is out of bounds.

Slice<T>(T[], int, int)

Takes a slice of the specified array starting from a given index and extending for a specified number of elements.

```
public static T[] Slice<T>(this T[] array, int index, int count)
```

Parameters

array `T[]`

The source array containing the elements to slice.

index `int`

The starting index in the source array from which the slice begins.

count `int`

The number of elements to include in the slice.

Returns

`T[]`

A new array containing the sliced elements from the **array** starting at **index**.

Type Parameters

T

The type of the elements in the array.

Exceptions

[ArgumentNullException](#)

Thrown when **array** is `null`.

[ArgumentOutOfRangeException](#)

Thrown when **index** or **count** is less than zero.

[ArgumentException](#)

Thrown when `index` and `count` describe an invalid range within `array`.

Class DateOnlyExtensions

Namespace: [Bodu.Extensions](#)

Assembly: Bodu.CoreLib.dll

Provides a set of [static](#) ([Shared](#) in Visual Basic) methods that extend the [DateOnly](#) class.

```
public static class DateOnlyExtensions
```

Inheritance

[object](#) ← DateOnlyExtensions

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Class DateTimeExtensions

Namespace: [Bodu.Extensions](#)

Assembly: Bodu.CoreLib.dll

Provides a set of [static](#) ([Shared](#) in Visual Basic) methods that extend the [DateTime](#) class.

```
public static class DateTimeExtensions
```

Inheritance

[object](#) ← DateTimeExtensions

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

Add(DateTime, int, int, double)

Adds the specified number of years, months, and fractional days to the given [DateTime](#).

```
public static DateTime Add(this DateTime date, int years, int months, double days)
```

Parameters

date [DateTime](#)

The starting [DateTime](#) value to which the offsets will be applied.

years [int](#)

The number of calendar years to add. A negative value subtracts years.

months [int](#)

The number of calendar months to add. A negative value subtracts months.

days [double](#)

The number of days (including fractional values) to add. A negative value subtracts days.

Returns

[DateTime](#)

A new [DateTime](#) adjusted by the specified number of years, months, and days, preserving the original time-of-day and [Kind](#).

Remarks

The adjustments are applied in the following order: years, then months, then days. This method respects calendar boundaries, leap years (e.g., Feb 29), and automatically normalizes the date when overflowing into the next month (e.g., adding 1 month to January 31 results in the last valid day of February).

Age(DateTime)

Calculates the age in full years between the specified [DateTime](#) and today.

```
public static int Age(this DateTime dateTime)
```

Parameters

[dateTime](#) [DateTime](#)

The date to calculate the age from, typically a date of birth or historical reference.

Returns

[int](#)

The number of full calendar years between [dateTime](#) and [Today](#).

Remarks

If [dateTime](#) is February 29 in a leap year and the current year is not a leap year, February 28 is used as the comparison point.

Age(DateTime, DateTime)

Calculates the age in full years between two [DateTime](#) values.

```
public static int Age(this DateTime firstDate, DateTime secondDate)
```

Parameters

firstDate [DateTime](#)

The earlier date to calculate from, typically a date of birth or historical reference.

secondDate [DateTime](#)

The later date to calculate to. This is usually the current date or another reference point.

Returns

[int](#)

The number of full calendar years that have elapsed between **firstDate** and **secondDate**.

Remarks

If **firstDate** is February 29 in a leap year and **secondDate** occurs in a non-leap year, the age is calculated based on February 28.

DaysInMonth(DateTime)

Returns the number of days in the month of the specified [DateTime](#), using the Gregorian calendar.

```
public static int DaysInMonth(this DateTime dateTime)
```

Parameters

dateTime [DateTime](#)

The [DateTime](#) whose month and year are used to determine the number of days.

Returns

[int](#)

The total number of days in the month for the given `dateTime`, based on the [GregorianCalendar](#).

Remarks

This method always evaluates the number of days using the proleptic Gregorian calendar, regardless of the current culture or calendar settings.

DaysInYear(DateTime)

Returns the number of days in the calendar year of the specified [DateTime](#), using the current culture's calendar.

```
public static int DaysInYear(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The [DateTime](#) for which to calculate the number of days in its calendar year.

Returns

[int](#)

The total number of days in the year corresponding to `dateTime`, as defined by the current culture's [Calendar](#).

Remarks

This method uses the calendar defined by [CurrentCulture](#). The result may vary depending on the calendar system (e.g., Gregorian, Hebrew, Hijri).

DaysInYear(DateTime, Calendar)

Returns the number of days in the calendar year of the specified [DateTime](#), using the specified [Calendar](#).

```
public static int DaysInYear(this DateTime dateTime, Calendar calendar)
```

Parameters

`dateTime` [DateTime](#)

The [DateTime](#) whose year is evaluated.

`calendar` [Calendar](#)

The [Calendar](#) used to determine the number of days in the year. If `null`, the current culture's calendar is used.

Returns

`int`

The number of days in the year of `dateTime`, based on the provided or fallback calendar.

Remarks

Use this method when you want to explicitly calculate based on a specific calendar system (e.g., [GregorianCalendar](#), [HebrewCalendar](#)). If `calendar` is `null`, the calendar from [CurrentCulture](#) is used.

EndOfDay(DateTime)

Returns a new [DateTime](#) representing the last possible tick of the same day as the specified `dateTime`.

```
public static DateTime EndOfDay(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The original [DateTime](#) whose date is used to determine the end of the day.

Returns

[DateTime](#)

A [DateTime](#) value set to 23:59:59.999999 on the same calendar day as `dateTime`, with the original [Kind](#) preserved.

Remarks

The result represents the maximum possible time value for the day, just before midnight of the following day. The [Kind](#) of the input is retained in the result.

FirstDateOfIsoWeek(int, int)

Returns the first date of the specified ISO 8601 week and year.

```
public static DateTime FirstDateOfIsoWeek(int isoYear, int isoWeek)
```

Parameters

isoYear [int](#)

The ISO 8601 year (e.g., 2024).

isoWeek [int](#)

The ISO 8601 week number (1–53).

Returns

[DateTime](#)

A [DateTime](#) value representing the Monday that begins the given ISO 8601 week.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if **isoWeek** or **isoYear** are not valid ISO 8601 values.

FirstDayOfMonth(DateTime)

Returns a new [DateTime](#) representing the first day of the same month and year as the input.

```
public static DateTime FirstDayOfMonth(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The input [DateTime](#) whose month and year are used.

Returns

[DateTime](#)

A [DateTime](#) set to the first day of the month at midnight (00:00:00), with the same [Kind](#) as the input.

Remarks

The time component is normalized to midnight (00:00:00), and the [Kind](#) is preserved.

FirstDayOfQuarter(DateTime)

Returns a [DateTime](#) representing the first day of the calendar quarter for the specified `dateTime`.

```
public static DateTime FirstDayOfQuarter(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The [DateTime](#) whose quarter is evaluated.

Returns

[DateTime](#)

A [DateTime](#) value representing the first day of the quarter, with the time component set to midnight (00:00:00) and the original [Kind](#) preserved.

Remarks

This method uses the standard calendar quarter definition:

- Q1 - January–March
- Q2 - April–June
- Q3 - July–September
- Q4 - October–December

FirstDayOfQuarter(DateTime, IQuarterProvider)

Returns the first day of the quarter based on a custom [IQuarterProvider](#) implementation.

```
public static DateTime FirstDayOfQuarter(this DateTime dateTIme, IQuarterProvider provider)
```

Parameters

dateTime [DateTime](#)

The [DateTime](#) value whose quarter is being evaluated.

provider [IQuarterProvider](#)

The [IQuarterProvider](#) that defines custom quarter logic.

Returns

[DateTime](#)

A [DateTime](#) representing the first calendar day of the applicable custom quarter, with the time set to midnight (00:00:00) and the [Kind](#) preserved.

Remarks

This method supports advanced or domain-specific definitions of quarters by delegating logic to the specified **provider**, such as 4-4-5 fiscal calendars or regional fiscal systems.

Exceptions

[ArgumentNullException](#)

Thrown if **provider** is [null](#).

[ArgumentOutOfRangeException](#)

Thrown if the **provider** returns an invalid quarter number or month (outside the range 1–12).

FirstDayOfQuarter(DateTime, QuarterDefinition)

Returns the first day of the quarter for the given **dateTime**, using the specified [QuarterDefinition](#).

```
public static DateTime FirstDayOfQuarter(this DateTime dateTime,  
QuarterDefinition definition)
```

Parameters

[dateTime DateTime](#)

The [DateTime](#) whose quarter is evaluated.

[definition QuarterDefinition](#)

The [QuarterDefinition](#) that defines the quarter structure to use.

Returns

[DateTime](#)

A [DateTime](#) representing the first day of the identified quarter, with the time component set to 00:00:00 and the original [Kind](#) preserved.

Remarks

This overload supports different quarter definitions such as the Calendar Year and various Financial Year systems, as defined by the [QuarterDefinition](#) enumeration.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if [definition](#) is not a defined value of the [QuarterDefinition](#) enumeration.

FirstDayOfWeek(DateTime)

Returns the first day of the week that contains the specified [DateTime](#), using the current culture's settings.

```
public static DateTime FirstDayOfWeek(this DateTime dateTime)
```

Parameters

dateTime [DateTime](#)

The input [DateTime](#) for which to determine the start of the week.

Returns

[DateTime](#)

A [DateTime](#) set to midnight (00:00:00) on the culturally defined first day of the week that includes `dateTime`. The result preserves the original [Kind](#).

Remarks

This method uses [CurrentCulture](#) to determine the start of the week (via [FirstDayOfWeek](#)). The result is normalized to midnight and excludes any time component.

FirstDayOfWeek(DateTime, CultureInfo)

Returns the first day of the week that contains the specified [DateTime](#), using the provided or current culture.

```
public static DateTime FirstDayOfWeek(this DateTime dateTime, CultureInfo culture)
```

Parameters

dateTime [DateTime](#)

The input [DateTime](#) for which to determine the start of the week.

culture [CultureInfo](#)

An optional [CultureInfo](#) that defines the starting day of the week via [FirstDayOfWeek](#). If `null`, [CurrentCulture](#) is used.

Returns

[DateTime](#)

A [DateTime](#) set to midnight (00:00:00) on the first day of the week that contains `dateTime`. The result preserves the original [Kind](#).

Remarks

This method computes the difference between the current day and the culture-defined first day of the week, then subtracts that number of days from `dateTime` and resets the time to midnight.

FirstDayOfYear(DateTime)

Returns a new [DateTime](#) representing the first day of the same calendar year as the specified `dateTime`.

```
public static DateTime FirstDayOfYear(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The input [DateTime](#) whose year is used to calculate the result.

Returns

[DateTime](#)

A [DateTime](#) set to January 1 of the same year as `dateTime`, with the time component set to 00:00:00 and the [Kind](#) preserved.

Remarks

The method resets the date to January 1 of the year and normalizes the time to midnight (00:00:00). The result always falls within the same calendar year as the original `dateTime`.

FirstWeekdayInMonth(DateTime, DayOfWeek)

Returns the first occurrence of the specified [DayOfWeek](#) within the same month and year as the provided [DateTime](#).

```
public static DateTime FirstWeekdayInMonth(this DateTime dateTime, DayOfWeek dayOfWeek)
```

Parameters

`dateTime` [DateTime](#)

The input [DateTime](#) whose month and year define the search range.

dayOfWeek [DayOfWeek](#)

The [DayOfWeek](#) to locate. For example, [Friday](#) will return the first Friday in the month.

Returns

[DateTime](#)

A new [DateTime](#) representing the first occurrence of the specified [dayOfWeek](#) in the month of [dateTime](#). The result has its time set to 00:00:00 and preserves the [Kind](#) of the input.

Remarks

The search begins from the 1st day of the month, and the result will always fall within the same month and year as [dateTime](#). The time component is reset to midnight (00:00:00), and the [Kind](#) is preserved.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if [dayOfWeek](#) is not a defined value of the [DayOfWeek](#) enumeration.

FirstWeekdayInYear(DateTime, DayOfWeek)

Returns the first occurrence of the specified [DayOfWeek](#) in the calendar year of the given [DateTime](#).

```
public static DateTime FirstWeekdayInYear(this DateTime dateTime, DayOfWeek dayOfWeek)
```

Parameters

[dateTime](#) [DateTime](#)

The input [DateTime](#) whose year is used as the search range.

dayOfWeek [DayOfWeek](#)

The [DayOfWeek](#) to locate. For example, [Monday](#) will return the first Monday of the year.

Returns

[DateTime](#)

A new [DateTime](#) set to the first occurrence of the specified weekday in the same year as `dateTime`, with the time component set to 00:00:00 and the [Kind](#) preserved.

Remarks

The returned [DateTime](#) is always within the same calendar year as `dateTime`, starting from January 1. The time component is normalized to 00:00:00, and the [Kind](#) is retained from the original `dateTime`.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `dayOfWeek` is not a defined value of the [DayOfWeek](#) enumeration.

FromUnixTimeMilliseconds(long)

Converts a Unix timestamp expressed in milliseconds since 1970-01-01T00:00:00Z to a [DateTime](#).

```
public static DateTime FromUnixTimeMilliseconds(this long dateTime)
```

Parameters

`dateTime` [long](#)

The number of milliseconds elapsed since the Unix epoch (1970-01-01T00:00:00Z).

Returns

[DateTime](#)

A [DateTime](#) in UTC that represents the specified Unix timestamp.

Remarks

The returned [DateTime](#) has [Utc](#).

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `dateTime` falls outside the supported Unix timestamp range.

FromUnixTimeSeconds(long)

Converts a Unix timestamp expressed in seconds since 1970-01-01T00:00:00Z to a [DateTime](#).

```
public static DateTime FromUnixTimeSeconds(this long dateTime)
```

Parameters

`dateTime` [long](#)

The number of seconds elapsed since the Unix epoch (1970-01-01T00:00:00Z).

Returns

[DateTime](#)

A [DateTime](#) in UTC that represents the specified Unix timestamp.

Remarks

The returned [DateTime](#) has [Utc](#).

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `dateTime` falls outside the supported Unix timestamp range.

GetElapsedTimeSince(DateTime)

Returns the amount of time that has elapsed between the current system time and the specified [DateTime](#).

```
public static TimeSpan GetElapsedTimeSince(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The [DateTime](#) to compare against the current time.

Returns

[TimeSpan](#)

A [TimeSpan](#) representing the duration from `dateTime` to [Now](#). The value is positive if the input is in the past, and negative if it is in the future.

Remarks

This method subtracts the specified `dateTime` from [Now](#), producing a positive result for past values and a negative result for future values. The comparison uses the system's current local time at the moment of evaluation.

GetFirstDateOfWeek(int, int, StandardWeekend)

Returns the first day of the specified week in a given year, using the [StandardWeekend](#) definition to infer the start of the week.

```
public static DateTime GetFirstDateOfWeek(int year, int week, StandardWeekend weekend)
```

Parameters

`year` [int](#)

The target year (e.g., 2024).

`week` [int](#)

The 1-based week number (e.g., 1–53).

`weekend` [StandardWeekend](#)

The [StandardWeekend](#) that determines how the week is structured.

Returns

[DateTime](#)

A [DateTime](#) value representing the first day of the specified week, with the time component set to 00:00:00.

Remarks

This method does not validate ISO 8601 compliance or locale-specific week rules. It is intended for systems using custom week logic.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `weekend` is not a defined [StandardWeekend](#) value.

GetFirstDateOfWeek(int, int, CultureInfo?)

Returns the [DateTime](#) representing the first day of the specified week number in a given year, using the [CalendarWeekRule](#) and [DayOfWeek](#) settings from the specified [CultureInfo](#).

```
public static DateTime GetFirstDateOfWeek(int year, int week, CultureInfo? culture = null)
```

Parameters

`year` [int](#)

The target year (e.g., 2024).

`week` [int](#)

The 1-based week number (e.g., 1–53).

`culture` [CultureInfo](#)

Optional [CultureInfo](#) used to determine the calendar week rule and start of the week. If `null`, the current thread's culture is used.

Returns

[DateTime](#)

A [DateTime](#) value representing the first day of the specified week, with the time component set to 00:00:00.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if the `week` does not correspond to a valid week number for the specified year and culture.

GetIso8601WeekOfYear(DateTime)

Gets the ISO 8601 week number of the year.

```
public static int GetIso8601WeekOfYear(this DateTime date)
```

Parameters

`dateTime` [DateTime](#)

Returns

[int](#)

GetIso8601Year(DateTime)

Returns the ISO 8601 year associated with the specified [DateTime](#).

```
public static int GetIso8601Year(this DateTime date)
```

Parameters

`date` [DateTime](#)

The [DateTime](#) to evaluate.

Returns

[int](#)

The ISO 8601 calendar year that contains the date's week.

Remarks

ISO weeks may belong to the previous or next calendar year depending on where the week falls.

GetStartMonthFromQuarter(QuarterDefinition, int)

Returns the 1-based starting month of the specified quarter, based on the given [QuarterDefinition](#).

```
public static int GetStartMonthFromQuarter(QuarterDefinition definition, int quarter)
```

Parameters

definition [QuarterDefinition](#)

The quarter system definition to apply (e.g., [CalendarYear](#), [FinancialJuly](#)).

quarter [int](#)

The quarter number to evaluate (1 to 4).

Returns

[int](#)

An integer between 1 and 12 representing the starting month of the specified quarter.

Remarks

This method calculates the starting month by reversing the offset of the quarter system specified by the **definition**. For example, in a July-based financial year ([FinancialJuly](#)), Q1 starts in July (month 7), Q2 in October (month 10), etc.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if **definition** is not a valid value of the [QuarterDefinition](#) enum, or if it is [Custom](#). Use an external provider to support custom definitions. Also thrown if **quarter** is not in the range 1 through 4.

IsLeapYear(DateTime)

Determines whether the year of the specified [DateTime](#) is a leap year in the Gregorian calendar.

```
public static bool IsLeapYear(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The input [DateTime](#) whose year is evaluated.

Returns

[bool](#)

[true](#) if the year is a leap year (i.e., contains February 29); otherwise, [false](#).

Remarks

This method follows the rules of the Gregorian calendar, where a leap year occurs every 4 years, except for years that are divisible by 100 but not by 400 (e.g., 2000 was a leap year, but 1900 was not).

IsWeekday(DateTime)

Determines whether the specified [DateTime](#) falls on a weekday using the default [SaturdaySunday](#) rule.

```
public static bool IsWeekday(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The [DateTime](#) to evaluate.

Returns

[bool](#)

[true](#) if the `dateTime` is not Saturday or Sunday; otherwise, [false](#).

Remarks

A weekday is defined as any day not considered a weekend by the default rule.

IsWeekday(DateTime, StandardWeekend, IWeekendProvider?)

Determines whether the specified [DateTime](#) falls on a weekday, based on the provided [Standard Weekend](#) rule.

```
public static bool IsWeekday(this DateTime dateTime, StandardWeekend weekend,
IWeekendProvider? provider = null)
```

Parameters

dateTime [DateTime](#)

The [DateTime](#) to evaluate.

weekend [StandardWeekend](#)

A [StandardWeekend](#) value that defines which days are considered part of the weekend.

provider [IWeekendProvider](#)

An optional [IWeekendProvider](#) used for custom weekend rules.

Returns

[bool](#)

[true](#) if the **dateTime** is considered a weekday under the specified weekend rule; otherwise, [false](#).

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if **weekend** is [Custom](#) but **provider** is null.

IsWeekend(DateTime)

Determines whether the specified [DateTime](#) falls on a weekend using the default [SaturdaySunday](#) rule.

```
public static bool IsWeekend(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The [DateTime](#) to evaluate.

Returns

[bool](#)

[true](#) if the `dateTime` occurs on Saturday or Sunday; otherwise, [false](#).

Remarks

This method assumes the standard Western weekend definition, where weekends are Saturday and Sunday.

IsWeekend(DateTime, StandardWeekend, IWeekendProvider?)

Determines whether the specified [DateTime](#) falls on a weekend, based on the provided [StandardWeekend](#) rule.

```
public static bool IsWeekend(this DateTime dateTime, StandardWeekend weekend,  
IWeekendProvider? provider = null)
```

Parameters

`dateTime` [DateTime](#)

The [DateTime](#) to evaluate.

`weekend` [StandardWeekend](#)

The [StandardWeekend](#) value that defines which days are considered part of the weekend.

`provider` [IWeekendProvider](#)

An optional custom [IWeekendProvider](#) used when `weekend` is [Custom](#).

Returns

[bool](#)

[true](#) if the `dateTime` occurs on a weekend day according to the specified rule or custom provider; otherwise, [false](#).

Remarks

Use this method to evaluate weekend status under different global or cultural definitions, such as Friday/Saturday in the Middle East or Sunday-only in some business contexts.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `weekend` is [Custom](#) but no `provider` is supplied.

IsWeekend(DayOfWeek, StandardWeekend, IWeekendProvider?)

Evaluates whether the specified [DayOfWeek](#) is considered part of the weekend using the given rule or provider.

```
public static bool IsWeekend(DayOfWeek dayOfWeek, StandardWeekend weekend, IWeekendProvider?  
provider = null)
```

Parameters

`dayOfWeek` [DayOfWeek](#)

The [DayOfWeek](#) value to check.

`weekend` [StandardWeekend](#)

The [StandardWeekend](#) rule to apply.

`provider` [IWeekendProvider](#)

A custom [IWeekendProvider](#) used when the rule is [Custom](#).

Returns

[bool](#)

[true](#) if the `dayOfWeek` falls on a weekend day; otherwise, [false](#).

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `weekend` is [Custom](#) and `provider` is null.

LastDayOfMonth(DateTime)

Returns a new [DateTime](#) representing the last day of the same month and year as the specified `dateTime`.

```
public static DateTime LastDayOfMonth(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The input [DateTime](#) whose month and year are used to determine the last day.

Returns

[DateTime](#)

A [DateTime](#) set to midnight (00:00:00) on the last day of the month, with the original [Kind](#) preserved.

Remarks

This method uses the rules of the Gregorian calendar to determine the number of days in the month, including leap year behavior for February. The time component is normalized to midnight (00:00:00).

LastDayOfQuarter(DateTime)

Returns a [DateTime](#) representing the last day of the calendar quarter for the specified `dateTime`.

```
public static DateTime LastDayOfQuarter(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The [DateTime](#) whose quarter is evaluated.

Returns

[DateTime](#)

A [DateTime](#) value representing the last day of the quarter, with the time component normalized to midnight (00:00:00) and the original [Kind](#) preserved.

Remarks

This method uses the standard calendar quarter definition:

- Q1 - January–March
- Q2 - April–June
- Q3 - July–September
- Q4 - October–December

LastDayOfQuarter(DateTime, IQuarterProvider)

Returns the last day of the quarter based on a custom [IQuarterProvider](#) implementation.

```
public static DateTime LastDayOfQuarter(this DateTime dateTime, IQuarterProvider provider)
```

Parameters

`dateTime` [DateTime](#)

The [DateTime](#) value whose quarter is being evaluated.

`provider` [IQuarterProvider](#)

The [IQuarterProvider](#) that defines custom quarter mappings and start months.

Returns

[DateTime](#)

A [DateTime](#) representing the last calendar day of the applicable custom quarter, with the time set to midnight (00:00:00) and the [Kind](#) preserved.

Remarks

This method supports advanced quarter systems such as 4-4-5 accounting or domain-specific fiscal quarters by delegating logic to the specified [provider](#).

Exceptions

[ArgumentNullException](#)

Thrown if [provider](#) is [null](#).

[ArgumentOutOfRangeException](#)

Thrown if the [provider](#) returns an invalid quarter number or month (outside the range 1–12).

LastDayOfQuarter(DateTime, QuarterDefinition)

Returns the last day of the quarter for the given [dateTime](#), using the specified [QuarterDefinition](#).

```
public static DateTime LastDayOfQuarter(this DateTime dateTime,  
QuarterDefinition definition)
```

Parameters

[dateTime](#) [DateTime](#)

The [DateTime](#) whose quarter is evaluated.

[definition](#) [QuarterDefinition](#)

The [QuarterDefinition](#) that defines the quarter structure to use.

Returns

[DateTime](#)

A [DateTime](#) representing the last calendar day of the identified quarter, with the time component set to 00:00:00 and the original [Kind](#) retained.

Remarks

This overload supports different quarter definitions such as the Calendar Year and various Financial Year systems, as defined by the [QuarterDefinition](#) enumeration.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `definition` is not a defined value of the [QuarterDefinition](#) enumeration.

LastDayOfWeek(DateTime)

Returns the last day of the week that contains the specified [DateTime](#), using the current culture's calendar settings.

```
public static DateTime LastDayOfWeek(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The input [DateTime](#) whose week context is evaluated.

Returns

[DateTime](#)

A [DateTime](#) representing the culturally defined last day of the week, with the time component set to midnight (00:00:00) and the original [Kind](#) preserved.

Remarks

This method uses [CurrentCulture](#) to determine the last day of the week. The result is computed by finding the next occurrence of that day from `dateTime` (or the same day if already matched). If the resulting date exceeds the valid range for [DateTime](#), an [ArgumentOutOfRangeException](#) is thrown.

LastDayOfWeek(DateTime, CultureInfo)

Returns the last day of the week that contains the specified [DateTime](#), using the specified [CultureInfo](#).

```
public static DateTime LastDayOfWeek(this DateTime dateTime, CultureInfo culture)
```

Parameters

`dateTime` [DateTime](#)

The [DateTime](#) from which the last day of the week is calculated.

`culture` [CultureInfo](#)

The [CultureInfo](#) that defines the first day of the week via its [DateTimeFormatInfo](#). If `null`, [CurrentCulture](#) is used.

Returns

[DateTime](#)

A [DateTime](#) value set to midnight on the culturally determined last day of the week relative to `dateTime`, preserving its [Kind](#).

Remarks

The last day of the week is determined by calling [FirstDayOfWeek](#) from the provided or current culture. If `dateTime` already falls on the last day, it is returned at midnight. Otherwise, the next occurrence is calculated.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if the resulting [DateTime](#) exceeds the supported [DateTime](#) range.

LastDayOfYear(DateTime)

Returns a new [DateTime](#) representing the last day of the same calendar year as the specified `dateTime`.

```
public static DateTime LastDayOfYear(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The input [DateTime](#) whose year is used to determine the result.

Returns

[DateTime](#)

A [DateTime](#) set to December 31 of the same year as `dateTime`, with the time component set to midnight (00:00:00) and the original [Kind](#) preserved.

Remarks

This method uses the Gregorian calendar and normalizes the time component to 00:00:00 (midnight).

LastWeekdayInMonth(DateTime, DayOfWeek)

Returns the last occurrence of the specified [DayOfWeek](#) within the same month and year as the given `dateTime`.

```
public static DateTime LastWeekdayInMonth(this DateTime dateTime, DayOfWeek dayOfWeek)
```

Parameters

`dateTime` [DateTime](#)

The [DateTime](#) whose month and year define the search range.

`dayOfWeek` [DayOfWeek](#)

The [DayOfWeek](#) value to locate. For example, [Friday](#) will return the last Friday of the month.

Returns

[DateTime](#)

A new [DateTime](#) set to midnight on the last occurrence of the specified `dayOfWeek` in the same month and year as `dateTime`, with the original [Kind](#) preserved.

Remarks

The search starts from the last day of the month and proceeds backward until the specified `dayOfWeek` is found. The result is normalized to midnight (00:00:00), and the original [Kind](#) is retained.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `dayOfWeek` is not a defined value in the [DayOfWeek](#) enumeration.

LastWeekdayInYear(DateTime, DayOfWeek)

Returns the last occurrence of the specified [DayOfWeek](#) within the same calendar year as the given `dateTime`.

```
public static DateTime LastWeekdayInYear(this DateTime dateTime, DayOfWeek dayOfWeek)
```

Parameters

`dateTime` [DateTime](#)

The [DateTime](#) whose year defines the search range.

`dayOfWeek` [DayOfWeek](#)

The [DayOfWeek](#) to locate. For example, [Sunday](#) returns the last Sunday in the year.

Returns

[DateTime](#)

A new [DateTime](#) set to midnight on the last occurrence of the specified `dayOfWeek` in the same year as `dateTime`, preserving the original [Kind](#).

Remarks

The search begins from December 31 of the given year and proceeds backward until the specified `dayOfWeek` is found. The time component is normalized to 00:00:00 (midnight), and the [Kind](#) is preserved.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `dayOfWeek` is not a valid [DayOfWeek](#) enumeration value.

MonthName(DateTime)

Returns the full name of the month for the specified [DateTime](#), using the current culture's formatting rules.

```
public static string MonthName(this DateTime dateTime)
```

Parameters

[dateTime](#) [DateTime](#)

The [DateTime](#) whose month value is used to retrieve the name.

Returns

[string](#)

A localized [string](#) representing the full month name, based on [CurrentCulture](#).

Remarks

This method uses the [MonthNames](#) or the [CurrentCulture](#) to retrieve the localized month name.

MonthName(DateTime, CultureInfo)

Returns the full name of the month for the specified [DateTime](#), using the formatting rules of the given [CultureInfo](#).

```
public static string MonthName(this DateTime dateTime, CultureInfo culture)
```

Parameters

[dateTime](#) [DateTime](#)

The [DateTime](#) whose month value is used to retrieve the name.

[culture](#) [CultureInfo](#)

An optional [CultureInfo](#) that provides culture-specific formatting. If [null](#), [CurrentCulture](#) is used.

Returns

[string](#)

A localized [string](#) representing the full month name based on the provided or current culture.

Remarks

This method uses the [MonthNames](#) collection of the given or current culture to return the localized full name of the month represented by `dateTime.Month.Month`.

NextOccurrence(DateTime, TimeSpan, DateTime)

Calculates the next occurrence of a periodic event based on a given start time and interval, relative to a specified point in time.

```
public static DateTime NextOccurrence(this DateTime start, TimeSpan interval,  
DateTime after)
```

Parameters

`start` [DateTime](#)

The [DateTime](#) representing the initial start time of the recurring event.

`interval` [TimeSpan](#)

A [TimeSpan](#) that defines how often the event recurs. Must be greater than zero.

`after` [DateTime](#)

A [DateTime](#) after which the next occurrence should be found.

Returns

[DateTime](#)

A [DateTime](#) representing the next occurrence of the event after the specified `after` time.

Remarks

The returned [DateTime](#) is aligned with the start time and repeated using the defined interval. For example, with a start of 9:00 AM and a 1-hour interval, the next occurrence after 10:45 AM would be 11:00 AM.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if the `interval` is zero or negative.

NextWeekday(DateTime, DayOfWeek)

Returns a new [DateTime](#) representing the next calendar occurrence of the specified [DayOfWeek](#) after the given `dateTime`.

```
public static DateTime NextWeekday(this DateTime dateTime, DayOfWeek dayOfWeek)
```

Parameters

`dateTime` [DateTime](#)

The starting [DateTime](#) from which to search forward.

`dayOfWeek` [DayOfWeek](#)

The desired [DayOfWeek](#) to locate. For example, [Monday](#) will return the next Monday.

Returns

[DateTime](#)

A [DateTime](#) with the same time-of-day and [Kind](#) as `dateTime`, representing the next occurrence of `dayOfWeek`.

Remarks

If `dateTime` already falls on the specified `dayOfWeek`, the result will be exactly 7 days later (the next calendar occurrence).

The returned value preserves the original time-of-day and [Kind](#) values of `dateTime`.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `dayOfWeek` is not a valid [DayOfWeek](#) enum value.

NthDayOfWeekInMonth(DateTime, DayOfWeek, WeekOfMonthOrdinal)

Returns a new [DateTime](#) that represents the Nth occurrence of the specified day of the week in the month.

```
public static DateTime NthDayOfWeekInMonth(this DateTime dateTime, DayOfWeek dayOfWeek,  
WeekOfMonthOrdinal ordinal)
```

Parameters

dateTime [DateTime](#)

The date providing the month and year context (day component is ignored).

dayOfWeek [DayOfWeek](#)

The day of the week to locate.

ordinal [WeekOfMonthOrdinal](#)

The ordinal occurrence to locate (e.g., [First](#), [Second](#), [Last](#)).

Note: [Fifth](#) is relatively rare and only occurs in months where five instances of the specified weekday exist.

Returns

[DateTime](#)

A [DateTime](#) representing the specified occurrence of the weekday within the same month and year as **dateTime**.

Remarks

For [Last](#), the last matching weekday in the month is returned. For all others, the method computes the Nth occurrence starting from the first of the month.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `dayOfWeek` or `ordinal` is not a defined enum value, or if the specified ordinal does not exist in the given month.

PreviousWeekday(DateTime, DayOfWeek)

Returns a new [DateTime](#) representing the most recent occurrence of the specified [DayOfWeek](#) prior to the given `dateTime`.

```
public static DateTime PreviousWeekday(this DateTime dateTime, DayOfWeek dayOfWeek)
```

Parameters

`dateTime` [DateTime](#)

The starting [DateTime](#) from which to search backward.

`dayOfWeek` [DayOfWeek](#)

The [DayOfWeek](#) to locate. For example, [Friday](#) returns the previous Friday before the given date.

Returns

[DateTime](#)

A [DateTime](#) with the same time-of-day and [Kind](#) as `dateTime`, representing the most recent occurrence of `dayOfWeek` prior to the input.

Remarks

If `dateTime` already falls on the specified `dayOfWeek`, the result will be exactly 7 days earlier (the previous calendar occurrence).

The returned [DateTime](#) preserves the original time-of-day and [Kind](#) values.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `dayOfWeek` is not a defined value of the [DayOfWeek](#) enumeration.

Quarter(DateTime)

Returns the quarter number (1–4) of the year for the specified [DateTime](#), using the standard calendar quarter definition.

```
public static int Quarter(this DateTime dateTime)
```

Parameters

[dateTime](#) [DateTime](#)

The [DateTime](#) value to evaluate.

Returns

[int](#)

An integer between 1 and 4 representing the quarter that contains the [dateTime](#).

Remarks

This method uses the standard calendar quarter structure:

- Q1 - January–March
- Q2 - April–June
- Q3 - July–September
- Q4 - October–December

Quarter(DateTime, IQuarterProvider)

Returns the quarter number (1–4) for the specified [DateTime](#), using a custom [IQuarterProvider](#) implementation.

```
public static int Quarter(this DateTime dateTime, IQuarterProvider provider)
```

Parameters

[dateTime](#) [DateTime](#)

The [DateTime](#) value to evaluate.

provider [IQuarterProvider](#)

An implementation of [IQuarterProvider](#) that determines the quarter based on custom logic.

Returns

[int](#)

An integer between 1 and 4 representing the quarter that includes the specified [dateTime](#).

Remarks

Use this overload to support advanced or non-standard fiscal calendars, such as 4-4-5 financial periods, retail accounting calendars, or region-specific quarter models not covered by [QuarterDefinition](#).

Exceptions

[ArgumentNullException](#)

Thrown if [provider](#) is [null](#).

[ArgumentOutOfRangeException](#)

Thrown if the quarter value returned by the provider is outside the valid range of 1 through 4.

Quarter(DateTime, QuarterDefinition)

Returns the quarter number (1–4) for the specified [DateTime](#), using the given [QuarterDefinition](#) to determine the fiscal calendar structure.

```
public static int Quarter(this DateTime dateTime, QuarterDefinition definition)
```

Parameters

[dateTime](#) [DateTime](#)

The [DateTime](#) value to evaluate.

[definition](#) [QuarterDefinition](#)

The quarter system definition to apply (e.g., [CalendarYear](#), [FinancialJuly](#)).

Returns

[int](#)

An integer between 1 and 4 representing the quarter that includes the specified `dateTime`.

Remarks

This method supports predefined quarter structures aligned to calendar or financial years. The result is based on adjusting the input month by the definition offset and mapping the result to a 1-based quarter.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `definition` is not a valid value of the [QuarterDefinition](#) enum, or if it is [Custom](#). Use the `Quarter(DateTime, IQuarterProvider)` overload to support custom definitions.

StartOfDay(DateTime)

Returns a new [DateTime](#) representing the beginning of the same calendar day as the specified `dateTime`, with the time set to 00:00:00.

```
public static DateTime StartOfDay(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The input [DateTime](#) whose date is preserved while the time component is reset.

Returns

[DateTime](#)

A [DateTime](#) value set to midnight (00:00:00) on the same day as `dateTime`, preserving the original [Kind](#).

Remarks

This method is functionally similar to accessing `dateTime.Date`, but explicitly retains the original [Kind](#).

ToDateOnly(DateTime)

Converts the specified [DateTime](#) to a [DateOnly](#), preserving only the year, month, and day components.

```
public static DateOnly ToDateOnly(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The [DateTime](#) instance to convert.

Returns

[DateOnly](#)

A [DateOnly](#) representing the date component of the input `dateTime`.

Remarks

This method is available on .NET 6.0 or later. The resulting [DateOnly](#) does not retain the time or [Kind](#) information.

ToDateTimeOffset(DateTime)

Converts the specified [DateTime](#) to a [DateTimeOffset](#), interpreting the value based on its [Kind](#).

```
public static DateTimeOffset ToDateTimeOffset(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The input [DateTime](#) to convert.

Returns

[DateTimeOffset](#)

A [DateTimeOffset](#) representing the same point in time as `dateTime`, using its [Kind](#) to determine the offset (local, UTC, or unspecified).

Remarks

This method respects the [Kind](#) value:

- **UTC** - Returns a [DateTimeOffset](#) with zero offset.
- **Local** - Applies the system's local offset.
- **Unspecified** - Assumes the local time zone context.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if the conversion results in a UTC time that is outside the allowable range for [DateTimeOffset](#).

ToDateTimeOffset(DateTime, TimeSpan)

Converts the specified [DateTime](#) to a [DateTimeOffset](#) using the provided UTC offset.

```
public static DateTimeOffset ToDateTimeOffset(this DateTime dateTime, TimeSpan offset)
```

Parameters

`dateTime` [DateTime](#)

The input [DateTime](#) to convert.

`offset` [TimeSpan](#)

The [TimeSpan](#) offset from UTC to associate with the resulting [DateTimeOffset](#).

Returns

[DateTimeOffset](#)

A [DateTimeOffset](#) representing the same clock time as `dateTime` with the specified `offset` applied.

Remarks

Use this overload when the offset is known or must be explicitly applied (e.g., fixed timezone data). The `offset` must be within ±14 hours.

Exceptions

[ArgumentException](#)

Thrown if `offset` is not valid for the specified `dateTime.Kind.Kind`, or if it creates an invalid combination.

[ArgumentOutOfRangeException](#)

Thrown if the resulting UTC time is outside the valid range supported by [DateTimeOffset](#).

ToISOString(DateTime)

Converts the specified [DateTime](#) to an ISO 8601 formatted string, respecting the [Kind](#).

```
public static string ToISOString(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The [DateTime](#) to convert.

Returns

[string](#)

A string representation of `dateTime` in ISO 8601 format:

- If [Kind](#) is [Utc](#), the result ends with 'Z'.
- If [Kind](#) is [Local](#), the result includes the local time zone offset.
- If [Kind](#) is [Unspecified](#), the result omits any time zone information.

Remarks

Uses the "o" (round-trip) format for Local and Unspecified kinds, and "yyyy-MM-ddTHH:mm:ss.fffffffZ" for Utc to enforce ISO 8601 'Z' suffix.

ToISOString(DateTime, bool)

Converts the specified [DateTime](#) to an ISO 8601 formatted string, allowing fractional seconds to be omitted.

```
public static string ToISOString(this DateTime dateTime, bool includeFractionalSeconds)
```

Parameters

dateTime [DateTime](#)

The [DateTime](#) to convert.

includeFractionalSeconds [bool](#)

Whether to include fractional seconds (7 digits) in the result.

Returns

[string](#)

A string representation of **dateTime** in ISO 8601 format, with or without fractional seconds.

Remarks

When **includeFractionalSeconds** is **true**, uses the "o" (round-trip) format; otherwise, uses "yyyy-MM-ddTHH:mm:ss".

ToISOString(DateTime, DateTimeKind)

Converts the specified [DateTime](#) to an ISO 8601 formatted string, using an explicit [DateTimeKind](#).

```
public static string ToISOString(this DateTime dateTime, DateTimeKind kind)
```

Parameters

dateTime [DateTime](#)

The [DateTime](#) to convert.

kind [DateTimeKind](#)

The [DateTimeKind](#) to apply before formatting.

Returns

[string](#)

A string representation of the input in ISO 8601 format, formatted as:

- [Utc](#) → UTC with trailing 'Z'.
- [Local](#) → Local time with offset.
- [Unspecified](#) → No time zone indicator.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `kind` is not a valid [DateTimeKind](#) value.

ToIsoString(DateTime, string, CultureInfo?)

Converts the specified [DateTime](#) to a formatted string using a custom format and optional culture.

```
public static string ToIsoString(this DateTime dateTime, string format, CultureInfo? culture = null)
```

Parameters

`dateTime` [DateTime](#)

The [DateTime](#) to format.

`format` [string](#)

A valid custom format string (e.g., "yyyy-MM-ddTHH:mm:ss").

`culture` [CultureInfo](#)

An optional [CultureInfo](#) to apply. If `null`, [InvariantCulture](#) is used.

Returns

[string](#)

A string formatted using the specified pattern and culture.

Exceptions

[ArgumentNullException](#)

Thrown if `format` is null or empty.

ToMidday(DateTime)

Returns a new [DateTime](#) representing 12:00 PM (noon) on the same calendar day as the specified `dateTime`.

```
public static DateTime ToMidday(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The input [DateTime](#) whose date is used.

Returns

[DateTime](#)

A [DateTime](#) with the same year, month, and day as `dateTime`, and the time component set to 12:00:00.000 (midday). The [Kind](#) is preserved.

Remarks

This method sets the time-of-day component to exactly 12:00 PM (midday), while preserving the original date and [Kind](#).

ToMidnight(DateTime)

Returns a new [DateTime](#) representing midnight (00:00:00) at the start of the same calendar day as the specified `dateTime`.

```
public static DateTime ToMidnight(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The input [DateTime](#) whose date is preserved.

Returns

[DateTime](#)

A [DateTime](#) with the same year, month, and day as `dateTime`, with the time component set to 00:00:00 (midnight). The original [Kind](#) is preserved.

Remarks

This method is functionally equivalent to accessing `dateTime.Date`, but it explicitly retains the original [Kind](#).

To TimeSpan(DateTime)

Returns a new [TimeSpan](#) representing the time-of-day component of the specified [DateTime](#).

```
public static TimeSpan To TimeSpan(this DateTime dateTime)
```

Parameters

`dateTime` [DateTime](#)

The input [DateTime](#) whose time portion will be extracted.

Returns

[TimeSpan](#)

A [TimeSpan](#) containing the hour, minute, second, and fractional second values of `dateTime`.

Remarks

This method returns the time elapsed since midnight for the given date. The result is unaffected by the [Kind](#) property.

ToUnixTimeMilliseconds(DateTime)

Converts the specified [DateTime](#) to the number of milliseconds that have elapsed since the Unix epoch (1970-01-01T00:00:00Z).

```
public static long ToUnixTimeMilliseconds(this DateTime dateTime)
```

Parameters

dateTime [DateTime](#)

The [DateTime](#) to convert. It is converted to UTC before calculation.

Returns

[long](#)

The number of milliseconds since the Unix epoch.

Remarks

This method converts **dateTime** to UTC via [ToUniversalTime\(\)](#) before computing the elapsed time.

ToUnixTimeSeconds(DateTime)

Converts the specified [DateTime](#) to the number of seconds that have elapsed since the Unix epoch (1970-01-01T00:00:00Z).

```
public static long ToUnixTimeSeconds(this DateTime dateTime)
```

Parameters

dateTime [DateTime](#)

The [DateTime](#) to convert. It is converted to UTC before calculation.

Returns

[long](#)

The number of seconds since the Unix epoch.

Remarks

This method converts `dateTime` to UTC via [ToUniversalTime\(\)](#) before computing the elapsed time.

Truncate(DateTime, DateTimeResolution)

Returns a new [DateTime](#) with the value truncated to the specified `resolution`, resetting all smaller time components to zero.

```
public static DateTime Truncate(this DateTime dateTime, DateTimeResolution resolution)
```

Parameters

`dateTime` [DateTime](#)

The [DateTime](#) value to truncate.

`resolution` [DateTimeResolution](#)

The [DateTimeResolution](#) level to which the input should be truncated.

Returns

[DateTime](#)

A [DateTime](#) with smaller time components cleared based on the specified `resolution`.

Remarks

This method returns a new [DateTime](#) with smaller components reset to zero depending on the specified `resolution`. The original [Kind](#) is preserved in the result.

For example, given the input `2024-04-18T14:37:56.7891234`:

Resolution	Truncation Behavior and Result
Year	Truncates to the start of the year: January 1 at 00:00:00. Result: <code>2024-01-01T00:00:00.0000000</code>
Month	Truncates to the start of the month: 1st of April at 00:00:00. Result: <code>2024-04-01T00:00:00.0000000</code>

Resolution	Truncation Behavior and Result
Day	Truncates to the start of the day (midnight). Result: <code>2024-04-18T00:00:00.000000</code>
Hour	Minute, second, and sub-second components set to zero. Result: <code>2024-04-18T14:00:00.000000</code>
Minute	Second and sub-second components set to zero. Result: <code>2024-04-18T14:37:00.000000</code>
Second	Sub-second (millisecond and ticks) components set to zero. Result: <code>2024-04-18T14:37:56.000000</code>
Millisecond	Ticks below the millisecond level set to zero. Result: <code>2024-04-18T14:37:56.789000</code>
Tick	No truncation applied; the original value is returned. Result: <code>2024-04-18T14:37:56.789123</code>

Exceptions

[ArgumentException](#)

Thrown if `resolution` is not a defined value of the [DateTimeResolution](#) enumeration.

WeekOfMonth(DateTime)

Returns the 1-based week number of the month for the specified [DateTime](#), using the current culture's calendar week rule and first day of the week.

```
public static int WeekOfMonth(this DateTime date)
```

Parameters

`date` [DateTime](#)

The date to evaluate.

Returns

[int](#)

The 1-based week number of the month for the specified date.

Remarks

This method uses the [CurrentCulture](#) settings to determine the calendar week rule and the first day of the week. The calculation is based on the difference between the week number of the target date and the week number of the first day of the month, plus one.

WeekOfMonth(DateTime, CalendarWeekRule, DayOfWeek)

Returns the 1-based week number of the month for the specified [DateTime](#), using the specified [CalendarWeekRule](#) and [DayOfWeek](#).

```
public static int WeekOfMonth(this DateTime date, CalendarWeekRule weekRule,  
DayOfWeek weekStart)
```

Parameters

date [DateTime](#)

The date to evaluate.

weekRule [CalendarWeekRule](#)

The calendar week rule used to determine the first week of the month.

weekStart [DayOfWeek](#)

The day considered the start of the week.

Returns

[int](#)

The 1-based week number of the month for the specified date.

Remarks

The week number is calculated as the difference between the week of year for the given date and the week of year for the first day of the same month, plus one. This ensures that the result is always 1-based

and reflects the position of the week within the calendar month.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `weekRule` or `weekStart` is not a defined enum value.

WeekOfMonth(DateTime, CultureInfo)

Returns the 1-based week number of the month for the specified [DateTime](#), using the calendar week rule and first day of the week defined by the specified [CultureInfo](#).

```
public static int WeekOfMonth(this DateTime date, CultureInfo culture)
```

Parameters

`date` [DateTime](#)

The date to evaluate.

`culture` [CultureInfo](#)

The culture providing the calendar rule and first day of the week. If `null`, the current thread culture is used.

Returns

[int](#)

The 1-based week number of the month for the specified date.

Remarks

This method uses the specified culture's [CalendarWeekRule](#) and [FirstDayOfWeek](#) to compute the week number of the date relative to the first day of the same month.

WeekOfYear(DateTime)

Returns the week number (1–53) of the year that contains the specified [DateTime](#), using the current culture's calendar and week rule.

```
public static int WeekOfYear(this DateTime dateTime)
```

Parameters

dateTime [DateTime](#)

The [DateTime](#) to evaluate.

Returns

[int](#)

The culture-specific week number of the year that contains **dateTime**.

Remarks

This method uses the [CalendarWeekRule](#) and [DayOfWeek](#) settings of [CurrentCulture](#). Results may vary between cultures (e.g., the U.S. and ISO week numbers differ).

WeekOfYear(DateTime, CultureInfo?)

Returns the week number (1–53) of the year that contains the specified [DateTime](#), using the given **culture**.

```
public static int WeekOfYear(this DateTime dateTime, CultureInfo? culture)
```

Parameters

dateTime [DateTime](#)

The [DateTime](#) to evaluate.

culture [CultureInfo](#)

The [CultureInfo](#) used to determine the calendar system, week rule, and first day of the week. If **null**, [CurrentCulture](#) is used.

Returns

[int](#)

The week number of the year according to the specified culture's calendar and formatting rules.

Remarks

This method uses [Calendar](#), [CalendarWeekRule](#), and [FirstDayOfWeek](#) from the specified culture to compute the result.

WeekdayOrdinal(DateTime)

Returns the ordinal occurrence of the weekday (e.g., first, second, third) for the specified [DateTime](#) within its month.

```
public static WeekOfMonthOrdinal WeekdayOrdinal(this DateTime dateTime)
```

Parameters

[dateTime](#) [DateTime](#)

The [DateTime](#) to evaluate.

Returns

[WeekOfMonthOrdinal](#)

A [WeekOfMonthOrdinal](#) value representing which occurrence of the [DayOfWeek](#) the given date is, within the same calendar month (e.g., the 2nd Tuesday).

Remarks

The method calculates how many full weeks have passed since the first day of the month to determine the ordinal position of the weekday for the given [dateTime](#).

Note: The [Fifth](#) value is uncommon and only applies to months that contain five occurrences of the specified [DayOfWeek](#).

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if the computed ordinal value exceeds the defined values in [WeekOfMonthOrdinal](#).

Enum DateTimeResolution

Namespace: [Bodu.Extensions](#)

Assembly: Bodu.CoreLib.dll

Specifies the resolution level for a [DateTime](#) value.

```
public enum DateTimeResolution
```

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#)

Fields

Day = 2

Indicates resolution to a whole day.

Hour = 3

Indicates resolution to a whole hours.

Millisecond = 6

Indicates resolution to a whole millisecond.

Minute = 4

Indicates resolution to a whole minute.

Month = 1

Indicates resolution to a whole month.

Second = 5

Indicates resolution to a whole second.

Tick = 7

Indicates resolution to a whole tick.

Year = 0

Indicates resolution to a whole year.

Interface IQuarterProvider

Namespace: [Bodu.Extensions](#)

Assembly: Bodu.CoreLib.dll

Provides custom quarter calculation logic for a [DateTime](#).

```
public interface IQuarterProvider
```

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#)

Remarks

Implement this interface to support [Custom](#) in [Quarter\(DateTime, QuarterDefinition\)](#) and related methods like [FirstDayOfQuarter\(DateTime, IQuarterProvider\)](#). This allows support for non-standard or domain-specific quarterly definitions such as 4-4-5 calendars or region-specific fiscal models.

Methods

GetQuarter(DateTime)

Gets the quarter number (1–4) for the specified [dateTime](#).

```
int GetQuarter(DateTime dateTime)
```

Parameters

[dateTime](#) [DateTime](#)

The input [DateTime](#) for which the quarter is to be determined.

Returns

[int](#)

An integer in the range 1 to 4 representing the quarter that contains the given [dateTime](#).

Remarks

The returned value must be in the range 1 to 4. Implementations should throw an exception for unsupported values or ambiguous ranges.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if the input `dateTime` cannot be mapped to a valid quarter.

GetStartMonthFromQuarter(int)

Gets the starting month number (1–12) corresponding to the specified `quarter`.

```
int GetStartMonthFromQuarter(int quarter)
```

Parameters

`quarter` [int](#)

The quarter number for which to retrieve the first month (must be 1–4).

Returns

[int](#)

An integer representing the first month of the specified quarter. For example, a return value of 4 corresponds to April.

Remarks

This method is used to compute the start of a quarter and must always return a valid month in the range 1–12.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `quarter` is not between 1 and 4.

Interface IWeekendProvider

Namespace: [Bodu.Extensions](#)

Assembly: Bodu.CoreLib.dll

Defines a provider interface for determining whether a given [DayOfWeek](#) is considered a weekend day.

```
public interface IWeekendProvider
```

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#)

Remarks

Implement this interface to supply custom weekend definitions that are not covered by built-in [Standard Weekend](#) options.

Methods

IsWeekend(DayOfWeek)

Determines whether the specified [DayOfWeek](#) is considered a weekend day.

```
bool IsWeekend(DayOfWeek dayOfWeek)
```

Parameters

dayOfWeek [DayOfWeek](#)

The day of the week to evaluate.

Returns

[bool](#)

[true](#) if the day is considered part of the weekend; otherwise, [false](#).

Class NumericExtensions

Namespace: [Bodu.Extensions](#)

Assembly: Bodu.CoreLib.dll

```
public static class NumericExtensions
```

Inheritance

[object](#) ← NumericExtensions

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

GetBytes<T>(T, bool)

Converts the specified numeric value to a byte array. Uses bitwise shifts for numeric types (e.g., [int](#), [long](#), [ushort](#), etc.) to generate the byte representation, and built-in methods like See [BitConverter](#) for methods that convert numeric types (e.g., [float](#), [double](#)) to byte arrays.

```
public static byte[] GetBytes<T>(this T value, bool asBigEndian = false)
```

Parameters

value [T](#)

The value to convert to a byte array. The value must be of a supported numeric type, such as [byte](#), [short](#), [int](#), [long](#), [ushort](#), [uint](#), [ulong](#), [float](#), or [double](#).

asBigEndian [bool](#)

If [true](#), forces the byte array to be in big-endian order; otherwise, [false](#), the array will be in the system's native endianness.

Returns

[byte\[\]](#)

A byte array representing the value in the specified or native endianness.

Type Parameters

T

The type of the value to convert. This should be a numeric type or a type supported by [BitConverter](#).

Exceptions

[InvalidOperationException](#)

Thrown if T is not a supported type (numeric or BitConverter-supported).

ReverseBits(byte)

Reverses the order of the bits of an unsigned [byte](#) value.

```
public static byte ReverseBits(this byte value)
```

Parameters

value [byte](#)

The unsigned byte value to reverse the bits for.

Returns

[byte](#)

A byte where the order of the bits is reversed.

Remarks

This method performs the reversal of the bits in the 8-bit unsigned byte value.

ReverseBits(byte[])

Reverses the order of the bits in a byte array.

```
public static byte[] ReverseBits(this byte[] bytes)
```

Parameters

bytes [byte](#)[]

The byte array whose bits are to be reversed.

Returns

[byte](#)[]

A new byte array with the bits of each byte reversed.

Remarks

This method processes each byte in the array individually and reverses the bits within each byte.

ReverseBits(short)

Reverses the order of the bits of an signed [short](#) value.

```
public static short ReverseBits(this short value)
```

Parameters

value [short](#)

The 16-bit unsigned integer value to reverse the bits for.

Returns

[short](#)

A 16-bit unsigned integer where the order of the bits is reversed.

Remarks

This method performs the reversal of the bits in the 16-bit unsigned integer value.

ReverseBits(int)

Reverses the order of the bits of an signed [int](#) value.

```
public static int ReverseBits(this int value)
```

Parameters

value [int](#)

The 32-bit unsigned integer value to reverse the bits for.

Returns

[int](#)

A 32-bit unsigned integer where the order of the bits is reversed.

Remarks

This method performs the reversal of the bits in the 32-bit unsigned integer value.

ReverseBits(long)

Reverses the order of the bits of an unsigned [long](#) value.

```
public static long ReverseBits(this long value)
```

Parameters

value [long](#)

The 64-bit unsigned integer value to reverse the bits for.

Returns

[long](#)

A 64-bit unsigned integer where the order of the bits is reversed.

Remarks

This method performs the reversal of the bits in the 64-bit unsigned integer value.

ReverseBits(ushort)

Reverses the order of the bits of an unsigned [ushort](#) value.

```
public static ushort ReverseBits(this ushort value)
```

Parameters

value [ushort](#)

The 16-bit unsigned integer value to reverse the bits for.

Returns

[ushort](#)

A 16-bit unsigned integer where the order of the bits is reversed.

Remarks

This method performs the reversal of the bits in the 16-bit unsigned integer value.

ReverseBits(uint)

Reverses the order of the bits of an unsigned [uint](#) value.

```
public static uint ReverseBits(this uint value)
```

Parameters

value [uint](#)

The 32-bit unsigned integer value to reverse the bits for.

Returns

[uint](#)

A 32-bit unsigned integer where the order of the bits is reversed.

Remarks

This method performs the reversal of the bits in the 32-bit unsigned integer value.

ReverseBits(ulong)

Reverses the order of the bits of an unsigned [ulong](#) value.

```
public static ulong ReverseBits(this ulong value)
```

Parameters

value [ulong](#)

The 64-bit unsigned integer value to reverse the bits for.

Returns

[ulong](#)

A 64-bit unsigned integer where the order of the bits is reversed.

Remarks

This method performs the reversal of the bits in the 64-bit unsigned integer value.

Enum QuarterDefinition

Namespace: [Bodu.Extensions](#)

Assembly: Bodu.CoreLib.dll

Specifies how quarters (Q1–Q4) are defined for a given calendar or financial year system.

```
public enum QuarterDefinition
```

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#)

Fields

`CalendarYear = 0`

Indicates that the year is divided into four quarters beginning in January.

Q1 = Jan–Mar, Q2 = Apr–Jun, Q3 = Jul–Sep, Q4 = Oct–Dec.

`Custom = 99`

Indicates that the quarter system is defined by a custom rule. Requires external logic to determine quarter boundaries.

`FinancialApril = 9`

Indicates that the fiscal year begins in April.

Q1 = Apr–Jun, Q2 = Jul–Sep, Q3 = Oct–Dec, Q4 = Jan–Mar.

Common in India, the UK, and parts of Japan and Canada.

`FinancialFebruary = 11`

Indicates that the fiscal year begins in February.

Q1 = Feb–Apr, Q2 = May–Jul, Q3 = Aug–Oct, Q4 = Nov–Jan.

Common in retail and some fiscal 4-4-5 accounting practices.

`FinancialJuly = 6`

Indicates that the fiscal year begins in July.

Q1 = Jul–Sep, Q2 = Oct–Dec, Q3 = Jan–Mar, Q4 = Apr–Jun.

Common in Australia and New Zealand.

FinancialOctober = 3

Indicates that the fiscal year begins in October.

Q1 = Oct–Dec, Q2 = Jan–Mar, Q3 = Apr–Jun, Q4 = Jul–Sep.

Used by the US Federal Government.

Remarks

This enumeration supports both standard calendar-based and regional fiscal year definitions, such as the Australian, US Federal, and ISO calendar conventions. Use [Custom](#) to define a custom quarterly system (e.g., 5–4–4 or 13-week accounting periods).

Enum StandardWeekend

Namespace: [Bodu.Extensions](#)

Assembly: Bodu.CoreLib.dll

Represents common weekend configurations used across global regions and cultures.

```
public enum StandardWeekend
```

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#).

Fields

Custom = 6

Custom definition provided via an [IWeekendProvider](#).

FridayOnly = 3

Friday only is considered the weekend (e.g., traditional Islamic week).

FridaySaturday = 1

Friday and Saturday are weekends (used in many Middle Eastern countries).

None = 5

No standard weekend is defined (all days may be working or off depending on context).

SaturdaySunday = 0

Saturday and Sunday are considered weekends (used in most Western countries).

SundayOnly = 4

Sunday only is considered the weekend (used historically and in some religious institutions).

ThursdayFriday = 2

Thursday and Friday are weekends (used in some Middle Eastern academic/institutional contexts).

Enum WeekOfMonthOrdinal

Namespace: [Bodu.Extensions](#)

Assembly: Bodu.CoreLib.dll

Specifies the ordinal position of a weekday within a given month. Commonly used in recurrence rules, such as "the second Tuesday of the month."

```
public enum WeekOfMonthOrdinal
```

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#)

Fields

Fifth = 4

Represents the fifth occurrence of a specific weekday in the month (e.g., the 5th Monday).

IMPORTANT

This is relatively rare and only occurs in months where five instances of the specified weekday exist.

First = 0

Represents the first occurrence of a specific weekday in the month (e.g., the 1st Monday).

Fourth = 3

Represents the fourth occurrence of a specific weekday in the month (e.g., the 4th Monday).

Last = 5

Represents the last occurrence of a specific weekday in the month (e.g., the last Monday).

Second = 1

Represents the second occurrence of a specific weekday in the month (e.g., the 2nd Monday).

Third = 2

Represents the third occurrence of a specific weekday in the month (e.g., the 3rd Monday).

Namespace Bodu.Globalization.Extensions

Classes

[DateTimeFormatInfoExtensions](#)

Provides a set of [static](#) ([Shared](#) in Visual Basic) methods that extend the [DateTimeFormatInfo](#) class.

Class DateTimeFormatInfoExtensions

Namespace: [Bodu.Globalization.Extensions](#)

Assembly: Bodu.CoreLib.dll

Provides a set of [static](#) (Shared in Visual Basic) methods that extend the [DateTimeFormatInfo](#) class.

```
public static class DateTimeFormatInfoExtensions
```

Inheritance

[object](#) ← DateTimeFormatInfoExtensions

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

LastDayOfWeek(DateTimeFormatInfo)

Returns the last day of the week based on the specified [DateTimeFormatInfo](#).

```
public static DayOfWeek LastDayOfWeek(this DateTimeFormatInfo info)
```

Parameters

info [DateTimeFormatInfo](#)

The [DateTimeFormatInfo](#) containing week configuration.

Returns

[DayOfWeek](#)

The last [DayOfWeek](#) of the week according to the provided culture.

Exceptions

[ArgumentNullException](#)

Thrown if `info` is `null`.

Namespace Bodu.Security.Cryptography

Classes

[Adler32](#)

Computes an Adler-32 checksum over input data.

[Bernstein](#)

Computes the hash for the input data using the [Bernstein](#) (djb2) hash algorithm.

[Crc](#)

Implementation of the cyclic redundancy check (CRC) error-detecting algorithm. This class cannot be inherited.

[CrcLookupTableCache](#)

A class to manage CRC lookup permutationTable caching based on CRC parameters.

[CrcStandard](#)

Represents the configuration settings for a CRC algorithm, including parameters like polynomial, initial value, reflection settings, and more.

[CrcUtility](#)

A utility class for handling CRC operations such as generating CRC lookup tables.

[CryptoUtilities](#)

Provides general-purpose utility methods used by cryptographic components and implementations. This includes bit manipulation, secure random byte generation, and helper functions to ensure compliance with cryptographic constraints such as non-zero padding, key generation, or exclusion of reserved byte values.

[CubeHash](#)

Computes the hash for the input data by using the [CubeHash](#) hash algorithm. This class cannot be inherited.

[Elf64](#)

Computes a 64-bit non-cryptographic hash using the ELF (Executable and Linkable Format) hashing algorithm.

[Fletcher](#)

Base class for computing hash using the Fletcher hash algorithm family (Fletcher-16, Fletcher-32, Fletcher-64). This class cannot be inherited.

[Fletcher16](#)

Computes the hash for the input data using the [Fletcher16](#) hash algorithm. This class cannot be inherited.

[Fletcher32](#)

Computes the hash for the input data using the [Fletcher32](#) hash algorithm. This class cannot be inherited.

[Fletcher64](#)

Computes the hash for the input data using the [Fletcher64](#) hash algorithm. This class cannot be inherited.

[JSHash](#)

Computes a non-cryptographic 32-bit hash using the [JSHash](#) algorithm by Justin Sobel.

[Pearson](#)

Computes a hash value using the Pearson hashing algorithm—a fast, lightweight, and non-cryptographic hash function suitable for basic checksums and hash-based lookups.

[SipHash](#)

Provides the base implementation of the [SipHash](#) cryptographic hash algorithm—a fast, secure, and keyed pseudorandom function optimized for short input messages. See the [official SipHash specification](#) for details.

[SipHash128](#)

[SipHash64](#)

Provides a sealed implementation of the [SipHash](#) cryptographic hash algorithm that produces a 64-bit hash output. This implementation uses a keyed Add-Rotate-XOR (ARX) construction optimized for short messages. See the [official SipHash specification](#) for details.

[TweakableSymmetricAlgorithm](#)

Represents a symmetric encryption algorithm that supports an additional tweak value in addition to the encryption key and initialization vector (IV).

Interfaces

[IResumableHashAlgorithm](#)

Represents a hash algorithm that supports resuming a previously finalized hash state and continuing the hash computation with additional input data.

Enums

[TransformMode](#)

Defines the direction of a cryptographic transformation.

Class Adler32

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

Computes an Adler-32 checksum over input data.

```
public sealed class Adler32 : HashAlgorithm, ICryptoTransform, IDisposable
```

Inheritance

[object](#) ← [HashAlgorithm](#) ← Adler32

Implements

[ICryptoTransform](#), [IDisposable](#)

Inherited Members

[HashAlgorithm.Clear\(\)](#), [HashAlgorithm.ComputeHash\(byte\[\]\)](#),
[HashAlgorithm.ComputeHash\(byte\[\], int, int\)](#), [HashAlgorithm.ComputeHash\(Stream\)](#),
[HashAlgorithm.ComputeHashAsync\(Stream, CancellationToken\)](#), [HashAlgorithm.Create\(\)](#),
[HashAlgorithm.Create\(string\)](#), [HashAlgorithm.Dispose\(\)](#),
[HashAlgorithm.TransformBlock\(byte\[\], int, int, byte\[\], int\)](#),
[HashAlgorithm.TransformFinalBlock\(byte\[\], int, int\)](#),
[HashAlgorithm.TryComputeHash\(ReadOnlySpan<byte>, Span<byte>, out int\)](#),
[HashAlgorithm.CanReuseTransform](#), [HashAlgorithm.CanTransformMultipleBlocks](#),
[HashAlgorithm.Hash](#), [HashAlgorithm.HashSize](#), [HashAlgorithm.InputBlockSize](#),
[HashAlgorithm.OutputBlockSize](#), [object.Equals\(object\)](#), [object.Equals\(object, object\)](#),
[object.GetHashCode\(\)](#), [object.GetType\(\)](#), [object.ReferenceEquals\(object, object\)](#),
[object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\], out bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], byte\[\], CancellationToken\)](#),

[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], string, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, string, Encoding, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], string\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, string\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#)

Remarks

Adler-32 is a fast, non-cryptographic checksum algorithm designed by Mark Adler for use in the zlib compression library. It produces a 32-bit checksum composed of two sums (A and B) modulo 65521.

IMPORTANT

This algorithm is **not** cryptographically secure and should **not** be used for digital signatures, password hashing, or integrity verification in security-sensitive contexts.

Constructors

Adler32()

Initializes a new instance of the [Adler32](#) class.

```
public Adler32()
```

Methods

Dispose(bool)

Releases the unmanaged resources used by the [HashAlgorithm](#) and optionally releases the managed resources.

```
protected override void Dispose(bool disposing)
```

Parameters

disposing [bool](#)

[true](#) to release both managed and unmanaged resources; [false](#) to release only unmanaged resources.

HashCore(byte[], int, int)

Processes a block of data by feeding it into the Fletcher algorithm.

```
protected override void HashCore(byte[] array, int ibStart, int cbSize)
```

Parameters

array [byte](#)[]

The byte array containing the data to be hashed.

ibStart [int](#)

The offset at which to start processing in the byte array.

cbSize [int](#)

The length of the data to process.

HashFinal()

When overridden in a derived class, finalizes the hash computation after the last data is processed by the cryptographic hash algorithm.

```
protected override byte[] HashFinal()
```

Returns

[byte](#)[]

The computed hash code.

Initialize()

Initializes or resets the internal state for a new checksum computation.

```
public override void Initialize()
```

Class Bernstein

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

Computes the hash for the input data using the [Bernstein](#) (djb2) hash algorithm.

```
public sealed class Bernstein : HashAlgorithm, ICryptoTransform, IDisposable
```

Inheritance

[object](#) ← [HashAlgorithm](#) ← Bernstein

Implements

[ICryptoTransform](#), [IDisposable](#)

Inherited Members

[HashAlgorithm.Clear\(\)](#), [HashAlgorithm.ComputeHash\(byte\[\]\)](#),
[HashAlgorithm.ComputeHash\(byte\[\], int, int\)](#), [HashAlgorithm.ComputeHash\(Stream\)](#),
[HashAlgorithm.ComputeHashAsync\(Stream, CancellationToken\)](#), [HashAlgorithm.Create\(\)](#),
[HashAlgorithm.Create\(string\)](#), [HashAlgorithm.Dispose\(\)](#),
[HashAlgorithm.TransformBlock\(byte\[\], int, int, byte\[\], int\)](#),
[HashAlgorithm.TransformFinalBlock\(byte\[\], int, int\)](#),
[HashAlgorithm.TryComputeHash\(ReadOnlySpan<byte>, Span<byte>, out int\)](#), [HashAlgorithm.Hash](#),
[HashAlgorithm.HashSize](#), [HashAlgorithm.InputBlockSize](#), [HashAlgorithm.OutputBlockSize](#),
[object.Equals\(object, object\)](#), [object.GetType\(\)](#), [object.ReferenceEquals\(object, object\)](#),
[object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\], out bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], byte\[\], CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], string, CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#),

[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, string, Encoding, byte\[\], CancellationToken\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], string\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, string\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#),
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#),
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#).

Remarks

The [Bernstein](#) class implements the non-cryptographic hash function known as djb2, created by Daniel J. Bernstein. It is widely used in hash tables, data indexing, and similar scenarios where speed and simplicity are preferred over cryptographic guarantees.

This implementation includes an optional variant of the algorithm that uses an XOR instead of addition when combining characters into the hash. You can control this behavior with the [UseModifiedAlgorithm](#) property:

- Set [UseModifiedAlgorithm](#) to [false](#) to use the standard djb2 logic: `hash = (hash * 33) + c.`
- Set [UseModifiedAlgorithm](#) to [true](#) to use the XOR-modified variant: `hash = (hash * 33) ^ c.` This version may offer improved distribution properties in certain hash permutationTable implementations.

Both versions produce a 32-bit integer hash from the input stream of bytes.

IMPORTANT

This algorithm is **not** cryptographically secure and should **not** be used for password hashing, digital signatures, or any use case that requires secure integrity or confidentiality.

Constructors

Bernstein()

Initializes a new instance of the [Bernstein](#) class with default parameters.

```
public Bernstein()
```

Fields

DefaultInitialValue

The default initial value used to seed the hash algorithm. This is constant.

```
public const uint DefaultInitialValue = 5381
```

Field Value

[uint](#)

Properties

CanReuseTransform

Gets a value indicating whether the current transform can be reused.

```
public override bool CanReuseTransform { get; }
```

Property Value

[bool](#)

Always [true](#).

CanTransformMultipleBlocks

When overridden in a derived class, gets a value indicating whether multiple blocks can be transformed.

```
public override bool CanTransformMultipleBlocks { get; }
```

Property Value

[bool](#)

[true](#) if multiple blocks can be transformed; otherwise, [false](#).

InitialValue

Gets or sets the initial seed value used to start the hash computation.

```
public uint InitialValue { get; set; }
```

Property Value

[uint](#)

The initial hash code value. Defaults to [DefaultInitialValue](#).

Exceptions

[ObjectDisposedException](#)

Instance has been disposed and its members are accessed.

[CryptographicUnexpectedOperationException](#)

The hash computation has already started.

UseModifiedAlgorithm

Gets or sets a value indicating whether to use the XOR-modified variant of the [Bernstein](#) hash algorithm.

```
public bool UseModifiedAlgorithm { get; set; }
```

Property Value

bool

[true](#) to use the modified algorithm (`hash = (hash * 33) ^ c;`) or [false](#) to use the original djb2 form (`hash = (hash * 33) + c`). The default is [false](#).

Exceptions

[ObjectDisposedException](#)

Instance has been disposed and its members are accessed.

[CryptographicUnexpectedOperationException](#)

The hash computation has already started.

Methods

Dispose(bool)

Releases the unmanaged resources used by the [HashAlgorithm](#) and optionally releases the managed resources.

```
protected override void Dispose(bool disposing)
```

Parameters

[disposing](#) bool

[true](#) to release both managed and unmanaged resources; [false](#) to release only unmanaged resources.

Equals(object)

Determines whether the specified object is equal to the current object.

```
public override bool Equals(object obj)
```

Parameters

obj [object](#)

The object to compare with the current object.

Returns

[bool](#)

[true](#) if the specified object is equal to the current object; otherwise, [false](#).

GetHashCode()

Serves as the default hash function.

```
public override int GetHashCode()
```

Returns

[int](#)

A hash code for the current object.

HashCore(byte[], int, int)

When overridden in a derived class, routes data written to the object into the hash algorithm for computing the hash.

```
protected override void HashCore(byte[] array, int ibStart, int cbSize)
```

Parameters

array [byte](#)[]

The input to compute the hash code for.

ibStart [int](#)

The offset into the byte array from which to begin using data.

`cbSize` [int](#)

The number of bytes in the byte array to use as data.

HashFinal()

Finalizes the hash computation and returns the resulting hash value.

```
protected override byte[] HashFinal()
```

Returns

[byte](#)[]

A byte array containing the computed hash.

Initialize()

Resets the hash algorithm to its initial state.

```
public override void Initialize()
```

Class Crc

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

Implementation of the cyclic redundancy check (CRC) error-detecting algorithm. This class cannot be inherited.

```
public sealed class Crc : HashAlgorithm, ICryptoTransform,  
IDisposable, IResumableHashAlgorithm
```

Inheritance

[object](#) ← [HashAlgorithm](#) ← Crc

Implements

[ICryptoTransform](#), [IDisposable](#), [IResumableHashAlgorithm](#)

Inherited Members

[HashAlgorithm.Clear\(\)](#), [HashAlgorithm.ComputeHash\(byte\[\]\)](#),
[HashAlgorithm.ComputeHash\(byte\[\], int, int\)](#), [HashAlgorithm.ComputeHash\(Stream\)](#),
[HashAlgorithm.ComputeHashAsync\(Stream, CancellationToken\)](#), [HashAlgorithm.Create\(\)](#),
[HashAlgorithm.Create\(string\)](#), [HashAlgorithm.Dispose\(\)](#),
[HashAlgorithm.TransformBlock\(byte\[\], int, int, byte\[\], int\)](#),
[HashAlgorithm.TransformFinalBlock\(byte\[\], int, int\)](#),
[HashAlgorithm.TryComputeHash\(ReadOnlySpan<byte>, Span<byte>, out int\)](#),
[HashAlgorithm.CanReuseTransform](#), [HashAlgorithm.CanTransformMultipleBlocks](#),
[HashAlgorithm.Hash](#), [HashAlgorithm.HashSize](#), [HashAlgorithm.InputBlockSize](#),
[HashAlgorithm.OutputBlockSize](#), [object.Equals\(object, object\)](#), [object.GetType\(\)](#),
[object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\], out bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#),

[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], string, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, string, Encoding, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], string\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, string\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#)

Remarks

The [Crc](#) class computes CRC hashes based on various CRC standards, including CRC32. This class uses CRC parameters defined in the [CrcStandard](#) class, such as the polynomial, initial value, reflection settings, and XOR out value. It provides methods for CRC calculation and updates using byte arrays.

IMPORTANT

This algorithm is **not** cryptographically secure and should **not** be used for digital signatures, password hashing, or integrity verification in security-sensitive contexts.

Constructors

Crc()

Initializes a new instance of the [Crc](#) class using the default CRC standard (CRC32_ISOHDLC).

```
public Crc()
```

Remarks

The default CRC standard is CRC32 with the following parameters:

Crc(CrcStandard)

Initializes a new instance of the [Crc](#) class using the specified CRC parameters.

```
public Crc(CrcStandard crcStandard)
```

Parameters

crcStandard [CrcStandard](#)

The [CrcStandard](#) to use in creating the CRC value.

Remarks

Initializes the CRC with the given parameters for a customizable CRC computation.

Exceptions

[ArgumentNullException](#)

Thrown when the **crcStandard** is [null](#).

[ArgumentOutOfRangeException](#)

Thrown when the **crcStandard** size is outside the supported range (1 to 64 bits).

Properties

CrcStandard

Gets the [CrcStandard](#) used by this instance to perform the CRC operation.

```
public CrcStandard CrcStandard { get; }
```

Property Value

[CrcStandard](#)

The CRC parameters containing details such as polynomial, size, and reflection settings.

Remarks

This property gives access to the configuration used for CRC calculation. The user can inspect the CRC parameters, including the polynomial, initial value, and reflection settings.

GlobalCache

Gets or sets the global cache used to manage CRC lookup tables.

```
public static CrcLookupTableCache GlobalCache { get; set; }
```

Property Value

[CrcLookupTableCache](#)

Remarks

This static property allows users to set a global cache that can be shared across all instances of [Crc](#). If not set, a default cache will be used. Setting this property allows the user to manage the cache externally and reuse lookup tables across multiple CRC instances.

Exceptions

[InvalidOperationException](#)

Thrown when setting the [GlobalCache](#) as [null](#).

InitialValue

Gets the initial value used in the CRC calculation.

```
public ulong InitialValue { get; }
```

Property Value

[ulong](#)

The initial value for the CRC calculation.

Name

Gets the name of the CRC standard.

```
public string Name { get; }
```

Property Value

[string](#)

The name of the CRC algorithm.

Polynomial

Gets the polynomial used in the CRC calculation.

```
public ulong Polynomial { get; }
```

Property Value

[ulong](#)

The polynomial value used in the CRC calculation.

ReflectIn

Gets a value indicating whether the data data is reflected during the CRC calculation.

```
public bool ReflectIn { get; }
```

Property Value

[bool](#)

[true](#) if data data is reflected; otherwise, [false](#).

ReflectOut

Gets a value indicating whether the CRC result is reflected before XORing with [XOrOut](#).

```
public bool ReflectOut { get; }
```

Property Value

[bool](#)

[true](#) if the result is reflected; otherwise, [false](#).

Size

Gets the size, in bits, of the CRC checksum.

```
public int Size { get; }
```

Property Value

[int](#)

The size of the CRC in bits.

XOrOut

Gets the value to XOR the final CRC result with.

```
public ulong XorOut { get; }
```

Property Value

[ulong](#)

The XOR value for the final CRC result.

Methods

ComputeHash(ReadOnlySpan<byte>)

Computes the CRC hash of the specified input data contained in the provided [ReadOnlySpan<T>](#).

```
public byte[] ComputeHash(ReadOnlySpan<byte> data)
```

Parameters

data [ReadOnlySpan](#)<[byte](#)>

The input data to compute the hash for.

Returns

[byte](#)[]

A byte array containing the computed CRC hash value.

Remarks

This method initializes the internal CRC state, processes the input data using the configured CRC variant, and finalizes the result into a hash of the appropriate size. It supports both reflected and unreflected input, and will apply bytewise or bitwise logic based on the parameters defined in [CrcStandard](#).

ComputeHashFrom(byte[], byte[])

Resumes a hash computation from a previously finalized hash value and processes additional input, returning the new finalized hash result as a byte array.

```
public byte[] ComputeHashFrom(byte[] previousHash, byte[] newData)
```

Parameters

previousHash [byte](#)[]

The previously finalized hash value to resume from.

newData [byte](#)[]

The additional input data to include in the resumed hash calculation.

Returns

[byte\[\]](#)

A byte array containing the new finalized hash result.

Remarks

This overload reverses finalization on `previousHash` by undoing XOR and reflection (if applicable), continues the CRC computation with the full `newData` array, and returns the finalized CRC hash value as a new byte array.

Exceptions

[ArgumentException](#)

Thrown if the `previousHash` length does not match [HashSize](#).

ComputeHashFrom(byte[], byte[], int, int)

Resumes a hash computation from a previously finalized hash value and processes a specified range of new data, returning the new finalized hash result as a byte array.

```
public byte[] ComputeHashFrom(byte[] previousHash, byte[] newData, int offset, int length)
```

Parameters

`previousHash` [byte\[\]](#)

The previously finalized hash value to resume from.

`newData` [byte\[\]](#)

The buffer containing additional input data.

`offset` [int](#)

The zero-based offset into `newData` at which to begin reading data.

`length` [int](#)

The number of bytes to read from `newData`.

Returns

[byte\[\]](#)

A byte array containing the new finalized hash result.

Remarks

This overload reverses finalization on `previousHash` by undoing XOR and reflection (if applicable), continues the CRC computation with a sliced segment of `newData` (starting at `offset` and spanning `length`), and returns the finalized CRC hash value as a new byte array.

Exceptions

[ArgumentException](#)

Thrown if the `previousHash` length does not match [HashSize](#), or if the offset and length exceed the bounds of `newData`.

ComputeHashFrom(ReadOnlySpan<byte>, ReadOnlySpan<byte>)

Resumes a hash computation from a previously finalized hash value and processes additional input, returning the new finalized hash result as a byte array.

```
public byte[] ComputeHashFrom(ReadOnlySpan<byte> previousHash, ReadOnlySpan<byte> newData)
```

Parameters

`previousHash` [ReadOnlySpan<byte>](#)

The previously finalized hash value to resume from.

`newData` [ReadOnlySpan<byte>](#)

The additional input data to include in the resumed hash calculation.

Returns

[byte\[\]](#)

A byte array containing the new finalized hash result.

Remarks

This method reverses finalization on the provided [previousHash](#), resumes the CRC computation with the contents of [newData](#), and returns the final CRC hash as a new byte array.

Exceptions

[ArgumentException](#)

Thrown if the [previousHash](#) length does not match [HashSize](#).

Dispose(bool)

Releases the unmanaged resources used by the [HashAlgorithm](#) and optionally releases the managed resources.

```
protected override void Dispose(bool disposing)
```

Parameters

[disposing](#) [bool](#)

[true](#) to release both managed and unmanaged resources; [false](#) to release only unmanaged resources.

Equals(object)

Determines whether the specified object is equal to the current object.

```
public override bool Equals(object obj)
```

Parameters

[obj](#) [object](#)

The object to compare with the current object.

Returns

[bool](#)

[true](#) if the specified object is equal to the current object; otherwise, [false](#).

GetHashCode()

Serves as the default hash function.

```
public override int GetHashCode()
```

Returns

[int](#)

A hash code for the current object.

HashCore(byte[], int, int)

When overridden in a derived class, routes data written to the object into the hash algorithm for computing the hash.

```
protected override void HashCore(byte[] array, int ibStart, int cbSize)
```

Parameters

array [byte](#)[]

The input to compute the hash code for.

ibStart [int](#)

The offset into the byte array from which to begin using data.

cbSize [int](#)

The number of bytes in the byte array to use as data.

HashCore(ReadOnlySpan<byte>)

Routes data written to the object into the hash algorithm for computing the hash.

```
protected override void HashCore(ReadOnlySpan<byte> source)
```

Parameters

source [ReadOnlySpan<byte>](#)

The input to compute the hash code for.

HashFinal()

When overridden in a derived class, finalizes the hash computation after the last data is processed by the cryptographic hash algorithm.

```
protected override byte[] HashFinal()
```

Returns

[byte\[\]](#)

The computed hash code.

Initialize()

Initializes the current instance of the [Crc](#) class using the initial value from the CRC parameters.

```
public override void Initialize()
```

Remarks

This method initializes the internal state (CRC value) based on the specified CRC parameters. If [ReflectIn](#) is true, the initial value is reversed.

TryComputeHashFrom(ReadOnlySpan<byte>, ReadOnlySpan<byte>, Span<byte>, out int)

Resumes a hash computation from a previously finalized hash value, processes additional input, and writes the new finalized hash to the specified destination span.

```
public bool TryComputeHashFrom(ReadOnlySpan<byte> previousHash, ReadOnlySpan<byte> newData,  
Span<byte> destination, out int bytesWritten)
```

Parameters

previousHash [ReadOnlySpan<byte>](#)

The previously finalized hash value to resume from.

newData [ReadOnlySpan<byte>](#)

The additional input data to include in the resumed hash calculation.

destination [Span<byte>](#)

The destination buffer to write the finalized hash value to.

bytesWritten [int](#)

Outputs the number of bytes written to the destination buffer.

Returns

[bool](#)

[true](#) if the resumed and finalized hash was written successfully; otherwise, [false](#) if the destination span was too small.

Remarks

This method reverses finalization on **previousHash** by undoing the XOR and reflection (if applicable), continues the CRC computation with **newData**, and finalizes the result into **destination**.

Exceptions

[ArgumentException](#)

Thrown if the `previousHash` length does not match [HashSize](#).

TryFinalizeHash(Span<byte>, out int)

Finalizes the CRC computation and writes the resulting hash value into the specified destination buffer.

```
public bool TryFinalizeHash(Span<byte> destination, out int bytesWritten)
```

Parameters

`destination` [Span](#)<[byte](#)>

The span to write the finalized CRC hash value into.

`bytesWritten` [int](#)

When this method returns, contains the number of bytes written to `destination`.

Returns

[bool](#)

[true](#) if the hash was successfully written to the destination; otherwise, [false](#).

Remarks

This method applies any final transformations required by the CRC specification, including reflection and XOR output, and serializes the internal CRC value into the provided destination buffer.

TryHashFinal(Span<byte>, out int)

Attempts to finalize the hash computation after the last data is processed by the hash algorithm.

```
protected override bool TryHashFinal(Span<byte> destination, out int bytesWritten)
```

Parameters

`destination` [Span](#)<[byte](#)>

The buffer to receive the hash value.

`bytesWritten` [int](#)

When this method returns, the total number of bytes written into `destination`. This parameter is treated as uninitialized.

Returns

[bool](#)

[true](#) if `destination` is long enough to receive the hash value; otherwise, [false](#).

Class CrcLookupTableCache

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

A class to manage CRC lookup permutationTable caching based on CRC parameters.

```
public class CrcLookupTableCache
```

Inheritance

[object](#) ← CrcLookupTableCache

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#).

Constructors

CrcLookupTableCache()

```
public CrcLookupTableCache()
```

Methods

GetLookupTable(int, ulong, bool)

```
public ImmutableArray<ulong> GetLookupTable(int size, ulong polynomial, bool reflectIn)
```

Parameters

size [int](#)

polynomial [ulong](#)

`reflectIn` [bool](#)

Returns

[ImmutableArray](#)<[ulong](#)>

Class CrcStandard

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

Represents the configuration settings for a CRC algorithm, including parameters like polynomial, initial value, reflection settings, and more.

```
[Serializable]
public sealed class CrcStandard : ISerializable, IEquatable<CrcStandard>
```

Inheritance

[object](#) ← CrcStandard

Implements

[ISerializable](#), [IEquatable](#)<[CrcStandard](#)>

Inherited Members

[object.Equals\(object, object\)](#), [object.GetType\(\)](#), [object.ReferenceEquals\(object, object\)](#),
[object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#)

Constructors

CrcStandard(string, int, ulong, ulong, bool, bool, ulong)

Initializes a new instance of the [CrcStandard](#) class with the specified parameters.

```
public CrcStandard(string name, int size, ulong polynomial, ulong initialValue, bool
reflectIn, bool reflectOut, ulong xOrOut)
```

Parameters

name [string](#)

The name of the CRC standard.

size [int](#)

The size, in bits, of the CRC checksum.

polynomial [ulong](#)

The CRC polynomial value.

initialValue [ulong](#)

The initial value used for the CRC calculation.

reflectIn [bool](#)

Indicates whether to reflect the input during the CRC calculation.

reflectOut [bool](#)

Indicates whether to reflect the output during the CRC calculation.

xOrOut [ulong](#)

The value to XOR the final output with.

Exceptions

[ArgumentException](#)

Thrown if **name** is null or empty.

[ArgumentOutOfRangeException](#)

Thrown if **size** is outside the valid range.

Fields

MaxSize

The maximum size allowed for a CRC standard (in bits).

```
public const int MaxSize = 64
```

Field Value

[int↗](#)

MinSize

The minimum size allowed for a CRC standard (in bits).

```
public const int MinSize = 1
```

Field Value

[int↗](#)

Properties

ARC

Gets the [CrcStandard](#) that defines the **CRC-16/ARC** cyclic redundancy check algorithm standard.

```
public static CrcStandard ARC { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **ARC** definition.

Remarks

The **ARC** is an alias of the [CRC16_ARC](#) standard.

See Also

[CRC16_ARC](#), [CRC16](#), [CRC16_LHA](#), [CRCIBM](#)

BCRC32

Gets the [CrcStandard](#) that defines the **CRC-32/BZIP2** cyclic redundancy check algorithm standard.

```
public static CrcStandard BCRC32 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **B-CRC-32** definition.

Remarks

The **BCRC32** is an alias of the [CRC32_BZIP2](#) standard.

See Also

[CRC32_BZIP2](#), [CRC32_AAL5](#), [CRC32_DECTB](#)

CKSUM

Gets the [CrcStandard](#) that defines the **CRC-32/CKSUM** cyclic redundancy check algorithm standard.

```
public static CrcStandard CKSUM { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CKSUM** definition.

Remarks

The **CKSUM** is an alias of the [CRC32_CKSUM](#) standard.

See Also

[CRC32_CKSUM](#), [CRC32_POSIX](#)

CRC10

Gets the [CrcStandard](#) that defines the **CRC-10/ATM** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC10 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-10** definition.

Remarks

The **CRC10** is an alias of the [CRC10_ATM](#) standard.

See Also

[CRC10_ATM](#), [CRC10_I610](#)

CRC10_ATM

Gets the [CrcStandard](#) that defines the **CRC-10/ATM** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC10_ATM { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-10/ATM** definition.

Remarks

The **CRC-10/ATM** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 19 April 2009, Updated: 7 May 2022), with the following definition.

- Width: **10**
- Polynomial: **0x0233**
- Initial Value: **0x0000**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x0000**

The **CRC-10/ATM** standard is also known by the aliases **CRC-10**, and **CRC-10/I-610**.

See Also

[CRC10](#), [CRC10_I610](#)

CRC10_CDMA2000

Gets the [CrcStandard](#) that defines the **CRC-10/CDMA2000** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC10_CDMA2000 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-10/CDMA2000** definition.

Remarks

NOTE

The parameters for the **CRC-10/CDMA2000** standard are not widely tested or confirmed.

The **CRC-10/CDMA2000** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 14 October 2013, Updated: 28 December 2019), with the following definition.

- Width: **10**
- Polynomial: **0x03D9**
- Initial Value: **0x03FF**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x0000**

CRC10_GSM

Gets the [CrcStandard](#) that defines the **CRC-10/GSM** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC10_GSM { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-10/GSM** definition.

Remarks

NOTE

The parameters for the **CRC-10/GSM** standard are not widely tested or confirmed.

The **CRC-10/GSM** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 17 February 2017), with the following definition.

- Width: **10**
- Polynomial: **0x0175**
- Initial Value: **0x0000**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x03FF**

CRC10_I610

Gets the [CrcStandard](#) that defines the **CRC-10/ATM** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC10_I610 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-10/I-610** definition.

Remarks

The **CRC10_I610** is an alias of the [CRC10_ATM](#) standard.

See Also

[CRC10_ATM](#), [CRC10](#)

CRC11

Gets the [CrcStandard](#) that defines the **CRC-11/FLEXRAY** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC11 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-11** definition.

Remarks

The **CRC11** is an alias of the [CRC11_FLEXRAY](#) standard.

See Also

[CRC11_FLEXRAY](#)

CRC11_FLEXRAY

Gets the [CrcStandard](#) that defines the **CRC-11/FLEXRAY** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC11_FLEXRAY { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-11/FLEXRAY** definition.

Remarks

The **CRC-11/FLEXRAY** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 3 November 2007, Updated: 7 May 2022), with the following definition.

- Width: **11**
- Polynomial: **0x0385**
- Initial Value: **0x001A**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x0000**

The **CRC-11/FLEXRAY** standard is also known by the alias **CRC-11**.

See Also

[CRC11](#)

CRC11_UMTS

Gets the [CrcStandard](#) that defines the **CRC-11/UMTS** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC11_UMTS { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-11/UMTS** definition.

Remarks

 **NOTE**

The parameters for the **CRC-11/UMTS** standard are not widely tested or confirmed.

The **CRC-11/UMTS** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 8 February 2016, Updated: 7 May 2022), with the following definition.

- Width: **11**
- Polynomial: **0x0307**
- Initial Value: **0x0000**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x0000**

CRC12_3GPP

Gets the [CrcStandard](#) that defines the **CRC-12/UMTS** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC12_3GPP { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-12/3GPP** definition.

Remarks

The **CRC12_3GPP** is an alias of the [CRC12_UMTS](#) standard.

See Also

[CRC12_UMTS](#)

CRC12_CDMA2000

Gets the [CrcStandard](#) that defines the **CRC-12/CDMA2000** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC12_CDMA2000 { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-12/CDMA2000** definition.

Remarks

NOTE

The parameters for the **CRC-12/CDMA2000** standard are not widely tested or confirmed.

The **CRC-12/CDMA2000** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 14 October 2013, Updated: 28 December 2019), with the following definition.

- Width: **12**
- Polynomial: **0x0F13**
- Initial Value: **0xFFFF**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x0000**

CRC12_DECT

Gets the [CrcStandard](#) that defines the **CRC-12/DECT** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC12_DECT { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-12/DECT** definition.

Remarks

NOTE

The parameters for the **CRC-12/DECT** standard are not widely tested or confirmed.

The **CRC-12/DECT** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 29 July 2010, Updated: 15 February 2017), with the following definition.

- Width: **12**
- Polynomial: **0x080F**
- Initial Value: **0x0000**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x0000**

The **CRC-12/DECT** standard is also known by the alias **X-CRC-12**.

See Also

[XCRC12](#)

CRC12_GSM

Gets the [CrcStandard](#) that defines the **CRC-12/GSM** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC12_GSM { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-12/GSM** definition.

Remarks

NOTE

The parameters for the **CRC-12/GSM** standard are not widely tested or confirmed.

The **CRC-12/GSM** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 17 February 2017), with the following definition.

- Width: **12**
- Polynomial: **0x0D31**
- Initial Value: **0x0000**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x0FFF**

CRC12_UMTS

Gets the [CrcStandard](#) that defines the **CRC-12/UMTS** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC12_UMTS { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-12/UMTS** definition.

Remarks

NOTE

The parameters for the **CRC-12/UMTS** standard are not widely tested or confirmed.

The **CRC-12/UMTS** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 10 December 2009, Updated: 25 May 2022), with the following definition.

- Width: **12**
- Polynomial: **0x080F**
- Initial Value: **0x0000**
- Reflect In: **False**
- Reflect Out: **True**
- XOR Out: **0x0000**

The **CRC-12/UMTS** standard is also known by the alias **CRC-12/3GPP**.

See Also

[CRC12_3GPP](#)

CRC13_BBC

Gets the [**CrcStandard**](#) that defines the **CRC-13/BBC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC13_BBC { get; }
```

Property Value

[**CrcStandard**](#)

A [**CrcStandard**](#) object with the properties set to the **CRC-13/BBC** definition.

Remarks

The **CRC-13/BBC** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 14 October 2013, Updated: 6 February 2017), with the following definition.

- Width: **13**
- Polynomial: **0x1CF5**
- Initial Value: **0x0000**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x0000**

CRC14_DARC

Gets the [CrcStandard](#) that defines the **CRC-14/DARC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC14_DARC { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-14/DARC** definition.

Remarks

The **CRC-14/DARC** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 14 December 2009, Updated: 20 November 2018), with the following definition.

- Width: **14**
- Polynomial: **0x0805**
- Initial Value: **0x0000**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0x0000**

CRC14_GSM

Gets the [CrcStandard](#) that defines the **CRC-14/GSM** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC14_GSM { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-14/GSM** definition.

Remarks

NOTE

The parameters for the **CRC-14/GSM** standard are not widely tested or confirmed.

The [CRC-14/GSM](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 17 February 2017), with the following definition.

- Width: [14](#)
- Polynomial: [0x202D](#)
- Initial Value: [0x0000](#)
- Reflect In: [False](#)
- Reflect Out: [False](#)
- XOR Out: [0x3FFF](#)

CRC15

Gets the [CrcStandard](#) that defines the [CRC-15/CAN](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC15 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-15](#) definition.

Remarks

The [CRC15](#) is an alias of the [CRC15_CAN](#) standard.

See Also

[CRC15_CAN](#)

CRC15_CAN

Gets the [CrcStandard](#) that defines the [CRC-15/CAN](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC15_CAN { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-15/CAN](#) definition.

Remarks

NOTE

The parameters for the [CRC-15/CAN](#) standard are not widely tested or confirmed.

The [CRC-15/CAN](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 2 November 2007, Updated: 7 May 2022), with the following definition.

- Width: [15](#)
- Polynomial: [0x4599](#)
- Initial Value: [0x0000](#)
- Reflect In: [False](#)
- Reflect Out: [False](#)
- XOR Out: [0x0000](#)

The [CRC-15/CAN](#) standard is also known by the alias [CRC-15](#).

See Also

[CRC15](#)

CRC15_MPT1327

Gets the [CrcStandard](#) that defines the [CRC-15/MPT1327](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC15_MPT1327 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-15/MPT1327](#) definition.

Remarks

The [CRC-15/MPT1327](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 16 July 2012, Updated: 29 December 2021), with the following definition.

- Width: `15`
- Polynomial: `0x6815`
- Initial Value: `0x0000`
- Reflect In: `False`
- Reflect Out: `False`
- XOR Out: `0x0001`

CRC16

Gets the [CrcStandard](#) that defines the **CRC-16/ARC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16** definition.

Remarks

The **CRC16** is an alias of the [CRC16 ARC](#) standard.

See Also

[CRC16 ARC](#), [ARC](#), [CRC16 LHA](#), [CRCIBM](#)

CRC16_ACORN

Gets the [CrcStandard](#) that defines the **CRC-16/XMODEM** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_ACORN { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/ACORN** definition.

Remarks

The [CRC16_ACORN](#) is an alias of the [CRC16_XMODEM](#) standard.

See Also

[CRC16_XMODEM](#), [CRC16_LTE](#), [CRC16_V41MSB](#), [XMODEM](#), [ZMODEM](#)

CRC16_ARC

Gets the [CrcStandard](#) that defines the [CRC-16/ARC](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_ARC { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-16/ARC](#) definition.

Remarks

The [CRC-16/ARC](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 30 March 2005, Updated: 7 May 2022), with the following definition.

- Width: [16](#)
- Polynomial: [0x8005](#)
- Initial Value: [0x0000](#)
- Reflect In: [True](#)
- Reflect Out: [True](#)
- XOR Out: [0x0000](#)

The [CRC-16/ARC](#) standard is also known by the aliases [ARC](#), [CRC-16](#), [CRC-16/LHA](#), and [CRC-IBM](#).

See Also

[ARC](#), [CRC16](#), [CRC16_LHA](#), [CRCIBM](#)

CRC16_AUGCCITT

Gets the [CrcStandard](#) that defines the [CRC-16/SPI-FUJITSU](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_AUGCCITT { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/AUG-CCITT** definition.

Remarks

The **CRC16_AUGCCITT** is an alias of the [CRC16_SPIFUJITSU](#) standard.

See Also

[CRC16_SPIFUJITSU](#)

CRC16_AUTOSAR

Gets the [CrcStandard](#) that defines the **CRC-16/IBM-3740** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_AUTOSAR { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/AUTOSAR** definition.

Remarks

The **CRC16_AUTOSAR** is an alias of the [CRC16_IBM3740](#) standard.

See Also

[CRC16_IBM3740](#), [CRC16_CCITTFALSE](#)

CRC16_BUYPASS

Gets the [CrcStandard](#) that defines the **CRC-16/UMTS** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_BUYPASS { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/BUYPASS** definition.

Remarks

The **CRC16_BUYPASS** is an alias of the [CRC16_UMTS](#) standard.

See Also

[CRC16_UMTS](#), [CRC16_VERIFONE](#)

CRC16_CCITT

Gets the [CrcStandard](#) that defines the **CRC-16/KERMIT** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_CCITT { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/CCITT** definition.

Remarks

The **CRC16_CCITT** is an alias of the [CRC16_KERMIT](#) standard.

See Also

[CRC16_KERMIT](#), [CRC16_CCITTRUE](#), [CRC16_V41LSB](#), [CRCCCITT](#), [KERMIT](#)

CRC16_CCITTFALSE

Gets the [CrcStandard](#) that defines the **CRC-16/IBM-3740** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_CCITTFALSE { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/CCITT-FALSE** definition.

Remarks

The **CRC16_CCITTFALSE** is an alias of the [CRC16_IBM3740](#) standard.

See Also

[CRC16_IBM3740](#), [CRC16_AUTOSAR](#)

CRC16_CCITTRUE

Gets the [CrcStandard](#) that defines the **CRC-16/KERMIT** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_CCITTRUE { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/CCITT-TRUE** definition.

Remarks

The **CRC16_CCITTRUE** is an alias of the [CRC16_KERMIT](#) standard.

See Also

[CRC16_KERMIT](#), [CRC16_CCITT](#), [CRC16_V41LSB](#), [CRCCCITT](#), [KERMIT](#)

CRC16_CDMA2000

Gets the [CrcStandard](#) that defines the **CRC-16/CDMA2000** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_CDMA2000 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/CDMA2000** definition.

Remarks

NOTE

The parameters for the **CRC-16/CDMA2000** standard are not widely tested or confirmed.

The **CRC-16/CDMA2000** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 14 October 2013, Updated: 28 December 2019), with the following definition.

- Width: **16**
- Polynomial: **0xC867**
- Initial Value: **0xFFFF**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x0000**

CRC16_CMS

Gets the [CrcStandard](#) that defines the **CRC-16/CMS** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_CMS { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/CMS** definition.

Remarks

NOTE

The parameters for the **CRC-16/CMS** standard are not widely tested or confirmed.

The **CRC-16/CMS** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 13 July 2016, Updated: 28 December 2019), with the following definition.

- Width: **16**
- Polynomial: **0x8005**

- Initial Value: `0xFFFF`
- Reflect In: `False`
- Reflect Out: `False`
- XOR Out: `0x0000`

CRC16_DARC

Gets the [CrcStandard](#) that defines the **CRC-16/GENIBUS** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_DARC { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/DARC** definition.

Remarks

The **CRC16_DARC** is an alias of the [CRC16_GENIBUS](#) standard.

See Also

[CRC16_GENIBUS](#), [CRC16_EPC](#), [CRC16_EPCC1G2](#), [CRC16_ICODE](#)

CRC16_DDS110

Gets the [CrcStandard](#) that defines the **CRC-16/DDS-110** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_DDS110 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/DDS-110** definition.

Remarks

The **CRC-16/DDS-110** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 30 November 2009, Updated: 28 December 2019), with the following definition.

- Width: `16`
- Polynomial: `0x8005`
- Initial Value: `0x800D`
- Reflect In: `False`
- Reflect Out: `False`
- XOR Out: `0x0000`

CRC16_DECTR

Gets the [CrcStandard](#) that defines the **CRC-16/DECT-R** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_DECTR { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/DECT-R** definition.

Remarks

The **CRC-16/DECT-R** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 7 December 2009, Updated: 28 December 2019), with the following definition.

- Width: `16`
- Polynomial: `0x0589`
- Initial Value: `0x0000`
- Reflect In: `False`
- Reflect Out: `False`
- XOR Out: `0x0001`

The **CRC-16/DECT-R** standard is also known by the alias **R-CRC-16**.

See Also

[RCRC16](#)

CRC16_DECTX

Gets the [CrcStandard](#) that defines the **CRC-16/DECT-X** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_DECTX { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/DECT-X** definition.

Remarks

The **CRC-16/DECT-X** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 30 July 2010, Updated: 2 January 2021), with the following definition.

- Width: **16**
- Polynomial: **0x0589**
- Initial Value: **0x0000**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x0000**

The **CRC-16/DECT-X** standard is also known by the alias **X-CRC-16**.

See Also

[XCRC16](#)

CRC16_DNP

Gets the [CrcStandard](#) that defines the **CRC-16/DNP** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_DNP { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/DNP** definition.

Remarks

The **CRC-16/DNP** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 2 November 2007, Updated: 28 December 2019), with the following definition.

- Width: `16`
- Polynomial: `0x3D65`
- Initial Value: `0x0000`
- Reflect In: `True`
- Reflect Out: `True`
- XOR Out: `0xFFFF`

CRC16_EN13757

Gets the [CrcStandard](#) that defines the **CRC-16/EN-13757** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_EN13757 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/EN-13757** definition.

Remarks

The **CRC-16/EN-13757** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 1 October 2008, Updated: 28 December 2019), with the following definition.

- Width: `16`
- Polynomial: `0x3D65`
- Initial Value: `0x0000`
- Reflect In: `False`
- Reflect Out: `False`
- XOR Out: `0xFFFF`

CRC16_EPC

Gets the [CrcStandard](#) that defines the **CRC-16/GENIBUS** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_EPC { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/EPC** definition.

Remarks

The **CRC16_EPC** is an alias of the [CRC16_GENIBUS](#) standard.

See Also

[CRC16_GENIBUS](#), [CRC16_DARC](#), [CRC16_EPCC1G2](#), [CRC16_ICODE](#)

CRC16_EPCC1G2

Gets the [CrcStandard](#) that defines the **CRC-16/GENIBUS** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_EPCC1G2 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/EPC-C1G2** definition.

Remarks

The **CRC16_EPCC1G2** is an alias of the [CRC16_GENIBUS](#) standard.

See Also

[CRC16_GENIBUS](#), [CRC16_DARC](#), [CRC16_EPC](#), [CRC16_ICODE](#)

CRC16_GENIBUS

Gets the [CrcStandard](#) that defines the **CRC-16/GENIBUS** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_GENIBUS { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/GENIBUS** definition.

Remarks

The **CRC-16/GENIBUS** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 2 November 2007, Updated: 3 January 2021), with the following definition.

- Width: **16**
- Polynomial: **0x1021**
- Initial Value: **0xFFFF**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0xFFFF**

The **CRC-16/GENIBUS** standard is also known by the aliases **CRC-16/DARC**, **CRC-16/EPC**, **CRC-16/EPC-C1G2**, and **CRC-16/I-CODE**.

See Also

[CRC16_DARC](#), [CRC16_EPC](#), [CRC16_EPCC1G2](#), [CRC16_ICODE](#)

CRC16_GSM

Gets the [CrcStandard](#) that defines the **CRC-16/GSM** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_GSM { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/GSM** definition.

Remarks

The **CRC-16/GSM** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 17 February 2017, Updated: 19 April 2019), with the following definition.

- Width: **16**
- Polynomial: **0x1021**
- Initial Value: **0x0000**
- Reflect In: **False**
- Reflect Out: **False**

- XOR Out: `0xFFFF`

CRC16_IBM3740

Gets the [CrcStandard](#) that defines the **CRC-16/IBM-3740** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_IBM3740 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/IBM-3740** definition.

Remarks

The **CRC-16/IBM-3740** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 30 March 2005, Updated: 7 May 2022), with the following definition.

- Width: `16`
- Polynomial: `0x1021`
- Initial Value: `0xFFFF`
- Reflect In: `False`
- Reflect Out: `False`
- XOR Out: `0x0000`

The **CRC-16/IBM-3740** standard is also known by the aliases **CRC-16/AUTOSAR**, and **CRC-16/CCITT-FALSE**.

See Also

[CRC16_AUTOSAR](#), [CRC16_CCITTFALSE](#)

CRC16_IBMSDLC

Gets the [CrcStandard](#) that defines the **CRC-16/IBM-SDLC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_IBMSDLC { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/IBM-SDLC** definition.

Remarks

The **CRC-16/IBM-SDLC** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 10 September 2005, Updated: 2 January 2021), with the following definition.

- Width: **16**
- Polynomial: **0x1021**
- Initial Value: **0xFFFF**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0xFFFF**

The **CRC-16/IBM-SDLC** standard is also known by the aliases **CRC-16/ISO-HDLC**, **CRC-16/ISO-IEC-14443-3-B**, **CRC-16/X-25**, **CRC-B**, and **X-25**.

See Also

[CRC16_ISOHDLC](#), [CRC16_ISOIEC144433B](#), [CRC16_X25](#), [CRCB](#), [X25](#)

CRC16_ICODE

Gets the [CrcStandard](#) that defines the **CRC-16/GENIBUS** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_ICODE { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/I-CODE** definition.

Remarks

The **CRC16_ICODE** is an alias of the [CRC16_GENIBUS](#) standard.

See Also

[CRC16_GENIBUS](#), [CRC16_DARC](#), [CRC16_EPC](#), [CRC16_EPCC1G2](#)

CRC16_IEC611582

Gets the [CrcStandard](#) that defines the **CRC-16/PROFIBUS** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_IEC611582 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/IEC-61158-2** definition.

Remarks

The **CRC16_IEC611582** is an alias of the [CRC16_PROFIBUS](#) standard.

See Also

[CRC16_PROFIBUS](#)

CRC16_ISOHDLC

Gets the [CrcStandard](#) that defines the **CRC-16/IBM-SDLC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_ISOHDLC { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/ISO-HDLC** definition.

Remarks

The **CRC16_ISOHDLC** is an alias of the [CRC16_IBMSDLC](#) standard.

See Also

[CRC16_IBMSDLC](#), [CRC16_ISOIEC144433B](#), [CRC16_X25](#), [CRCB_X25](#)

CRC16_ISOIEC144433A

Gets the [CrcStandard](#) that defines the [CRC-16/ISO-IEC-14443-3-A](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_ISOIEC144433A { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-16/ISO-IEC-14443-3-A](#) definition.

Remarks

The [CRC-16/ISO-IEC-14443-3-A](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 30 April 2011, Updated: 29 December 2021), with the following definition.

- Width: [16](#)
- Polynomial: [0x1021](#)
- Initial Value: [0xC6C6](#)
- Reflect In: [True](#)
- Reflect Out: [True](#)
- XOR Out: [0x0000](#)

The [CRC-16/ISO-IEC-14443-3-A](#) standard is also known by the alias [CRC-A](#).

See Also

[CRCA](#)

CRC16_ISOIEC144433B

Gets the [CrcStandard](#) that defines the [CRC-16/IBM-SDLC](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_ISOIEC144433B { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-16/ISO-IEC-14443-3-B](#) definition.

Remarks

The [CRC16_ISOIEC144433B](#) is an alias of the [CRC16_IBMSDLC](#) standard.

See Also

[CRC16_IBMSDLC](#), [CRC16_ISOHDLC](#), [CRC16_X25](#), [CRCB](#), [X25](#)

CRC16_KERMIT

Gets the [CrcStandard](#) that defines the [CRC-16/KERMIT](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_KERMIT { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-16/KERMIT](#) definition.

Remarks

The [CRC-16/KERMIT](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 10 September 2005, Updated: 23 December 2021), with the following definition.

- Width: [16](#)
- Polynomial: [0x1021](#)
- Initial Value: [0x0000](#)
- Reflect In: [True](#)
- Reflect Out: [True](#)
- XOR Out: [0x0000](#)

The [CRC-16/KERMIT](#) standard is also known by the aliases [CRC-16/CCITT](#), [CRC-16/CCITT-TRUE](#), [CRC-16/V-41-LSB](#), [CRC-CCITT](#), and [KERMIT](#).

See Also

[CRC16_CCITT](#), [CRC16_CCITTRUE](#), [CRC16_V41LSB](#), [CRCCCITT](#), [KERMIT](#)

CRC16_LHA

Gets the [CrcStandard](#) that defines the [CRC-16/ARC](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_LHA { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-16/LHA](#) definition.

Remarks

The [CRC16_LHA](#) is an alias of the [CRC16_ARC](#) standard.

See Also

[CRC16_ARC](#), [ARC](#), [CRC16](#), [CRCIBM](#)

CRC16_LJ1200

Gets the [CrcStandard](#) that defines the [CRC-16/LJ1200](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_LJ1200 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-16/LJ1200](#) definition.

Remarks

NOTE

The parameters for the [CRC-16/LJ1200](#) standard are not widely tested or confirmed.

The [CRC-16/LJ1200](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 20 February 2016, Updated: 6 June 2018), with the following definition.

- Width: [16](#)
- Polynomial: [0x6F63](#)
- Initial Value: [0x0000](#)

- Reflect In: `False`
- Reflect Out: `False`
- XOR Out: `0x0000`

CRC16_LTE

Gets the [CrcStandard](#) that defines the **CRC-16/XMODEM** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_LTE { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/LTE** definition.

Remarks

The **CRC16_LTE** is an alias of the [CRC16_XMODEM](#) standard.

See Also

[CRC16_XMODEM](#), [CRC16_ACORN](#), [CRC16_V41MSB](#), [XMODEM](#), [ZMODEM](#)

CRC16_M17

Gets the [CrcStandard](#) that defines the **CRC-16/M17** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_M17 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/M17** definition.

Remarks

The **CRC-16/M17** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 7 May 2022), with the following definition.

- Width: `16`
- Polynomial: `0x5935`
- Initial Value: `0xFFFF`
- Reflect In: `False`
- Reflect Out: `False`
- XOR Out: `0x0000`

CRC16_MAXIM

Gets the [CrcStandard](#) that defines the **CRC-16/MAXIM-DOW** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_MAXIM { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/MAXIM** definition.

Remarks

The `CRC16_MAXIM` is an alias of the [CRC16_MAXIMDOW](#) standard.

See Also

[CRC16_MAXIMDOW](#)

CRC16_MAXIMDOW

Gets the [CrcStandard](#) that defines the **CRC-16/MAXIM-DOW** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_MAXIMDOW { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/MAXIM-DOW** definition.

Remarks

The **CRC-16/MAXIM-DOW** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 26 April 2009, Updated: 28 December 2019), with the following definition.

- Width: **16**
- Polynomial: **0x8005**
- Initial Value: **0x0000**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0xFFFF**

The **CRC-16/MAXIM-DOW** standard is also known by the alias **CRC-16/MAXIM**.

See Also

[CRC16_MAXIM](#)

CRC16_MCRF4XX

Gets the [CrcStandard](#) that defines the **CRC-16/MCRF4XX** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_MCRF4XX { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/MCRF4XX** definition.

Remarks

The **CRC-16/MCRF4XX** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 18 January 2008, Updated: 2 January 2021), with the following definition.

- Width: **16**
- Polynomial: **0x1021**
- Initial Value: **0xFFFF**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0x0000**

CRC16_MODBUS

Gets the [CrcStandard](#) that defines the **CRC-16/MODBUS** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_MODBUS { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/MODBUS** definition.

Remarks

The **CRC-16/MODBUS** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 27 February 2007, Updated: 2 January 2021), with the following definition.

- Width: **16**
- Polynomial: **0x8005**
- Initial Value: **0xFFFF**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0x0000**

The **CRC-16/MODBUS** standard is also known by the alias **MODBUS**.

See Also

[MODBUS](#)

CRC16_NRSC5

Gets the [CrcStandard](#) that defines the **CRC-16/NRSC-5** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_NRSC5 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/NRSC-5** definition.

Remarks

The [CRC-16/NRSC-5](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 16 November 2018, Updated: 20 March 2019), with the following definition.

- Width: [16](#)
- Polynomial: [0x080B](#)
- Initial Value: [0xFFFF](#)
- Reflect In: [True](#)
- Reflect Out: [True](#)
- XOR Out: [0x0000](#)

CRC16_OPENSAFETYA

Gets the [CrcStandard](#) that defines the [CRC-16/OPENSafety-A](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_OPENSAFETYA { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-16/OPENSafety-A](#) definition.

Remarks

The [CRC-16/OPENSafety-A](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 22 July 2016, Updated: 7 May 2022), with the following definition.

- Width: [16](#)
- Polynomial: [0x5935](#)
- Initial Value: [0x0000](#)
- Reflect In: [False](#)
- Reflect Out: [False](#)
- XOR Out: [0x0000](#)

CRC16_OPENSAFETYB

Gets the [CrcStandard](#) that defines the [CRC-16/OPENSafety-B](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_OPENSAFETYB { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-16/OPENSafety-B** definition.

Remarks

The **CRC-16/OPENSafety-B** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 22 July 2016, Updated: 7 May 2022), with the following definition.

- Width: **16**
- Polynomial: **0x755B**
- Initial Value: **0x0000**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x0000**

CRC16_PROFIBUS

Gets the [CrcStandard](#) that defines the **CRC-16/PROFIBUS** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_PROFIBUS { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-16/PROFIBUS** definition.

Remarks

The **CRC-16/PROFIBUS** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 5 July 2016, Updated: 7 May 2022), with the following definition.

- Width: **16**
- Polynomial: **0x1DCF**
- Initial Value: **0xFFFF**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0xFFFF**

The **CRC-16/PROFIBUS** standard is also known by the alias **CRC-16/IEC-61158-2**.

See Also

[CRC16 IEC611582](#)

CRC16_RIELLO

Gets the [CrcStandard](#) that defines the **CRC-16/RIELLO** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_RIELLO { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/RIELLO** definition.

Remarks

 **NOTE**

The parameters for the **CRC-16/RIELLO** standard are not widely tested or confirmed.

The **CRC-16/RIELLO** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 18 April 2009, Updated: 28 December 2019), with the following definition.

- Width: **16**
- Polynomial: **0x1021**
- Initial Value: **0XB2AA**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0x0000**

CRC16_SPIFUJITSU

Gets the [CrcStandard](#) that defines the **CRC-16/SPI-FUJITSU** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_SPIFUJITSU { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-16/SPI-FUJITSU** definition.

Remarks

The **CRC-16/SPI-FUJITSU** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 30 October 2007, Updated: 28 December 2019), with the following definition.

- Width: **16**
- Polynomial: **0x1021**
- Initial Value: **0x1D0F**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x0000**

The **CRC-16/SPI-FUJITSU** standard is also known by the alias **CRC-16/AUG-CCITT**.

See Also

[CRC16_AUGCCITT](#)

CRC16_T10DIF

Gets the [CrcStandard](#) that defines the **CRC-16/T10-DIF** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_T10DIF { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-16/T10-DIF** definition.

Remarks

The **CRC-16/T10-DIF** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 29 September 2009, Updated: 2 January 2021), with the following definition.

- Width: **16**
- Polynomial: **0x8BB7**

- Initial Value: `0x0000`
- Reflect In: `False`
- Reflect Out: `False`
- XOR Out: `0x0000`

CRC16_TELEDISK

Gets the [CrcStandard](#) that defines the **CRC-16/TELEDISK** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_TELEDISK { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/TELEDISK** definition.

Remarks

The **CRC-16/TELEDISK** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 10 December 2009, Updated: 6 June 2018), with the following definition.

- Width: `16`
- Polynomial: `0xA097`
- Initial Value: `0x0000`
- Reflect In: `False`
- Reflect Out: `False`
- XOR Out: `0x0000`

CRC16_TMS37157

Gets the [CrcStandard](#) that defines the **CRC-16/TMS37157** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_TMS37157 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/TMS37157** definition.

Remarks

The **CRC-16/TMS37157** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 17 April 2011, Updated: 28 December 2019), with the following definition.

- Width: **16**
- Polynomial: **0x1021**
- Initial Value: **0x89EC**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0x0000**

CRC16_UMTS

Gets the [CrcStandard](#) that defines the **CRC-16/UMTS** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_UMTS { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/UMTS** definition.

Remarks

The **CRC-16/UMTS** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 4 June 2008, Updated: 7 May 2022), with the following definition.

- Width: **16**
- Polynomial: **0x8005**
- Initial Value: **0x0000**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x0000**

The **CRC-16/UMTS** standard is also known by the aliases **CRC-16/BUYPASS**, and **CRC-16/VERIFONE**.

See Also

[CRC16_BUYPASS](#), [CRC16_VERIFONE](#)

CRC16_USB

Gets the [CrcStandard](#) that defines the **CRC-16/USB** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_USB { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/USB** definition.

Remarks

NOTE

The parameters for the **CRC-16/USB** standard are not widely tested or confirmed.

The **CRC-16/USB** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 1 December 2007, Updated: 6 February 2017), with the following definition.

- Width: **16**
- Polynomial: **0x8005**
- Initial Value: **0xFFFF**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0xFFFF**

CRC16_V41LSB

Gets the [CrcStandard](#) that defines the **CRC-16/KERMIT** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_V41LSB { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/V-41-LSB** definition.

Remarks

The [CRC16_V41LSB](#) is an alias of the [CRC16_KERMIT](#) standard.

See Also

[CRC16_KERMIT](#), [CRC16_CCITT](#), [CRC16_CCITTTRUE](#), [CRCCCITT](#), [KERMIT](#)

CRC16_V41MSB

Gets the [CrcStandard](#) that defines the [CRC-16/XMODEM](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_V41MSB { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-16/V-41-MSB](#) definition.

Remarks

The [CRC16_V41MSB](#) is an alias of the [CRC16_XMODEM](#) standard.

See Also

[CRC16_XMODEM](#), [CRC16_ACORN](#), [CRC16_LTE](#), [XMODEM](#), [ZMODEM](#)

CRC16_VERIFONE

Gets the [CrcStandard](#) that defines the [CRC-16/UMTS](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_VERIFONE { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-16/VERIFONE](#) definition.

Remarks

The **CRC16_VERIFONE** is an alias of the [CRC16_UMTS](#) standard.

See Also

[CRC16_UMTS](#), [CRC16_BUYPASS](#)

CRC16_X25

Gets the [CrcStandard](#) that defines the **CRC-16/IBM-SDLC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_X25 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/X-25** definition.

Remarks

The **CRC16_X25** is an alias of the [CRC16_IBMSDLC](#) standard.

See Also

[CRC16_IBMSDLC](#), [CRC16_ISOHDLC](#), [CRC16_ISOIEC144433B](#), [CRCB_X25](#)

CRC16_XMODEM

Gets the [CrcStandard](#) that defines the **CRC-16/XMODEM** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC16_XMODEM { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-16/XMODEM** definition.

Remarks

The **CRC-16/XMODEM** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 30 March 2005, Updated: 7 May 2022), with the following definition.

- Width: `16`
- Polynomial: `0x1021`
- Initial Value: `0x0000`
- Reflect In: `False`
- Reflect Out: `False`
- XOR Out: `0x0000`

The **CRC-16/XMODEM** standard is also known by the aliases **CRC-16/ACORN**, **CRC-16/LTE**, **CRC-16/V-41-MSB**, **XMODEM**, and **ZMODEM**.

See Also

[CRC16_ACORN](#), [CRC16_LTE](#), [CRC16_V41MSB](#), [XMODEM](#), [ZMODEM](#)

CRC17_CANFD

Gets the [CrcStandard](#) that defines the **CRC-17/CAN-FD** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC17_CANFD { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-17/CAN-FD** definition.

Remarks

NOTE

The parameters for the **CRC-17/CAN-FD** standard are not widely tested or confirmed.

The **CRC-17/CAN-FD** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 20 June 2017, Updated: 3 January 2021), with the following definition.

- Width: `17`
- Polynomial: `0x0001685B`
- Initial Value: `0x00000000`
- Reflect In: `False`
- Reflect Out: `False`
- XOR Out: `0x00000000`

CRC21_CANFD

Gets the [CrcStandard](#) that defines the **CRC-21/CAN-FD** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC21_CANFD { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-21/CAN-FD** definition.

Remarks

NOTE

The parameters for the **CRC-21/CAN-FD** standard are not widely tested or confirmed.

The **CRC-21/CAN-FD** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 20 June 2017, Updated: 3 January 2021), with the following definition.

- Width: **21**
- Polynomial: **0x00102899**
- Initial Value: **0x00000000**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x00000000**

CRC24

Gets the [CrcStandard](#) that defines the **CRC-24/OPENPGP** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC24 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-24** definition.

Remarks

The **CRC24** is an alias of the [CRC24_OPENPGP](#) standard.

See Also

[CRC24_OPENPGP](#)

CRC24_BLE

Gets the [CrcStandard](#) that defines the **CRC-24/BLE** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC24_BLE { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-24/BLE** definition.

Remarks

The **CRC-24/BLE** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 7 February 2016, Updated: 7 May 2022), with the following definition.

- Width: **24**
- Polynomial: **0x0000065B**
- Initial Value: **0x00555555**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0x00000000**

CRC24_FLEXRAYA

Gets the [CrcStandard](#) that defines the **CRC-24/FLEXRAY-A** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC24_FLEXRAYA { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-24/FLEXRAY-A** definition.

Remarks

The **CRC-24/FLEXRAY-A** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 3 November 2007, Updated: 6 February 2017), with the following definition.

- Width: **24**
- Polynomial: **0x005D6DCB**
- Initial Value: **0x00FEDCBA**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x00000000**

CRC24_FLEXRAYB

Gets the [CrcStandard](#) that defines the **CRC-24/FLEXRAY-B** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC24_FLEXRAYB { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-24/FLEXRAY-B** definition.

Remarks

The **CRC-24/FLEXRAY-B** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 3 November 2007, Updated: 6 February 2017), with the following definition.

- Width: **24**
- Polynomial: **0x005D6DCB**
- Initial Value: **0x00ABCDEF**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x00000000**

CRC24_INTERLAKEN

Gets the [CrcStandard](#) that defines the **CRC-24/INTERLAKEN** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC24_INTERLAKEN { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-24/INTERLAKEN** definition.

Remarks

NOTE

The parameters for the **CRC-24/INTERLAKEN** standard are not widely tested or confirmed.

The **CRC-24/INTERLAKEN** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 10 February 2016, Updated: 28 December 2019), with the following definition.

- Width: **24**
- Polynomial: **0x00328B63**
- Initial Value: **0x00FFFFFF**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x00FFFFFF**

CRC24_LTEA

Gets the [CrcStandard](#) that defines the **CRC-24/LTE-A** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC24_LTEA { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-24/LTE-A** definition.

Remarks

NOTE

The parameters for the **CRC-24/LTE-A** standard are not widely tested or confirmed.

The **CRC-24/LTE-A** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 13 February 2016, Updated: 7 May 2022), with the following definition.

- Width: **24**
- Polynomial: **0x00864CFB**
- Initial Value: **0x00000000**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x00000000**

CRC24_LTEB

Gets the [CrcStandard](#) that defines the **CRC-24/LTE-B** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC24_LTEB { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-24/LTE-B** definition.

Remarks

NOTE

The parameters for the **CRC-24/LTE-B** standard are not widely tested or confirmed.

The **CRC-24/LTE-B** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 13 February 2016, Updated: 7 May 2022), with the following definition.

- Width: **24**
- Polynomial: **0x00800063**
- Initial Value: **0x00000000**

- Reflect In: `False`
- Reflect Out: `False`
- XOR Out: `0x00000000`

CRC24_OPENPGP

Gets the [CrcStandard](#) that defines the **CRC-24/OPENPGP** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC24_OPENPGP { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-24/OPENPGP** definition.

Remarks

The **CRC-24/OPENPGP** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 2 November 2007, Updated: 25 March 2019), with the following definition.

- Width: `24`
- Polynomial: `0x00864CFB`
- Initial Value: `0x00B704CE`
- Reflect In: `False`
- Reflect Out: `False`
- XOR Out: `0x00000000`

The **CRC-24/OPENPGP** standard is also known by the alias **CRC-24**.

See Also

[CRC24](#)

CRC24_OS9

Gets the [CrcStandard](#) that defines the **CRC-24/OS-9** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC24_OS9 { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the [CRC-24/OS-9](#) definition.

Remarks

The [CRC-24/OS-9](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 25 July 2018, Updated: 28 December 2019), with the following definition.

- Width: [24](#)
- Polynomial: [0x00800063](#)
- Initial Value: [0x00FFFFFF](#)
- Reflect In: [False](#)
- Reflect Out: [False](#)
- XOR Out: [0x00FFFFFF](#)

CRC30_CDMA

Gets the [CrcStandard](#) that defines the [CRC-30/CDMA](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC30_CDMA { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the [CRC-30/CDMA](#) definition.

Remarks

(i) NOTE

The parameters for the [CRC-30/CDMA](#) standard are not widely tested or confirmed.

The [CRC-30/CDMA](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 20 February 2016, Updated: 3 January 2021), with the following definition.

- Width: [30](#)

- Polynomial: `0x2030B9C7`
- Initial Value: `0xFFFFFFFF`
- Reflect In: `False`
- Reflect Out: `False`
- XOR Out: `0xFFFFFFFF`

CRC31_PHILIPS

Gets the [CrcStandard](#) that defines the **CRC-31/PHILIPS** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC31_PHILIPS { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-31/PHILIPS** definition.

Remarks

The **CRC-31/PHILIPS** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 12 April 2012, Updated: 28 December 2019), with the following definition.

- Width: `31`
- Polynomial: `0x04C11DB7`
- Initial Value: `0x7FFFFFFF`
- Reflect In: `False`
- Reflect Out: `False`
- XOR Out: `0x7FFFFFFF`

CRC32

Gets the [CrcStandard](#) that defines the **CRC-32/ISO-HDLC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32** definition.

Remarks

The **CRC32** is an alias of the [CRC32 ISOHDLC](#) standard.

See Also

[CRC32 ISOHDLC](#), [CRC32 ADCCP](#), [CRC32 V42](#), [CRC32 XZ](#), [PKZIP](#)

CRC32C

Gets the [CrcStandard](#) that defines the **CRC-32/ISCSI** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32C { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32C** definition.

Remarks

The **CRC32C** is an alias of the [CRC32 ISCSI](#) standard.

See Also

[CRC32 ISCSI](#), [CRC32 BASE91C](#), [CRC32 CASTAGNOLI](#), [CRC32 INTERLAKEN](#)

CRC32D

Gets the [CrcStandard](#) that defines the **CRC-32/BASE91-D** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32D { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32D** definition.

Remarks

The **CRC32D** is an alias of the [CRC32_BASE91D](#) standard.

See Also

[CRC32_BASE91D](#)

CRC32Q

Gets the [CrcStandard](#) that defines the **CRC-32/AIXM** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32Q { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32Q** definition.

Remarks

The **CRC32Q** is an alias of the [CRC32_AIXM](#) standard.

See Also

[CRC32_AIXM](#)

CRC32_AAL5

Gets the [CrcStandard](#) that defines the **CRC-32/BZIP2** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_AAL5 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32/AAL5** definition.

Remarks

The [CRC32_AAL5](#) is an alias of the [CRC32_BZIP2](#) standard.

See Also

[CRC32_BZIP2](#), [CRC32_DECTB](#), [BCRC32](#)

CRC32_ADCCP

Gets the [CrcStandard](#) that defines the [CRC-32/ISO-HDLC](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_ADCCP { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-32/ADCCP](#) definition.

Remarks

The [CRC32_ADCCP](#) is an alias of the [CRC32_ISOHDLC](#) standard.

See Also

[CRC32_ISOHDLC](#), [CRC32](#), [CRC32_V42](#), [CRC32_XZ](#), [PKZIP](#)

CRC32_AIXM

Gets the [CrcStandard](#) that defines the [CRC-32/AIXM](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_AIXM { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-32/AIXM](#) definition.

Remarks

The [CRC-32/AIXM](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 29 April 2009, Updated: 2 January 2021), with the following definition.

- Width: 32
- Polynomial: 0x814141AB
- Initial Value: 0x00000000
- Reflect In: False
- Reflect Out: False
- XOR Out: 0x00000000

The [CRC-32/AIXM](#) standard is also known by the alias [CRC-32Q](#).

See Also

[CRC32Q](#)

CRC32_AUTOSAR

Gets the [CrcStandard](#) that defines the [CRC-32/AUTOSAR](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_AUTOSAR { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-32/AUTOSAR](#) definition.

Remarks

The [CRC-32/AUTOSAR](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 24 July 2016, Updated: 7 May 2022), with the following definition.

- Width: 32
- Polynomial: 0xF4ACFB13
- Initial Value: 0xFFFFFFFF
- Reflect In: True
- Reflect Out: True
- XOR Out: 0xFFFFFFFF

CRC32_BASE91C

Gets the [CrcStandard](#) that defines the [CRC-32/ISCSI](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_BASE91C { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32/BASE91-C** definition.

Remarks

The **CRC32_BASE91C** is an alias of the [CRC32_ISCSI](#) standard.

See Also

[CRC32_ISCSI](#), [CRC32_CASTAGNOLI](#), [CRC32_INTERLAKEN](#), [CRC32C](#)

CRC32_BASE91D

Gets the [CrcStandard](#) that defines the **CRC-32/BASE91-D** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_BASE91D { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32/BASE91-D** definition.

Remarks

The **CRC-32/BASE91-D** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 28 November 2008, Updated: 21 March 2019), with the following definition.

- Width: **32**
- Polynomial: **0xA833982B**
- Initial Value: **0xFFFFFFFF**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0xFFFFFFFF**

The **CRC-32/BASE91-D** standard is also known by the alias **CRC-32D**.

See Also

[CRC32D](#)

CRC32_BZIP2

Gets the [CrcStandard](#) that defines the **CRC-32/BZIP2** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_BZIP2 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32/BZIP2** definition.

Remarks

The **CRC-32/BZIP2** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 8 October 2008, Updated: 20 March 2019), with the following definition.

- Width: **32**
- Polynomial: **0x04C11DB7**
- Initial Value: **0xFFFFFFFF**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0xFFFFFFFF**

The **CRC-32/BZIP2** standard is also known by the aliases **CRC-32/AAL5**, **CRC-32/DECT-B**, and **B-CRC-32**.

See Also

[CRC32_AAL5](#), [CRC32_DECTB](#), [BCRC32](#)

CRC32_CASTAGNOLI

Gets the [CrcStandard](#) that defines the **CRC-32/ISCSI** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_CASTAGNOLI { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32/CASTAGNOLI** definition.

Remarks

The **CRC32_CASTAGNOLI** is an alias of the [CRC32_ISCSI](#) standard.

See Also

[CRC32_ISCSI](#), [CRC32_BASE91C](#), [CRC32_INTERLAKEN](#), [CRC32C](#)

CRC32_CDROMEDC

Gets the [CrcStandard](#) that defines the **CRC-32/CD-ROM-EDC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_CDROMEDC { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32/CD-ROM-EDC** definition.

Remarks

NOTE

The parameters for the **CRC-32/CD-ROM-EDC** standard are not widely tested or confirmed.

The **CRC-32/CD-ROM-EDC** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 19 April 2019, Updated: 7 May 2022), with the following definition.

- Width: **32**
- Polynomial: **0x8001801B**
- Initial Value: **0x00000000**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0x00000000**

CRC32_CKSUM

Gets the [CrcStandard](#) that defines the **CRC-32/CKSUM** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_CKSUM { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32/CKSUM** definition.

Remarks

The **CRC-32/CKSUM** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 30 March 2005, Updated: 7 May 2022), with the following definition.

- Width: **32**
- Polynomial: **0x04C11DB7**
- Initial Value: **0x00000000**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0xFFFFFFFF**

The **CRC-32/CKSUM** standard is also known by the aliases **CKSUM**, and **CRC-32/POSIX**.

See Also

[CKSUM](#), [CRC32_POSIX](#)

CRC32_DECTB

Gets the [CrcStandard](#) that defines the **CRC-32/BZIP2** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_DECTB { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32/DECT-B** definition.

Remarks

The **CRC32_DECTB** is an alias of the [CRC32_BZIP2](#) standard.

See Also

[CRC32_BZIP2](#), [CRC32_AAL5](#), [BCRC32](#)

CRC32_INTERLAKEN

Gets the [CrcStandard](#) that defines the **CRC-32/ISCSI** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_INTERLAKEN { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32/INTERLAKEN** definition.

Remarks

The **CRC32_INTERLAKEN** is an alias of the [CRC32_ISCSI](#) standard.

See Also

[CRC32_ISCSI](#), [CRC32_BASE91C](#), [CRC32_CASTAGNOLI](#), [CRC32C](#)

CRC32_ISCSI

Gets the [CrcStandard](#) that defines the **CRC-32/ISCSI** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_ISCSI { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32/ISCSI** definition.

Remarks

The **CRC-32/ISCSI** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 10 September 2005, Updated: 28 December 2019), with the following definition.

- Width: 32
- Polynomial: 0x1EDC6F41
- Initial Value: 0xFFFFFFFF
- Reflect In: True
- Reflect Out: True
- XOR Out: 0xFFFFFFFF

The **CRC-32/ISCSI** standard is also known by the aliases **CRC-32/BASE91-C**, **CRC-32/CASTAGNOLI**, **CRC-32/INTERLAKEN**, and **CRC-32C**.

See Also

[CRC32_BASE91C](#), [CRC32_CASTAGNOLI](#), [CRC32_INTERLAKEN](#), [CRC32C](#)

CRC32_ISOHDLC

Gets the [CrcStandard](#) that defines the **CRC-32/ISO-HDLC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_ISOHDLC { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32/ISO-HDLC** definition.

Remarks

The **CRC-32/ISO-HDLC** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 30 March 2005, Updated: 7 May 2022), with the following definition.

- Width: 32
- Polynomial: 0x04C11DB7
- Initial Value: 0xFFFFFFFF
- Reflect In: True
- Reflect Out: True
- XOR Out: 0xFFFFFFFF

The **CRC-32/ISO-HDLC** standard is also known by the aliases **CRC-32**, **CRC-32/ADCCP**, **CRC-32/V-42**, **CRC-32/XZ**, and **PKZIP**.

See Also

[CRC32](#), [CRC32 ADCCP](#), [CRC32 V42](#), [CRC32 XZ](#), [PKZIP](#)

CRC32_JAMCRC

Gets the [CrcStandard](#) that defines the **CRC-32/JAMCRC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_JAMCRC { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32/JAMCRC** definition.

Remarks

The **CRC-32/JAMCRC** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 30 March 2005, Updated: 21 November 2018), with the following definition.

- Width: **32**
- Polynomial: **0x04C11DB7**
- Initial Value: **0xFFFFFFFF**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0x00000000**

The **CRC-32/JAMCRC** standard is also known by the alias **JAMCRC**.

See Also

[JAMCRC](#)

CRC32_MEF

Gets the [CrcStandard](#) that defines the **CRC-32/MEF** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_MEF { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-32/MEF** definition.

Remarks

The **CRC-32/MEF** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 10 January 2022), with the following definition.

- Width: **32**
- Polynomial: **0x741B8CD7**
- Initial Value: **0xFFFFFFFF**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0x00000000**

CRC32_MPEG2

Gets the [CrcStandard](#) that defines the **CRC-32/MPEG-2** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_MPEG2 { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-32/MPEG-2** definition.

Remarks

The **CRC-32/MPEG-2** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 8 October 2008, Updated: 2 January 2021), with the following definition.

- Width: **32**
- Polynomial: **0x04C11DB7**
- Initial Value: **0xFFFFFFFF**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x00000000**

CRC32_POSIX

Gets the [CrcStandard](#) that defines the **CRC-32/CKSUM** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_POSIX { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32/POSIX** definition.

Remarks

The **CRC32_POSIX** is an alias of the [CRC32_CKSUM](#) standard.

See Also

[CRC32_CKSUM](#), [CKSUM](#)

CRC32_V42

Gets the [CrcStandard](#) that defines the **CRC-32/ISO-HDLC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_V42 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32/V-42** definition.

Remarks

The **CRC32_V42** is an alias of the [CRC32_ISOHDL](#)C standard.

See Also

[CRC32_ISOHDL](#), [CRC32](#), [CRC32_ADCCP](#), [CRC32_XZ](#), [PKZIP](#)

CRC32_XFER

Gets the [CrcStandard](#) that defines the **CRC-32/XFER** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_XFER { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32/XFER** definition.

Remarks

The **CRC-32/XFER** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 30 March 2005, Updated: 2 January 2021), with the following definition.

- Width: **32**
- Polynomial: **0x0000000AF**
- Initial Value: **0x00000000**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x00000000**

The **CRC-32/XFER** standard is also known by the alias **XFER**.

See Also

[XFER](#)

CRC32_XZ

Gets the [CrcStandard](#) that defines the **CRC-32/ISO-HDLC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC32_XZ { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-32/XZ** definition.

Remarks

The [CRC32_XZ](#) is an alias of the [CRC32_ISOHDL](#)C standard.

See Also

[CRC32_ISOHDL](#), [CRC32](#), [CRC32_ADCCP](#), [CRC32_V42](#), [PKZIP](#)

CRC3_GSM

Gets the [CrcStandard](#) that defines the [CRC-3/GSM](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC3_GSM { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-3/GSM](#) definition.

Remarks

 **NOTE**

The parameters for the [CRC-3/GSM](#) standard are not widely tested or confirmed.

The [CRC-3/GSM](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 17 February 2017), with the following definition.

- Width: 3
- Polynomial: 0x03
- Initial Value: 0x00
- Reflect In: False
- Reflect Out: False
- XOR Out: 0x07

CRC3_ROHC

Gets the [CrcStandard](#) that defines the [CRC-3/ROHC](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC3_ROHC { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-3/ROHC** definition.

Remarks

NOTE

The parameters for the **CRC-3/ROHC** standard are not widely tested or confirmed.

The **CRC-3/ROHC** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 14 December 2009, Updated: 6 February 2017), with the following definition.

- Width: **3**
- Polynomial: **0x03**
- Initial Value: **0x07**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0x00**

CRC40_GSM

Gets the [CrcStandard](#) that defines the **CRC-40/GSM** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC40_GSM { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-40/GSM** definition.

Remarks

NOTE

The parameters for the **CRC-40/GSM** standard are not widely tested or confirmed.

The [CRC-40/GSM](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 10 December 2009, Updated: 25 March 2019), with the following definition.

- Width: [40](#)
- Polynomial: [0x000000004820009](#)
- Initial Value: [0x0000000000000000](#)
- Reflect In: [False](#)
- Reflect Out: [False](#)
- XOR Out: [0x00000FFFFFFFFFFF](#)

CRC4_G704

Gets the [CrcStandard](#) that defines the [CRC-4/G-704](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC4_G704 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-4/G-704](#) definition.

Remarks

NOTE

The parameters for the [CRC-4/G-704](#) standard are not widely tested or confirmed.

The [CRC-4/G-704](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 7 October 2008, Updated: 21 November 2018), with the following definition.

- Width: [4](#)
- Polynomial: [0x03](#)
- Initial Value: [0x00](#)
- Reflect In: [True](#)
- Reflect Out: [True](#)
- XOR Out: [0x00](#)

The [CRC-4/G-704](#) standard is also known by the alias [CRC-4/ITU](#).

See Also

[CRC4_ITU](#)

CRC4_INTERLAKEN

Gets the [CrcStandard](#) that defines the **CRC-4/INTERLAKEN** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC4_INTERLAKEN { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-4/INTERLAKEN** definition.

Remarks

 **NOTE**

The parameters for the **CRC-4/INTERLAKEN** standard are not widely tested or confirmed.

The **CRC-4/INTERLAKEN** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 10 February 2016, Updated: 28 December 2019), with the following definition.

- Width: **4**
- Polynomial: **0x03**
- Initial Value: **0x0F**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x0F**

CRC4_ITU

Gets the [CrcStandard](#) that defines the **CRC-4/G-704** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC4_ITU { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-4/ITU** definition.

Remarks

The **CRC4_ITU** is an alias of the [CRC4_G704](#) standard.

See Also

[CRC4_G704](#)

CRC5_EPC

Gets the [CrcStandard](#) that defines the **CRC-5/EPC-C1G2** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC5_EPC { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-5/EPC** definition.

Remarks

The **CRC5_EPC** is an alias of the [CRC5_EPCC1G2](#) standard.

See Also

[CRC5_EPCC1G2](#)

CRC5_EPCC1G2

Gets the [CrcStandard](#) that defines the **CRC-5/EPC-C1G2** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC5_EPCC1G2 { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-5/EPC-C1G2** definition.

Remarks

The **CRC-5/EPC-C1G2** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 20 May 2009, Updated: 28 December 2019), with the following definition.

- Width: **5**
- Polynomial: **0x09**
- Initial Value: **0x09**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x00**

The **CRC-5/EPC-C1G2** standard is also known by the alias **CRC-5/EPC**.

See Also

[CRC5_EPC](#)

CRC5_G704

Gets the [CrcStandard](#) that defines the **CRC-5/G-704** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC5_G704 { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-5/G-704** definition.

Remarks

NOTE

The parameters for the **CRC-5/G-704** standard are not widely tested or confirmed.

The [CRC-5/G-704](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 7 October 2008, Updated: 21 November 2018), with the following definition.

- Width: 5
- Polynomial: 0x15
- Initial Value: 0x00
- Reflect In: True
- Reflect Out: True
- XOR Out: 0x00

The [CRC-5/G-704](#) standard is also known by the alias [CRC-5/ITU](#).

See Also

[CRC5_ITU](#)

CRC5_ITU

Gets the [CrcStandard](#) that defines the [CRC-5/G-704](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC5_ITU { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-5/ITU](#) definition.

Remarks

The [CRC5_ITU](#) is an alias of the [CRC5_G704](#) standard.

See Also

[CRC5_G704](#)

CRC5_USB

Gets the [CrcStandard](#) that defines the [CRC-5/USB](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC5_USB { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-5/USB** definition.

Remarks

NOTE

The parameters for the **CRC-5/USB** standard are not widely tested or confirmed.

The **CRC-5/USB** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 1 December 2007, Updated: 6 February 2017), with the following definition.

- Width: **5**
- Polynomial: **0x05**
- Initial Value: **0x1F**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0x1F**

CRC64

Gets the [CrcStandard](#) that defines the **CRC-64/ECMA-182** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC64 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-64** definition.

Remarks

The **CRC64** is an alias of the [CRC64 ECMA182](#) standard.

See Also

[CRC64 ECMA182](#)

CRC64_ECMA182

Gets the [CrcStandard](#) that defines the **CRC-64/ECMA-182** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC64_ECMA182 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-64/ECMA-182** definition.

Remarks

NOTE

The parameters for the **CRC-64/ECMA-182** standard are not widely tested or confirmed.

The **CRC-64/ECMA-182** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 6 April 2009, Updated: 7 May 2022), with the following definition.

- Width: **64**
- Polynomial: **0x42F0E1EBA9EA3693**
- Initial Value: **0x0000000000000000**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x0000000000000000**

The **CRC-64/ECMA-182** standard is also known by the alias **CRC-64**.

See Also

[CRC64](#)

CRC64_GOECMA

Gets the [CrcStandard](#) that defines the **CRC-64/XZ** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC64_GOECMA { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-64/GO-ECMA** definition.

Remarks

The **CRC64_GOECMA** is an alias of the [CRC64_XZ](#) standard.

See Also

[CRC64_XZ](#)

CRC64_GOISO

Gets the [CrcStandard](#) that defines the **CRC-64/GO-ISO** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC64_GOISO { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-64/GO-ISO** definition.

Remarks

The **CRC-64/GO-ISO** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 28 January 2017, Updated: 7 May 2022), with the following definition.

- Width: **64**
- Polynomial: **0x0000000000000001B**
- Initial Value: **0xFFFFFFFFFFFFFFFFF**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0xFFFFFFFFFFFFFFFFF**

CRC64_JONES

Gets the [CrcStandard](#) that defines the **CRC-64/JONES** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC64_JONES { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-64/JONES** definition.

Remarks

The **CRC-64/JONES** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 31 December 9999), with the following definition.

- Width: **64**
- Polynomial: **0xAD93D23594C935A9**
- Initial Value: **0xFFFFFFFFFFFFFFF**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0x0000000000000000**

CRC64_MS

Gets the [CrcStandard](#) that defines the **CRC-64/MS** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC64_MS { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-64/MS** definition.

Remarks

The **CRC-64/MS** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 7 January 2022), with the following definition.

- Width: **64**
- Polynomial: **0x259C84CBA6426349**
- Initial Value: **0xFFFFFFFFFFFFFFF**

- Reflect In: `True`
- Reflect Out: `True`
- XOR Out: `0x0000000000000000`

CRC64_WE

Gets the [CrcStandard](#) that defines the **CRC-64/WE** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC64_WE { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-64/WE** definition.

Remarks

The **CRC-64/WE** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 8 November 2009, Updated: 28 December 2019), with the following definition.

- Width: `64`
- Polynomial: `0x42F0E1EBA9EA3693`
- Initial Value: `0xFFFFFFFFFFFFFFFFF`
- Reflect In: `False`
- Reflect Out: `False`
- XOR Out: `0xFFFFFFFFFFFFFFFFF`

CRC64_XZ

Gets the [CrcStandard](#) that defines the **CRC-64/XZ** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC64_XZ { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-64/XZ** definition.

Remarks

The **CRC-64/XZ** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 4 July 2011, Updated: 7 May 2022), with the following definition.

- Width: **64**
- Polynomial: **0x42F0E1EBA9EA3693**
- Initial Value: **0xFFFFFFFFFFFFFFFFF**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0xFFFFFFFFFFFFFFFFF**

The **CRC-64/XZ** standard is also known by the alias **CRC-64/GO-ECMA**.

See Also

[CRC64_GOECMA](#)

CRC6_CDMA2000A

Gets the [CrcStandard](#) that defines the **CRC-6/CDMA2000-A** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC6_CDMA2000A { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-6/CDMA2000-A** definition.

Remarks

The **CRC-6/CDMA2000-A** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 14 October 2013, Updated: 28 December 2019), with the following definition.

- Width: **6**
- Polynomial: **0x27**
- Initial Value: **0x3F**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x00**

CRC6_CDMA2000B

Gets the [CrcStandard](#) that defines the **CRC-6/CDMA2000-B** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC6_CDMA2000B { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-6/CDMA2000-B** definition.

Remarks

NOTE

The parameters for the **CRC-6/CDMA2000-B** standard are not widely tested or confirmed.

The **CRC-6/CDMA2000-B** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 14 October 2013, Updated: 28 December 2019), with the following definition.

- Width: **6**
- Polynomial: **0x07**
- Initial Value: **0x3F**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x00**

CRC6_DARC

Gets the [CrcStandard](#) that defines the **CRC-6/DARC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC6_DARC { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-6/DARC** definition.

Remarks

The **CRC-6/DARC** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 14 December 2009, Updated: 20 November 2018), with the following definition.

- Width: 6
- Polynomial: 0x19
- Initial Value: 0x00
- Reflect In: True
- Reflect Out: True
- XOR Out: 0x00

CRC6_G704

Gets the [CrcStandard](#) that defines the **CRC-6/G-704** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC6_G704 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-6/G-704** definition.

Remarks

NOTE

The parameters for the **CRC-6/G-704** standard are not widely tested or confirmed.

The **CRC-6/G-704** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 7 October 2008, Updated: 21 November 2018), with the following definition.

- Width: 6
- Polynomial: 0x03
- Initial Value: 0x00
- Reflect In: True
- Reflect Out: True
- XOR Out: 0x00

The [CRC-6/G-704](#) standard is also known by the alias [CRC-6/ITU](#).

See Also

[CRC6_ITU](#)

CRC6_GSM

Gets the [CrcStandard](#) that defines the [CRC-6/GSM](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC6_GSM { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-6/GSM](#) definition.

Remarks

 **NOTE**

The parameters for the [CRC-6/GSM](#) standard are not widely tested or confirmed.

The [CRC-6/GSM](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 17 February 2017), with the following definition.

- Width: [6](#)
- Polynomial: [0x2F](#)
- Initial Value: [0x00](#)
- Reflect In: [False](#)
- Reflect Out: [False](#)
- XOR Out: [0x3F](#)

CRC6_ITU

Gets the [CrcStandard](#) that defines the [CRC-6/G-704](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC6_ITU { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-6/ITU** definition.

Remarks

The **CRC6_ITU** is an alias of the [CRC6_G704](#) standard.

See Also

[CRC6_G704](#)

CRC7

Gets the [CrcStandard](#) that defines the **CRC-7/MMC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC7 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-7** definition.

Remarks

The **CRC7** is an alias of the [CRC7_MMC](#) standard.

See Also

[CRC7_MMC](#)

CRC7_MMC

Gets the [CrcStandard](#) that defines the **CRC-7/MMC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC7_MM { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-7/MMC** definition.

Remarks

 **NOTE**

The parameters for the **CRC-7/MMC** standard are not widely tested or confirmed.

The **CRC-7/MMC** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 7 October 2008, Updated: 11 January 2022), with the following definition.

- Width: **7**
- Polynomial: **0x09**
- Initial Value: **0x00**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x00**

The **CRC-7/MMC** standard is also known by the alias **CRC-7**.

See Also

[CRC7](#)

CRC7_ROHC

Gets the [CrcStandard](#) that defines the **CRC-7/ROHC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC7_ROHC { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-7/ROHC** definition.

Remarks

NOTE

The parameters for the **CRC-7/ROHC** standard are not widely tested or confirmed.

The **CRC-7/ROHC** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 14 December 2009, Updated: 6 February 2017), with the following definition.

- Width: **7**
- Polynomial: **0x4F**
- Initial Value: **0x7F**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0x00**

CRC7_UMTS

Gets the [CrcStandard](#) that defines the **CRC-7/UMTS** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC7_UMTS { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-7/UMTS** definition.

Remarks

NOTE

The parameters for the **CRC-7/UMTS** standard are not widely tested or confirmed.

The **CRC-7/UMTS** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 8 February 2016, Updated: 7 May 2022), with the following definition.

- Width: **7**
- Polynomial: **0x45**
- Initial Value: **0x00**

- Reflect In: `False`
- Reflect Out: `False`
- XOR Out: `0x00`

CRC8

Gets the [CrcStandard](#) that defines the **CRC-8/SMBUS** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-8** definition.

Remarks

The **CRC8** is an alias of the [CRC8_SMBUS](#) standard.

See Also

[CRC8_SMBUS](#)

CRC8_AES

Gets the [CrcStandard](#) that defines the **CRC-8/TECH-3250** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_AES { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-8/AES** definition.

Remarks

The **CRC8_AES** is an alias of the [CRC8_TECH3250](#) standard.

See Also

CRC8_AUTOSAR

Gets the [CrcStandard](#) that defines the **CRC-8/AUTOSAR** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_AUTOSAR { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-8/AUTOSAR** definition.

Remarks

The **CRC-8/AUTOSAR** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 24 July 2016, Updated: 7 May 2022), with the following definition.

- Width: 8
- Polynomial: 0x2F
- Initial Value: 0xFF
- Reflect In: False
- Reflect Out: False
- XOR Out: 0xFF

CRC8_BLUETOOTH

Gets the [CrcStandard](#) that defines the **CRC-8/BLUETOOTH** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_BLUETOOTH { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-8/BLUETOOTH** definition.

Remarks

The **CRC-8/BLUETOOTH** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 20 June 2017, Updated: 7 May 2022), with the following definition.

- Width: 8
- Polynomial: **0xA7**
- Initial Value: **0x00**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0x00**

CRC8_CDMA2000

Gets the [CrcStandard](#) that defines the **CRC-8/CDMA2000** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_CDMA2000 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-8/CDMA2000** definition.

Remarks

NOTE

The parameters for the **CRC-8/CDMA2000** standard are not widely tested or confirmed.

The **CRC-8/CDMA2000** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 14 October 2013, Updated: 28 December 2019), with the following definition.

- Width: 8
- Polynomial: **0x9B**
- Initial Value: **0xFF**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x00**

CRC8_DARC

Gets the [CrcStandard](#) that defines the **CRC-8/DARC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_DARC { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-8/DARC** definition.

Remarks

The **CRC-8/DARC** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 14 December 2009, Updated: 29 November 2018), with the following definition.

- Width: **8**
- Polynomial: **0x39**
- Initial Value: **0x00**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0x00**

CRC8_DVBS2

Gets the [CrcStandard](#) that defines the **CRC-8/DVB-S2** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_DVBS2 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-8/DVB-S2** definition.

Remarks

NOTE

The parameters for the **CRC-8/DVB-S2** standard are not widely tested or confirmed.

The **CRC-8/DVB-S2** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 10 January 2014, Updated: 7 May 2022), with the following definition.

- Width: 8
- Polynomial: 0xD5
- Initial Value: 0x00
- Reflect In: False
- Reflect Out: False
- XOR Out: 0x00

CRC8_EBU

Gets the [CrcStandard](#) that defines the **CRC-8/TECH-3250** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_EBU { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-8/EBU** definition.

Remarks

The **CRC8_EBU** is an alias of the [CRC8_TECH3250](#) standard.

See Also

[CRC8_TECH3250](#), [CRC8_AES](#)

CRC8_GSMA

Gets the [CrcStandard](#) that defines the **CRC-8/GSM-A** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_GSMA { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-8/GSM-A** definition.

Remarks

i **NOTE**

The parameters for the **CRC-8/GSM-A** standard are not widely tested or confirmed.

The **CRC-8/GSM-A** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 17 February 2017, Updated: 7 May 2022), with the following definition.

- Width: **8**
- Polynomial: **0x1D**
- Initial Value: **0x00**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x00**

CRC8_GSMB

Gets the [CrcStandard](#) that defines the **CRC-8/GSM-B** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_GSMB { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-8/GSM-B** definition.

Remarks

i **NOTE**

The parameters for the **CRC-8/GSM-B** standard are not widely tested or confirmed.

The [CRC-8/GSM-B](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 17 February 2017), with the following definition.

- Width: 8
- Polynomial: 0x49
- Initial Value: 0x00
- Reflect In: False
- Reflect Out: False
- XOR Out: 0xFF

CRC8_HITAG

Gets the [CrcStandard](#) that defines the [CRC-8/HITAG](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_HITAG { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-8/HITAG](#) definition.

Remarks

The [CRC-8/HITAG](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 23 December 2021), with the following definition.

- Width: 8
- Polynomial: 0x1D
- Initial Value: 0xFF
- Reflect In: False
- Reflect Out: False
- XOR Out: 0x00

CRC8_I4321

Gets the [CrcStandard](#) that defines the [CRC-8/I-432-1](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_I4321 { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-8/I-432-1** definition.

Remarks

NOTE

The parameters for the **CRC-8/I-432-1** standard are not widely tested or confirmed.

The **CRC-8/I-432-1** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 19 April 2009, Updated: 21 November 2018), with the following definition.

- Width: **8**
- Polynomial: **0x07**
- Initial Value: **0x00**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x55**

The **CRC-8/I-432-1** standard is also known by the alias **CRC-8/ITU**.

See Also

[CRC8_ITU](#)

CRC8_ICODE

Gets the [CrcStandard](#) that defines the **CRC-8/I-CODE** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_ICODE { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-8/I-CODE** definition.

Remarks

The **CRC-8/I-CODE** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 3 November 2007, Updated: 6 February 2017), with the following definition.

- Width: 8
- Polynomial: **0x1D**
- Initial Value: **0xFD**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x00**

CRC8_ITU

Gets the [CrcStandard](#) that defines the **CRC-8/I-432-1** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_ITU { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-8/ITU** definition.

Remarks

The **CRC8_ITU** is an alias of the [CRC8_I4321](#) standard.

See Also

[CRC8_I4321](#)

CRC8_LTE

Gets the [CrcStandard](#) that defines the **CRC-8/LTE** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_LTE { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-8/LTE** definition.

Remarks

NOTE

The parameters for the **CRC-8/LTE** standard are not widely tested or confirmed.

The **CRC-8/LTE** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 13 February 2016, Updated: 7 May 2022), with the following definition.

- Width: **8**
- Polynomial: **0x9B**
- Initial Value: **0x00**
- Reflect In: **False**
- Reflect Out: **False**
- XOR Out: **0x00**

CRC8_MAXIM

Gets the [CrcStandard](#) that defines the **CRC-8/MAXIM-DOW** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_MAXIM { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-8/MAXIM** definition.

Remarks

The **CRC8_MAXIM** is an alias of the [CRC8_MAXIMDOW](#) standard.

See Also

[CRC8_MAXIMDOW](#), [DOWCRC](#)

CRC8_MAXIMDOW

Gets the [CrcStandard](#) that defines the **CRC-8/MAXIM-DOW** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_MAXIMDOW { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-8/MAXIM-DOW** definition.

Remarks

The **CRC-8/MAXIM-DOW** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 26 April 2009, Updated: 2 January 2021), with the following definition.

- Width: 8
- Polynomial: **0x31**
- Initial Value: **0x00**
- Reflect In: **True**
- Reflect Out: **True**
- XOR Out: **0x00**

The **CRC-8/MAXIM-DOW** standard is also known by the aliases **CRC-8/MAXIM**, and **DOW-CRC**.

See Also

[CRC8_MAXIM](#), [DOWCRC](#)

CRC8_MIFAREMAD

Gets the [CrcStandard](#) that defines the **CRC-8/MIFARE-MAD** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_MIFAREMAD { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-8/MIFARE-MAD** definition.

Remarks

The **CRC-8/MIFARE-MAD** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 21 February 2019), with the following definition.

- Width: 8
- Polynomial: 0x1D
- Initial Value: 0xC7
- Reflect In: False
- Reflect Out: False
- XOR Out: 0x00

CRC8_NRSC5

Gets the [CrcStandard](#) that defines the **CRC-8/NRSC-5** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_NRSC5 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-8/NRSC-5** definition.

Remarks

The **CRC-8/NRSC-5** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 16 November 2018, Updated: 20 March 2019), with the following definition.

- Width: 8
- Polynomial: 0x31
- Initial Value: 0xFF
- Reflect In: False
- Reflect Out: False
- XOR Out: 0x00

CRC8_OPENSAFETY

Gets the [CrcStandard](#) that defines the **CRC-8/OPENSAFETY** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_OPENSAFETY { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-8/OPENSafety** definition.

Remarks

The **CRC-8/OPENSafety** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 22 July 2016, Updated: 7 May 2022), with the following definition.

- Width: 8
- Polynomial: 0x2F
- Initial Value: 0x00
- Reflect In: False
- Reflect Out: False
- XOR Out: 0x00

CRC8_ROHC

Gets the [CrcStandard](#) that defines the **CRC-8/ROHC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_ROHC { get; }
```

Property Value

CrcStandard

A [CrcStandard](#) object with the properties set to the **CRC-8/ROHC** definition.

Remarks

(i) NOTE

The parameters for the **CRC-8/ROHC** standard are not widely tested or confirmed.

The **CRC-8/ROHC** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 14 December 2009, Updated: 6 February 2017), with the following definition.

- Width: 8

- Polynomial: `0x07`
- Initial Value: `0xFF`
- Reflect In: `True`
- Reflect Out: `True`
- XOR Out: `0x00`

CRC8_SAEJ1850

Gets the [CrcStandard](#) that defines the **CRC-8/SAE-J1850** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_SAEJ1850 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-8/SAE-J1850** definition.

Remarks

The **CRC-8/SAE-J1850** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 17 February 2016, Updated: 7 May 2022), with the following definition.

- Width: `8`
- Polynomial: `0x1D`
- Initial Value: `0xFF`
- Reflect In: `False`
- Reflect Out: `False`
- XOR Out: `0xFF`

CRC8_SMBUS

Gets the [CrcStandard](#) that defines the **CRC-8/SMBUS** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_SMBUS { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-8/SMBUS** definition.

Remarks

The **CRC-8/SMBUS** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 30 March 2005, Updated: 7 May 2022), with the following definition.

- Width: 8
- Polynomial: 0x07
- Initial Value: 0x00
- Reflect In: False
- Reflect Out: False
- XOR Out: 0x00

The **CRC-8/SMBUS** standard is also known by the alias **CRC-8**.

See Also

[CRC8](#)

CRC8_TECH3250

Gets the [CrcStandard](#) that defines the **CRC-8/TECH-3250** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_TECH3250 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-8/TECH-3250** definition.

Remarks

The **CRC-8/TECH-3250** standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 19 July 2012, Updated: 21 November 2018), with the following definition.

- Width: 8
- Polynomial: 0x1D
- Initial Value: 0xFF
- Reflect In: True
- Reflect Out: True
- XOR Out: 0x00

The [CRC-8/TECH-3250](#) standard is also known by the aliases [CRC-8/AES](#), and [CRC-8/EBU](#).

See Also

[CRC8 AES](#), [CRC8 EBU](#)

CRC8_WCDMA

Gets the [CrcStandard](#) that defines the [CRC-8/WCDMA](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRC8_WCDMA { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [CRC-8/WCDMA](#) definition.

Remarks

 **NOTE**

The parameters for the [CRC-8/WCDMA](#) standard are not widely tested or confirmed.

The [CRC-8/WCDMA](#) standard was taken from <http://reveng.sourceforge.net/crc-catalogue/> (Created: 8 November 2009, Updated: 28 December 2019), with the following definition.

- Width: 8
- Polynomial: 0x9B
- Initial Value: 0x00
- Reflect In: True
- Reflect Out: True
- XOR Out: 0x00

CRCA

Gets the [CrcStandard](#) that defines the [CRC-16/ISO-IEC-14443-3-A](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard CRCA { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-A** definition.

Remarks

The **CRCA** is an alias of the [CRC16_ISOIEC144433A](#) standard.

See Also

[CRC16_ISOIEC144433A](#)

CRCB

Gets the [CrcStandard](#) that defines the **CRC-16/IBM-SDLC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRCB { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-B** definition.

Remarks

The **CRCB** is an alias of the [CRC16_IBMSDLC](#) standard.

See Also

[CRC16_IBMSDLC](#), [CRC16_ISOHDLC](#), [CRC16_ISOIEC144433B](#), [CRC16_X25](#), [X25](#)

CRCCCITT

Gets the [CrcStandard](#) that defines the **CRC-16/KERMIT** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRCCCITT { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-CCITT** definition.

Remarks

The **CRCCCITT** is an alias of the [CRC16_KERMIT](#) standard.

See Also

[CRC16_KERMIT](#), [CRC16_CCITT](#), [CRC16_CCITTRUE](#), [CRC16_V41LSB](#), [KERMIT](#)

CRCIBM

Gets the [CrcStandard](#) that defines the **CRC-16/ARC** cyclic redundancy check algorithm standard.

```
public static CrcStandard CRCIBM { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **CRC-IBM** definition.

Remarks

The **CRCIBM** is an alias of the [CRC16_ARC](#) standard.

See Also

[CRC16_ARC](#), [ARC](#), [CRC16](#), [CRC16_LHA](#)

DOWCRC

Gets the [CrcStandard](#) that defines the **CRC-8/MAXIM-DOW** cyclic redundancy check algorithm standard.

```
public static CrcStandard DOWCRC { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **DOW-CRC** definition.

Remarks

The **DOWCRC** is an alias of the [CRC8_MAXIMDOW](#) standard.

See Also

[CRC8_MAXIMDOW](#), [CRC8_MAXIM](#)

InitialValue

Gets the initial value used in the CRC calculation.

```
public ulong InitialValue { get; init; }
```

Property Value

[ulong](#) ↗

The initial value for the CRC calculation.

JAMCRC

Gets the [CrcStandard](#) that defines the **CRC-32/JAMCRC** cyclic redundancy check algorithm standard.

```
public static CrcStandard JAMCRC { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **JAMCRC** definition.

Remarks

The **JAMCRC** is an alias of the [CRC32_JAMCRC](#) standard.

See Also

KERMIT

Gets the [CrcStandard](#) that defines the **CRC-16/KERMIT** cyclic redundancy check algorithm standard.

```
public static CrcStandard KERMIT { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **KERMIT** definition.

Remarks

The **KERMIT** is an alias of the [CRC16_KERMIT](#) standard.

See Also

[CRC16_KERMIT](#), [CRC16_CCITT](#), [CRC16_CCITTRUE](#), [CRC16_V41LSB](#), [CRCCCITT](#)

MODBUS

Gets the [CrcStandard](#) that defines the **CRC-16/MODBUS** cyclic redundancy check algorithm standard.

```
public static CrcStandard MODBUS { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **MODBUS** definition.

Remarks

The **MODBUS** is an alias of the [CRC16_MODBUS](#) standard.

See Also

[CRC16_MODBUS](#)

Name

Gets the name of the CRC standard.

```
public string Name { get; init; }
```

Property Value

[string](#)

The name of the CRC algorithm.

PKZIP

Gets the [CrcStandard](#) that defines the [CRC-32/ISO-HDLC](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard PKZIP { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [PKZIP](#) definition.

Remarks

The [PKZIP](#) is an alias of the [CRC32_ISOHDL](#)C standard.

See Also

[CRC32_ISOHDL](#)C, [CRC32](#), [CRC32_ADCCP](#), [CRC32_V42](#), [CRC32_XZ](#)

Polynomial

Gets the polynomial used in the CRC calculation.

```
public ulong Polynomial { get; init; }
```

Property Value

[ulong](#)

The polynomial value used in the CRC calculation.

RCRC16

Gets the [CrcStandard](#) that defines the **R-CRC-16** cyclic redundancy check algorithm standard.

```
public static CrcStandard RCRC16 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **R-CRC-16** definition.

Remarks

The **RCRC16** is an alias of the [CRC16_DECTR](#) standard.

See Also

[CRC16_DECTR](#)

ReflectIn

Gets a value indicating whether the input data is reflected during the CRC calculation.

```
public bool ReflectIn { get; init; }
```

Property Value

[bool](#)

[true](#) if input data is reflected; otherwise, [false](#).

ReflectOut

Gets a value indicating whether the CRC result is reflected before XORing with [XOrOut](#).

```
public bool ReflectOut { get; init; }
```

Property Value

[bool](#)

[true](#) if the result is reflected; otherwise, [false](#).

Size

Gets the size, in bits, of the CRC checksum.

```
public int Size { get; init; }
```

Property Value

[int](#)

The size of the CRC in bits.

X25

Gets the [CrcStandard](#) that defines the [CRC-16/IBM-SDLC](#) cyclic redundancy check algorithm standard.

```
public static CrcStandard X25 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the [X-25](#) definition.

Remarks

The [X25](#) is an alias of the [CRC16_IBMSDLC](#) standard.

See Also

[CRC16_IBMSDLC](#), [CRC16_ISOHDLC](#), [CRC16_ISOIEC144433B](#), [CRC16_X25](#), [CRCB](#)

XCRC12

Gets the [CrcStandard](#) that defines the **CRC-12/DECT** cyclic redundancy check algorithm standard.

```
public static CrcStandard XCRC12 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **X-CRC-12** definition.

Remarks

The **XCRC12** is an alias of the [CRC12_DECT](#) standard.

See Also

[CRC12_DECT](#)

XCRC16

Gets the [CrcStandard](#) that defines the **CRC-16/DECT-X** cyclic redundancy check algorithm standard.

```
public static CrcStandard XCRC16 { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **X-CRC-16** definition.

Remarks

The **XCRC16** is an alias of the [CRC16_DECTX](#) standard.

See Also

[CRC16_DECTX](#)

XFER

Gets the [CrcStandard](#) that defines the **CRC-32/XFER** cyclic redundancy check algorithm standard.

```
public static CrcStandard XFER { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **XFER** definition.

Remarks

The **XFER** is an alias of the [CRC32_XFER](#) standard.

See Also

[CRC32_XFER](#)

XMODEM

Gets the [CrcStandard](#) that defines the **CRC-16/XMODEM** cyclic redundancy check algorithm standard.

```
public static CrcStandard XMODEM { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **XMODEM** definition.

Remarks

The **XMODEM** is an alias of the [CRC16_XMODEM](#) standard.

See Also

[CRC16_XMODEM](#), [CRC16_ACORN](#), [CRC16_LTE](#), [CRC16_V41MSB](#), [ZMODEM](#)

XOrOut

Gets the value to XOR the final CRC result with.

```
public ulong XOrOut { get; init; }
```

Property Value

[ulong](#)

The XOR value for the final CRC result.

ZMODEM

Gets the [CrcStandard](#) that defines the **CRC-16/XMODEM** cyclic redundancy check algorithm standard.

```
public static CrcStandard ZMODEM { get; }
```

Property Value

[CrcStandard](#)

A [CrcStandard](#) object with the properties set to the **ZMODEM** definition.

Remarks

The **ZMODEM** is an alias of the [CRC16_XMODEM](#) standard.

See Also

[CRC16_XMODEM](#), [CRC16_ACORN](#), [CRC16_LTE](#), [CRC16_V41MSB](#), [XMODEM](#)

Methods

Equals(CrcStandard)

Determines whether the current [CrcStandard](#) object is equal to another [CrcStandard](#) object.

```
public bool Equals(CrcStandard other)
```

Parameters

other [CrcStandard](#)

The other [CrcStandard](#) object to compare.

Returns

[bool](#)

[true](#) if the two objects are equal; otherwise, [false](#).

Equals(object)

Determines whether the specified object is equal to the current object.

```
public override bool Equals(object obj)
```

Parameters

[obj](#) [object](#)

The object to compare with the current object.

Returns

[bool](#)

[true](#) if the specified object is equal to the current object; otherwise, [false](#).

GetCrcParameters(string)

Return the [CrcStandard](#) with the specified cyclic redundancy check algorithm standard.

```
public static CrcStandard GetCrcParameters(string name)
```

Parameters

[name](#) [string](#)

The name of the CRC Standard to use. See Remarks.

Returns

CrcStandard

The [CrcStandard](#) associated with the specified cyclic redundancy check algorithm standard.

Remarks

The properties for the CRC Standards were taken from <http://reveng.sourceforge.net/crc-catalogue/>.

The following permutationTable shows the valid values for the `name` parameter and the CRC Standard it maps to.

Parameter value-Implements

Exceptions

[ArgumentNullException](#)

`name` is [null](#) (Nothing in Visual Basic).

[ArgumentException](#)

`name` is a an unknown cyclic redundancy check standard.

GetHashCode()

Serves as the default hash function.

```
public override int GetHashCode()
```

Returns

[int](#)

A hash code for the current object.

Class CrcUtility

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

A utility class for handling CRC operations such as generating CRC lookup tables.

```
public static class CrcUtility
```

Inheritance

[object](#) ← CrcUtility

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

BuildLookupTable(int, ulong, bool)

Builds the CRC lookup permutationTable based on the specified parameters.

```
public static ulong[] BuildLookupTable(int size, ulong polynomial, bool reflectIn)
```

Parameters

size [int](#)

The size of the CRC in bits.

polynomial [ulong](#)

The CRC polynomial.

reflectIn [bool](#)

Indicates whether the input bytes should be reflected before processing.

Returns

[ulong](#)[]

An array of [ulong](#) representing the generated CRC lookup permutationTable.

Remarks

This method is used to generate a CRC lookup permutationTable. It performs the actual CRC computation based on the given parameters (size, polynomial, and reflectIn).

Class CryptoUtilities

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

Provides general-purpose utility methods used by cryptographic components and implementations. This includes bit manipulation, secure random byte generation, and helper functions to ensure compliance with cryptographic constraints such as non-zero padding, key generation, or exclusion of reserved byte values.

```
public static class CryptoUtilities
```

Inheritance

[object](#) ← CryptoUtilities

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

This helper class is intended for internal cryptographic infrastructure, test scaffolding, and algorithm development. It supports efficient and secure primitives such as span-based zero-allocation random generation and bit reflection.

All random methods use [Fill\(Span<byte>\)](#) and are optimized for repeatable use in critical paths, including optional inlining for performance.

Methods

DepadBlock(PaddingMode, int, byte[], int, int)

Removes padding from a block and returns a newly allocated array.

```
public static byte[] DepadBlock(PaddingMode padding, int blockSizeBytes, byte[] block, int offset, int count)
```

Parameters

padding [PaddingMode](#)

The padding mode used in the block.

blockSizeBytes [int](#)

The block size in bytes.

block [byte](#)[]

The input padded block.

offset [int](#)

Offset in the input buffer.

count [int](#)

Number of bytes to process.

Returns

[byte](#)[]

A new byte array with padding removed.

DepadBlock(PaddingMode, int, ReadOnlySpan<byte>, Span<byte>)

Removes padding from a block and writes the depadded data into the destination span.

```
public static int DepadBlock(PaddingMode padding, int blockSizeBytes, ReadOnlySpan<byte> source, Span<byte> destination)
```

Parameters

padding [PaddingMode](#)

The padding mode applied to the input data.

blockSizeBytes [int](#)

The block size in bytes for validation.

source [ReadOnlySpan<byte>](#)

The padded input data.

destination [Span<byte>](#)

The destination span to receive depadded data.

Returns

[int](#)

The number of unpadded bytes written to **destination**.

Exceptions

[CryptographicException](#)

Thrown if the padding is invalid, the source is not block-aligned, or the padding mode is unsupported.

PadBlock(PaddingMode, int, byte[], int, int)

Applies padding to a block and returns a newly allocated array.

```
public static byte[] PadBlock(PaddingMode padding, int blockSizeBytes, byte[] block, int offset, int count)
```

Parameters

padding [PaddingMode](#)

The padding mode to apply.

blockSizeBytes [int](#)

The block size in bytes.

block [byte\[\]](#)

The input buffer to pad.

offset [int](#)

The offset within the buffer to start reading.

count [int](#)

The number of bytes to read from the buffer.

Returns

[byte](#)[]

A new padded byte array.

PadBlock(PaddingMode, int, ReadOnlySpan<byte>, Span<byte>)

Applies padding to a block using the specified padding mode and writes to the destination span.

```
public static int PadBlock(PaddingMode padding, int blockSizeBytes, ReadOnlySpan<byte>  
source, Span<byte> destination)
```

Parameters

padding [PaddingMode](#)

Padding mode to apply.

blockSizeBytes [int](#)

The block size in bytes.

source [ReadOnlySpan](#)<[byte](#)>

Input data to pad.

destination [Span](#)<[byte](#)>

Destination span for the padded result.

Returns

[int](#)

Total bytes written to the destination span.

Exceptions

[ArgumentException](#)

Thrown when the destination span is too small.

[CryptographicException](#)

Thrown if the padding mode is invalid or the input is not aligned.

TryDepadBlock(PaddingMode, int, ReadOnlySpan<byte>, Span<byte>, out int)

Attempts to remove padding from the given input buffer using the specified mode.

```
public static bool TryDepadBlock(PaddingMode padding, int blockSizeBytes, ReadOnlySpan<byte> source, Span<byte> destination, out int bytesWritten)
```

Parameters

`padding` [PaddingMode](#)

The padding mode to validate and remove.

`blockSizeBytes` [int](#)

The block size in bytes.

`source` [ReadOnlySpan](#)<[byte](#)>

The padded input buffer.

`destination` [Span](#)<[byte](#)>

The destination span that receives the depadded data.

`bytesWritten` [int](#)

The number of bytes written to `destination`.

Returns

[bool](#)

`true` if depadding was successful; otherwise, `false`.

TryPadBlock(PaddingMode, int, ReadOnlySpan<byte>, Span<byte>, out int)

Attempts to apply padding to the given input buffer using the specified mode.

```
public static bool TryPadBlock(PaddingMode padding, int blockSizeBytes, ReadOnlySpan<byte>  
    source, Span<byte> destination, out int bytesWritten)
```

Parameters

`padding` [PaddingMode](#)

The padding mode to apply.

`blockSizeBytes` [int](#)

The block size in bytes.

`source` [ReadOnlySpan](#)<[byte](#)>

The input buffer to pad.

`destination` [Span](#)<[byte](#)>

The destination span that receives the padded data.

`bytesWritten` [int](#)

The number of bytes written to `destination`.

Returns

[bool](#)

`true` if padding was successfully applied; otherwise, `false`.

Class CubeHash

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

Computes the hash for the input data by using the [CubeHash](#) hash algorithm. This class cannot be inherited.

```
public sealed class CubeHash : HashAlgorithm, ICryptoTransform, IDisposable
```

Inheritance

[object](#) ← [HashAlgorithm](#) ← CubeHash

Implements

[ICryptoTransform](#), [IDisposable](#)

Inherited Members

[HashAlgorithm.Clear\(\)](#), [HashAlgorithm.ComputeHash\(byte\[\]\)](#),
[HashAlgorithm.ComputeHash\(byte\[\], int, int\)](#), [HashAlgorithm.ComputeHash\(Stream\)](#),
[HashAlgorithm.ComputeHashAsync\(Stream, CancellationToken\)](#), [HashAlgorithm.Create\(\)](#),
[HashAlgorithm.Create\(string\)](#), [HashAlgorithm.Dispose\(\)](#),
[HashAlgorithm.TransformBlock\(byte\[\], int, int, byte\[\], int\)](#),
[HashAlgorithm.TransformFinalBlock\(byte\[\], int, int\)](#),
[HashAlgorithm.TryComputeHash\(ReadOnlySpan<byte>, Span<byte>, out int\)](#), [HashAlgorithm.Hash](#),
[object.Equals\(object\)](#), [object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),
[object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\], out bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], byte\[\], CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], string, CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#),

[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, string, Encoding, byte\[\], CancellationToken\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], string\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, string\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#),
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#),
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#).

Remarks

The [CubeHash](#) algorithm is a cryptographic hash function submitted to the NIST SHA-3 hash competition by Daniel J. Bernstein. This implementation supports parameterized configuration to allow flexibility in hash size, round counts, and input block size.

See also: <https://en.wikipedia.org/wiki/CubeHash>

Constructors

CubeHash()

Initializes a new instance of the [CubeHash](#) class with default parameters.

```
public CubeHash()
```

Fields

MaxHashSize

The maximum allowable size of the computed hash, in bits.

```
public const int MaxHashSize = 512
```

Field Value

[int↗](#)

MaxInputBlockSize

The maximum allowable size of the input block, in bytes.

```
public const int MaxInputBlockSize = 128
```

Field Value

[int↗](#)

MaxRounds

The maximum number of rounds permitted for initialization, processing, or finalization.

```
public const int MaxRounds = 4096
```

Field Value

[int↗](#)

MinHashSize

The minimum allowable size of the computed hash, in bits.

```
public const int MinHashSize = 8
```

Field Value

[int↗](#)

MinInputBlockSize

The minimum allowable size of the input block, in bytes.

```
public const int MinInputBlockSize = 1
```

Field Value

[int↗](#)

MinRounds

The minimum number of rounds permitted for initialization, processing, or finalization.

```
public const int MinRounds = 1
```

Field Value

[int↗](#)

Properties

CanReuseTransform

Gets a value indicating whether the current transform can be reused.

```
public override bool CanReuseTransform { get; }
```

Property Value

[bool↗](#)

Always [true↗](#).

CanTransformMultipleBlocks

When overridden in a derived class, gets a value indicating whether multiple blocks can be transformed.

```
public override bool CanTransformMultipleBlocks { get; }
```

Property Value

[bool](#)

[true](#) if multiple blocks can be transformed; otherwise, [false](#).

FinalizationRounds

Gets or sets the number of finalization rounds applied after all input has been processed.

```
public int FinalizationRounds { get; set; }
```

Property Value

[int](#)

Remarks

Finalization rounds provide additional mixing of the internal state to ensure that the final hash output is highly sensitive to every bit of input data. Increasing this value strengthens final-state diffusion.

Exceptions

[ObjectDisposedException](#)

Instance has been disposed and its members are accessed.

[CryptographicUnexpectedOperationException](#)

The hash computation has already started.

[ArgumentOutOfRangeException](#)

Value is less than [MinRounds](#) or greater than [MaxRounds](#).

HashSize

Gets or sets the size, in bits, of the final computed hash output.

```
public int HashSize { get; set; }
```

Property Value

[int](#)

Remarks

The hash size determines the length of the digest returned by the algorithm. Valid values must be between [MinHashSize](#) and [MaxHashSize](#), and divisible by 8. Larger sizes increase output strength.

Exceptions

[ArgumentOutOfRangeException](#)

Value is not within range [MinHashSize](#) to [MaxHashSize](#), or is not a multiple of 8.

[ObjectDisposedException](#)

Instance has been disposed and its members are accessed.

[CryptographicUnexpectedOperationException](#)

The hash computation has already started.

InitializationRounds

Gets or sets the number of initialization rounds to run before processing input data.

```
public int InitializationRounds { get; set; }
```

Property Value

[int](#)

Remarks

Initialization rounds mix the initial state of the algorithm before the first input byte is processed. Increasing this value enhances initial diffusion but increases computation time.

Exceptions

[ArgumentOutOfRangeException](#)

Value is less than [MinRounds](#) or greater than [MaxRounds](#).

[ObjectDisposedException](#)

Instance has been disposed and its members are accessed.

[CryptographicUnexpectedOperationException](#)

The hash computation has already started.

InputBlockSize

Gets the input block size, in bytes, used by consumers of the [CubeHash](#) algorithm, such as [CryptoStream](#).

```
public override int InputBlockSize { get; }
```

Property Value

[int](#)

Remarks

This value reflects the configured [TransformBlockSize](#), which determines how many bytes are accumulated before a transformation round is triggered internally. While this value does not impact the correctness of the hash, feeding data in aligned blocks may improve performance in stream-based scenarios.

Name

Gets the descriptive name of the current configuration.

```
public string Name { get; }
```

Property Value

[string](#)

Remarks

This value includes initialization, round, and hash size parameters.

OutputBlockSize

When overridden in a derived class, gets the output block size.

```
public override int OutputBlockSize { get; }
```

Property Value

[int](#)

The output block size.

Remarks

This is equal to the configured [HashSize](#) divided by 8. For example, a 512-bit hash will produce an output block of 64 bytes. This value corresponds to the full digest returned after [HashFinal\(\)](#) is called.

Rounds

Gets or sets the number of transformation rounds applied to each full input block.

```
public int Rounds { get; set; }
```

Property Value

[int](#)

Remarks

A higher number of rounds provides greater mixing of the state per block, which improves security at the cost of speed.

Exceptions

[ArgumentOutOfRangeException](#)

Value is less than [MinRounds](#) or greater than [MaxRounds](#).

[ObjectDisposedException](#)

Instance has been disposed and its members are accessed.

[CryptographicUnexpectedOperationException](#)

The hash computation has already started.

TransformBlockSize

Gets or sets the size, in bytes, of the input block used by the CubeHash algorithm to determine when to perform a state transformation.

```
public int TransformBlockSize { get; set; }
```

Property Value

[int](#)

Remarks

Unlike [InputBlockSize](#), which is advisory, this property directly affects the output of the hash function. When the number of accumulated input bytes reaches [TransformBlockSize](#), a transformation round is triggered. Modifying this value changes the frequency of internal state updates, impacting both performance and security characteristics.

Exceptions

[ArgumentOutOfRangeException](#)

Value is not within range [MinInputBlockSize](#) to [MaxInputBlockSize](#).

[ObjectDisposedException](#)

Instance has been disposed and its members are accessed.

[CryptographicUnexpectedOperationException](#)

The hash computation has already started.

Methods

Dispose(bool)

Releases the unmanaged resources used by the [HashAlgorithm](#) and optionally releases the managed resources.

```
protected override void Dispose(bool disposing)
```

Parameters

disposing [bool](#)

[true](#) to release both managed and unmanaged resources; [false](#) to release only unmanaged resources.

HashCore(byte[], int, int)

When overridden in a derived class, routes data written to the object into the hash algorithm for computing the hash.

```
protected override void HashCore(byte[] array, int ibStart, int cbSize)
```

Parameters

array [byte](#)[]

The input to compute the hash code for.

ibStart [int](#)

The offset into the byte array from which to begin using data.

cbSize [int](#)

The number of bytes in the byte array to use as data.

HashFinal()

When overridden in a derived class, finalizes the hash computation after the last data is processed by the cryptographic hash algorithm.

```
protected override byte[] HashFinal()
```

Returns

[byte](#)[]

The computed hash code.

Initialize()

Resets the hash algorithm to its initial state.

```
public override void Initialize()
```

Class Elf64

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

Computes a 64-bit non-cryptographic hash using the ELF (Executable and Linkable Format) hashing algorithm.

```
public sealed class Elf64 : HashAlgorithm, ICryptoTransform, IDisposable
```

Inheritance

[object](#) ← [HashAlgorithm](#) ← Elf64

Implements

[ICryptoTransform](#), [IDisposable](#)

Inherited Members

[HashAlgorithm.Clear\(\)](#), [HashAlgorithm.ComputeHash\(byte\[\]\)](#),
[HashAlgorithm.ComputeHash\(byte\[\], int, int\)](#), [HashAlgorithm.ComputeHash\(Stream\)](#),
[HashAlgorithm.ComputeHashAsync\(Stream, CancellationToken\)](#), [HashAlgorithm.Create\(\)](#),
[HashAlgorithm.Create\(string\)](#), [HashAlgorithm.Dispose\(\)](#),
[HashAlgorithm.TransformBlock\(byte\[\], int, int, byte\[\], int\)](#),
[HashAlgorithm.TransformFinalBlock\(byte\[\], int, int\)](#),
[HashAlgorithm.TryComputeHash\(ReadOnlySpan<byte>, Span<byte>, out int\)](#), [HashAlgorithm.Hash](#),
[HashAlgorithm.HashSize](#), [HashAlgorithm.InputBlockSize](#), [HashAlgorithm.OutputBlockSize](#),
[object.Equals\(object, object\)](#), [object.GetType\(\)](#), [object.ReferenceEquals\(object, object\)](#),
[object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\], out bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], byte\[\], CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], string, CancellationToken\)](#),

[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, string, Encoding, byte\[\], CancellationToken\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], string\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, string\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#),
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#),
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#)

Remarks

ELF hashing is a simple, non-cryptographic routine originally used in the UNIX System V ELF object file format. It processes each byte of input by shifting and mixing bits to produce a pseudo-random but repeatable hash output.

This implementation uses a 64-bit internal state and is intended for fast hashing of byte sequences such as identifiers or text keys. It is **not suitable** for cryptographic purposes.

An optional [Seed](#) value may be specified to alter the initial state. The seed cannot be changed once hashing begins.

IMPORTANT

This algorithm is **not** cryptographically secure and should **not** be used for digital signatures, password hashing, or integrity verification in security-sensitive contexts.

Constructors

Elf64()

Initializes a new instance of the [HashAlgorithm](#) class.

```
public Elf64()
```

Properties

CanReuseTransform

Gets a value indicating whether the current transform can be reused.

```
public override bool CanReuseTransform { get; }
```

Property Value

[bool](#)

Always [true](#).

CanTransformMultipleBlocks

When overridden in a derived class, gets a value indicating whether multiple blocks can be transformed.

```
public override bool CanTransformMultipleBlocks { get; }
```

Property Value

[bool](#)

[true](#) if multiple blocks can be transformed; otherwise, [false](#).

Seed

Gets or sets the seed used to initialize the internal hash state.

```
public ulong Seed { get; set; }
```

Property Value

[ulong](#)

The seed value applied before hashing begins.

Remarks

Changing the seed influences the initial hash state and therefore the resulting hash output. Common seed values such as 31, 131, or 1313 are often used to reduce clustering or bias.

Exceptions

[ObjectDisposedException](#)

Instance has been disposed and its members are accessed.

[CryptographicUnexpectedOperationException](#)

The hash computation has already started.

Methods

Dispose(bool)

Releases the unmanaged resources used by the [HashAlgorithm](#) and optionally releases the managed resources.

```
protected override void Dispose(bool disposing)
```

Parameters

[disposing](#) [bool](#)

[true](#) to release both managed and unmanaged resources; [false](#) to release only unmanaged resources.

Equals(object)

Determines whether the specified object is equal to the current object.

```
public override bool Equals(object obj)
```

Parameters

obj [object](#)

The object to compare with the current object.

Returns

[bool](#)

[true](#) if the specified object is equal to the current object; otherwise, [false](#).

GetHashCode()

Serves as the default hash function.

```
public override int GetHashCode()
```

Returns

[int](#)

A hash code for the current object.

HashCore(byte[], int, int)

When overridden in a derived class, routes data written to the object into the hash algorithm for computing the hash.

```
protected override void HashCore(byte[] array, int ibStart, int cbSize)
```

Parameters

array [byte](#)[]

The input to compute the hash code for.

ibStart [int](#)

The offset into the byte array from which to begin using data.

`cbSize` [int](#)

The number of bytes in the byte array to use as data.

HashFinal()

When overridden in a derived class, finalizes the hash computation after the last data is processed by the cryptographic hash algorithm.

```
protected override byte[] HashFinal()
```

Returns

[byte](#)[]

The computed hash code.

Initialize()

Resets the hash algorithm to its initial state.

```
public override void Initialize()
```

Class Fletcher

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

Base class for computing hash using the Fletcher hash algorithm family (Fletcher-16, Fletcher-32, Fletcher-64). This class cannot be inherited.

```
public abstract class Fletcher : HashAlgorithm, ICryptoTransform, IDisposable
```

Inheritance

[object](#) ← [HashAlgorithm](#) ← Fletcher

Implements

[ICryptoTransform](#), [IDisposable](#)

Derived

[Fletcher16](#), [Fletcher32](#), [Fletcher64](#)

Inherited Members

[HashAlgorithm.HashSizeValue](#), [HashAlgorithm.HashValue](#), [HashAlgorithm.State](#),
[HashAlgorithm.Clear\(\)](#), [HashAlgorithm.ComputeHash\(byte\[\]\)](#),
[HashAlgorithm.ComputeHash\(byte\[\], int, int\)](#), [HashAlgorithm.ComputeHash\(Stream\)](#),
[HashAlgorithm.ComputeHashAsync\(Stream, CancellationToken\)](#), [HashAlgorithm.Create\(\)](#),
[HashAlgorithm.Create\(string\)](#), [HashAlgorithm.Dispose\(\)](#),
[HashAlgorithm.HashCore\(ReadOnlySpan<byte>\)](#),
[HashAlgorithm.TransformBlock\(byte\[\], int, int, byte\[\], int\)](#),
[HashAlgorithm.TransformFinalBlock\(byte\[\], int, int\)](#),
[HashAlgorithm.TryComputeHash\(ReadOnlySpan<byte>, Span<byte>, out int\)](#),
[HashAlgorithm.TryHashFinal\(Span<byte>, out int\)](#), [HashAlgorithm.CanReuseTransform](#),
[HashAlgorithm.CanTransformMultipleBlocks](#), [HashAlgorithm.Hash](#), [HashAlgorithm.HashSize](#),
[object.Equals\(object\)](#), [object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),
[object.MemberwiseClone\(\)](#), [object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\], out bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#),

[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, string\)](#) ,
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#) ,
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#) ,
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], string, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, string, Encoding, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], string\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, string\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#).

Remarks

The Fletcher algorithm is used to generate non-cryptographic hash values for a given byte sequence. It operates by computing two components (partA and partB) over the input data and produces a hash based on these.

This implementation handles Fletcher hash sizes of 16, 32, and 64 bits. Derived classes (e.g., Fletcher16, Fletcher32, Fletcher64) can implement specific hash sizes.

IMPORTANT

This algorithm is **not** cryptographically secure and should **not** be used for digital signatures, password hashing, or integrity verification in security-sensitive contexts.

Constructors

Fletcher(int)

Initializes a new instance of the [Fletcher](#) class with the specified hash size.

```
protected Fletcher(int hashSize)
```

Parameters

hashSize [int](#)

The hash size in bits. Valid values are 16, 32, or 64.

Exceptions

[ArgumentException](#)

Thrown if `hashSize` is not 16, 32, or 64.

Properties

InputBlockSize

When overridden in a derived class, gets the input block size.

```
public override int InputBlockSize { get; }
```

Property Value

[int](#)

The input block size.

OutputBlockSize

When overridden in a derived class, gets the output block size.

```
public override int OutputBlockSize { get; }
```

Property Value

[int](#)

The output block size.

Methods

Dispose(bool)

Releases the unmanaged resources used by the [HashAlgorithm](#) and optionally releases the managed resources.

```
protected override void Dispose(bool disposing)
```

Parameters

[disposing](#) [bool](#)

[true](#) to release both managed and unmanaged resources; [false](#) to release only unmanaged resources.

HashCore(byte[], int, int)

Processes a block of data by feeding it into the Fletcher algorithm.

```
protected override void HashCore(byte[] array, int ibStart, int cbSize)
```

Parameters

[array](#) [byte](#)[]

The byte array containing the data to be hashed.

[ibStart](#) [int](#)

The offset at which to start processing in the byte array.

[cbSize](#) [int](#)

The length of the data to process.

HashFinal()

Finalizes the hash computation and returns the resulting hash value.

```
protected override byte[] HashFinal()
```

Returns

[byte](#)[]

A byte array containing the computed hash.

Initialize()

Initializes the internal state of the hash algorithm.

```
public override void Initialize()
```

Class Fletcher16

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

Computes the hash for the input data using the [Fletcher16](#) hash algorithm. This class cannot be inherited.

```
public sealed class Fletcher16 : Fletcher, ICryptoTransform, IDisposable
```

Inheritance

[object](#) ← [HashAlgorithm](#) ← [Fletcher](#) ← [Fletcher16](#)

Implements

[ICryptoTransform](#), [IDisposable](#)

Inherited Members

[Fletcher.InputBlockSize](#), [Fletcher.OutputBlockSize](#), [Fletcher.Initialize\(\)](#), [HashAlgorithm.Clear\(\)](#),
[HashAlgorithm.ComputeHash\(byte\[\]\)](#), [HashAlgorithm.ComputeHash\(byte\[\], int, int\)](#),
[HashAlgorithm.ComputeHash\(Stream\)](#),
[HashAlgorithm.ComputeHashAsync\(Stream, CancellationToken\)](#), [HashAlgorithm.Create\(\)](#),
[HashAlgorithm.Create\(string\)](#), [HashAlgorithm.Dispose\(\)](#),
[HashAlgorithm.TransformBlock\(byte\[\], int, int, byte\[\], int\)](#),
[HashAlgorithm.TransformFinalBlock\(byte\[\], int, int\)](#),
[HashAlgorithm.TryComputeHash\(ReadOnlySpan<byte>, Span<byte>, out int\)](#),
[HashAlgorithm.CanReuseTransform](#), [HashAlgorithm.CanTransformMultipleBlocks](#),
[HashAlgorithm.Hash](#), [HashAlgorithm.HashSize](#), [object.Equals\(object\)](#),
[object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),
[object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\], out bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#),

[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], string, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, string, Encoding, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], string\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, string\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#)

Remarks

[Fletcher16](#) is a non-cryptographic hash algorithm that computes a 16-bit checksum by iterating over the input data. It was invented by Brian Kernighan and Dennis Ritchie, and is typically used for error-checking in applications such as network protocols.

Constructors

Fletcher16()

Initializes a new instance of the [Fletcher16](#) class with a 16-bit hash size.

```
public Fletcher16()
```

Class Fletcher32

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

Computes the hash for the input data using the [Fletcher32](#) hash algorithm. This class cannot be inherited.

```
public sealed class Fletcher32 : Fletcher, ICryptoTransform, IDisposable
```

Inheritance

[object](#) ← [HashAlgorithm](#) ← [Fletcher](#) ← [Fletcher32](#)

Implements

[ICryptoTransform](#), [IDisposable](#)

Inherited Members

[Fletcher.InputBlockSize](#), [Fletcher.OutputBlockSize](#), [Fletcher.Initialize\(\)](#), [HashAlgorithm.Clear\(\)](#),
[HashAlgorithm.ComputeHash\(byte\[\]\)](#), [HashAlgorithm.ComputeHash\(byte\[\], int, int\)](#),
[HashAlgorithm.ComputeHash\(Stream\)](#),
[HashAlgorithm.ComputeHashAsync\(Stream, CancellationToken\)](#), [HashAlgorithm.Create\(\)](#),
[HashAlgorithm.Create\(string\)](#), [HashAlgorithm.Dispose\(\)](#),
[HashAlgorithm.TransformBlock\(byte\[\], int, int, byte\[\], int\)](#),
[HashAlgorithm.TransformFinalBlock\(byte\[\], int, int\)](#),
[HashAlgorithm.TryComputeHash\(ReadOnlySpan<byte>, Span<byte>, out int\)](#),
[HashAlgorithm.CanReuseTransform](#), [HashAlgorithm.CanTransformMultipleBlocks](#),
[HashAlgorithm.Hash](#), [HashAlgorithm.HashSize](#), [object.Equals\(object\)](#),
[object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),
[object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\], out bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#),

[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], string, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, string, Encoding, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], string\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, string\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#)

Remarks

[Fletcher32](#) is a non-cryptographic hash algorithm that computes a 32-bit checksum by iterating over the input data. It was invented by Brian Kernighan and Dennis Ritchie, and is typically used for error-checking in applications such as network protocols.

Constructors

Fletcher32()

Initializes a new instance of the [Fletcher32](#) class with a 32-bit hash size.

```
public Fletcher32()
```

Class Fletcher64

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

Computes the hash for the input data using the [Fletcher64](#) hash algorithm. This class cannot be inherited.

```
public sealed class Fletcher64 : Fletcher, ICryptoTransform, IDisposable
```

Inheritance

[object](#) ← [HashAlgorithm](#) ← [Fletcher](#) ← [Fletcher64](#)

Implements

[ICryptoTransform](#), [IDisposable](#)

Inherited Members

[Fletcher.InputBlockSize](#), [Fletcher.OutputBlockSize](#), [Fletcher.Initialize\(\)](#), [HashAlgorithm.Clear\(\)](#),
[HashAlgorithm.ComputeHash\(byte\[\]\)](#), [HashAlgorithm.ComputeHash\(byte\[\], int, int\)](#),
[HashAlgorithm.ComputeHash\(Stream\)](#),
[HashAlgorithm.ComputeHashAsync\(Stream, CancellationToken\)](#), [HashAlgorithm.Create\(\)](#),
[HashAlgorithm.Create\(string\)](#), [HashAlgorithm.Dispose\(\)](#),
[HashAlgorithm.TransformBlock\(byte\[\], int, int, byte\[\], int\)](#),
[HashAlgorithm.TransformFinalBlock\(byte\[\], int, int\)](#),
[HashAlgorithm.TryComputeHash\(ReadOnlySpan<byte>, Span<byte>, out int\)](#),
[HashAlgorithm.CanReuseTransform](#), [HashAlgorithm.CanTransformMultipleBlocks](#),
[HashAlgorithm.Hash](#), [HashAlgorithm.HashSize](#), [object.Equals\(object\)](#),
[object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),
[object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\], out bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#),

[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], string, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, string, Encoding, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], string\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, string\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#)

Remarks

[Fletcher64](#) is a non-cryptographic hash algorithm that computes a 64-bit checksum by iterating over the input data. It was invented by Brian Kernighan and Dennis Ritchie, and is typically used for error-checking in applications such as network protocols.

Constructors

Fletcher64()

Initializes a new instance of the [Fletcher64](#) class with a 64-bit hash size.

```
public Fletcher64()
```

Interface IResumableHashAlgorithm

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

Represents a hash algorithm that supports resuming a previously finalized hash state and continuing the hash computation with additional input data.

```
public interface IResumableHashAlgorithm
```

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#)

Remarks

This interface is intended for use with non-cryptographic or stateful hash algorithms (such as CRC, FNV, or Jenkins) that allow the internal state to be reconstructed from a finalized hash output.

Implementations must reverse any finalization steps—such as XOR-out or reflection—before resuming the hash process.

Methods

ComputeHashFrom(byte[], byte[])

Resumes a hash computation from a previously finalized hash value and processes additional input, returning the new finalized hash result as a byte array.

```
byte[] ComputeHashFrom(byte[] previousHash, byte[] newData)
```

Parameters

previousHash [byte\[\]](#)

The previously finalized hash value to resume from.

newData [byte\[\]](#)

The additional input data to include in the resumed hash calculation.

Returns

[byte\[\]](#)

A byte array containing the new finalized hash result.

Exceptions

[ArgumentException](#)

Thrown if the `previousHash` length does not match [HashSize](#).

ComputeHashFrom(byte[], byte[], int, int)

Resumes a hash computation from a previously finalized hash value and processes a specified range of new data, returning the new finalized hash result as a byte array.

`byte[] ComputeHashFrom(byte[] previousHash, byte[] newData, int offset, int length)`

Parameters

`previousHash` [byte\[\]](#)

The previously finalized hash value to resume from.

`newData` [byte\[\]](#)

The buffer containing additional input data.

`offset` [int](#)

The zero-based offset into `newData` at which to begin reading data.

`length` [int](#)

The number of bytes to read from `newData`.

Returns

[byte\[\]](#)

A byte array containing the new finalized hash result.

Exceptions

[ArgumentException](#)

Thrown if the `previousHash` length does not match [HashSize](#), or if the offset and length exceed the bounds of `newData`.

ComputeHashFrom(ReadOnlySpan<byte>, ReadOnlySpan<byte>)

Resumes a hash computation from a previously finalized hash value and processes additional input, returning the new finalized hash result as a byte array.

```
byte[] ComputeHashFrom(ReadOnlySpan<byte> previousHash, ReadOnlySpan<byte> newData)
```

Parameters

`previousHash` [ReadOnlySpan](#)<`byte`>

The previously finalized hash value to resume from.

`newData` [ReadOnlySpan](#)<`byte`>

The additional input data to include in the resumed hash calculation.

Returns

`byte`[]

A byte array containing the new finalized hash result.

Exceptions

[ArgumentException](#)

Thrown if the `previousHash` length does not match [HashSize](#).

TryComputeHashFrom(ReadOnlySpan<byte>, ReadOnlySpan<byte>, Span<byte>, out int)

Resumes a hash computation from a previously finalized hash value, processes additional input, and writes the new finalized hash to the specified destination span.

```
bool TryComputeHashFrom(ReadOnlySpan<byte> previousHash, ReadOnlySpan<byte> newData,  
Span<byte> destination, out int bytesWritten)
```

Parameters

previousHash [ReadOnlySpan<byte>](#)

The previously finalized hash value to resume from.

newData [ReadOnlySpan<byte>](#)

The additional input data to include in the resumed hash calculation.

destination [Span<byte>](#)

The destination buffer to write the finalized hash value to.

bytesWritten [int](#)

Outputs the number of bytes written to the destination buffer.

Returns

[bool](#)

[true](#) if the resumed and finalized hash was written successfully; otherwise, [false](#) if the destination span was too small.

Exceptions

[ArgumentException](#)

Thrown if the **previousHash** length does not match [HashSize](#).

Class JSHash

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

Computes a non-cryptographic 32-bit hash using the [JSHash](#) algorithm by Justin Sobel.

```
public sealed class JSHash : HashAlgorithm, ICryptoTransform, IDisposable
```

Inheritance

[object](#) ← [HashAlgorithm](#) ← JSHash

Implements

[ICryptoTransform](#), [IDisposable](#)

Inherited Members

[HashAlgorithm.Clear\(\)](#), [HashAlgorithm.ComputeHash\(byte\[\]\)](#),
[HashAlgorithm.ComputeHash\(byte\[\], int, int\)](#), [HashAlgorithm.ComputeHash\(Stream\)](#),
[HashAlgorithm.ComputeHashAsync\(Stream, CancellationToken\)](#), [HashAlgorithm.Create\(\)](#),
[HashAlgorithm.Create\(string\)](#), [HashAlgorithm.Dispose\(\)](#),
[HashAlgorithm.TransformBlock\(byte\[\], int, int, byte\[\], int\)](#),
[HashAlgorithm.TransformFinalBlock\(byte\[\], int, int\)](#),
[HashAlgorithm.TryComputeHash\(ReadOnlySpan<byte>, Span<byte>, out int\)](#),
[HashAlgorithm.CanReuseTransform](#), [HashAlgorithm.CanTransformMultipleBlocks](#),
[HashAlgorithm.Hash](#), [HashAlgorithm.HashSize](#), [HashAlgorithm.InputBlockSize](#),
[HashAlgorithm.OutputBlockSize](#), [object.Equals\(object\)](#), [object.Equals\(object, object\)](#),
[object.GetHashCode\(\)](#), [object.GetType\(\)](#), [object.ReferenceEquals\(object, object\)](#),
[object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\], out bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], byte\[\], CancellationToken\)](#),

[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], string, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, string, Encoding, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], string\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, string\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#)

Remarks

JSHash is a compact and efficient bitwise hash function, originally developed by Justin Sobel. It is suitable for hash permutationTable indexing and other non-cryptographic use cases where speed is preferred over security.

The core hash function operates using the expression: `hash ^= (hash << 5) + (hash >> 2) + byte`.

IMPORTANT

This algorithm is **not** cryptographically secure and should **not** be used for digital signatures, password hashing, or integrity verification in security-sensitive contexts.

Constructors

`JSHash()`

Initializes a new instance of the [JSHash](#) class.

```
public JSHash()
```

Remarks

This constructor initializes the hash algorithm to a default 32-bit output with a seed value of `0x4E67C6A7`.

Methods

Dispose(bool)

Releases the unmanaged resources used by the [HashAlgorithm](#) and optionally releases the managed resources.

```
protected override void Dispose(bool disposing)
```

Parameters

`disposing` [bool](#)

`true` to release both managed and unmanaged resources; `false` to release only unmanaged resources.

HashCore(byte[], int, int)

When overridden in a derived class, routes data written to the object into the hash algorithm for computing the hash.

```
protected override void HashCore(byte[] array, int ibStart, int cbSize)
```

Parameters

`array` [byte](#)[]

The input to compute the hash code for.

`ibStart` [int](#)

The offset into the byte array from which to begin using data.

`cbSize` [int](#)

The number of bytes in the byte array to use as data.

HashFinal()

When overridden in a derived class, finalizes the hash computation after the last data is processed by the cryptographic hash algorithm.

```
protected override byte[] HashFinal()
```

Returns

[byte](#)[]

The computed hash code.

Initialize()

Resets the hash algorithm to its initial state.

```
public override void Initialize()
```

Class Pearson

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

Computes a hash value using the Pearson hashing algorithm—a fast, lightweight, and non-cryptographic hash function suitable for basic checksums and hash-based lookups.

```
public sealed class Pearson : HashAlgorithm, ICryptoTransform, IDisposable
```

Inheritance

[object](#) ← [HashAlgorithm](#) ← Pearson

Implements

[ICryptoTransform](#), [IDisposable](#)

Inherited Members

[HashAlgorithm.Clear\(\)](#), [HashAlgorithm.ComputeHash\(byte\[\]\)](#),
[HashAlgorithm.ComputeHash\(byte\[\], int, int\)](#), [HashAlgorithm.ComputeHash\(Stream\)](#),
[HashAlgorithm.ComputeHashAsync\(Stream, CancellationToken\)](#), [HashAlgorithm.Create\(\)](#),
[HashAlgorithm.Create\(string\)](#), [HashAlgorithm.Dispose\(\)](#),
[HashAlgorithm.TransformBlock\(byte\[\], int, int, byte\[\], int\)](#),
[HashAlgorithm.TransformFinalBlock\(byte\[\], int, int\)](#),
[HashAlgorithm.TryComputeHash\(ReadOnlySpan<byte>, Span<byte>, out int\)](#), [HashAlgorithm.Hash](#),
[HashAlgorithm.InputBlockSize](#), [HashAlgorithm.OutputBlockSize](#), [object.Equals\(object\)](#),
[object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),
[object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\], out bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], byte\[\], CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], string, CancellationToken\)](#),

[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, string, Encoding, byte\[\], CancellationToken\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], string\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, string\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#),
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#),
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#).

Remarks

The [Pearson hashing algorithm](#), introduced by Peter K. Pearson in 1990, computes a fixed-size hash (typically 8-bit or 16-bit) by transforming each byte of the input using a 256-element permutation table.

The algorithm operates as follows:

1. Initialize the hash result to 0 (or a seed value).
2. For each byte in the input, use the current hash value and the input byte as an index into the permutation table.
3. The output from the table becomes the new hash value.

When computing multi-byte hashes (e.g., 64-bit), the algorithm is repeated for each byte of the result, often using different initialization or byte offsets to reduce collisions.

IMPORTANT

This algorithm is **not** cryptographically secure. It must **not** be used for digital signatures, password hashing, or data integrity checks in security-critical applications.

Constructors

Pearson()

Initializes a new instance of the [Pearson](#) class with a default 8-bit hash size.

```
public Pearson()
```

Fields

MaxHashSize

The maximum allowable hash size in bits.

```
public const int MaxHashSize = 2048
```

Field Value

[int↗](#)

MinHashSize

The minimum allowable hash size in bits.

```
public const int MinHashSize = 8
```

Field Value

[int↗](#)

Properties

CanReuseTransform

Gets a value indicating whether the current transform can be reused.

```
public override bool CanReuseTransform { get; }
```

Property Value

[bool](#)

Always [true](#).

CanTransformMultipleBlocks

When overridden in a derived class, gets a value indicating whether multiple blocks can be transformed.

```
public override bool CanTransformMultipleBlocks { get; }
```

Property Value

[bool](#)

[true](#) if multiple blocks can be transformed; otherwise, [false](#).

HashSize

Gets or sets the size, in bits, of the hash.

```
public int HashSize { get; set; }
```

Property Value

[int](#)

Exceptions

[CryptographicException](#)

Thrown if modified after hashing has started.

[ArgumentOutOfRangeException](#)

Thrown if size is outside valid bounds.

[ArgumentException](#)

Thrown if the hash size is not divisible by 8.

Table

Gets or sets the 256-byte permutation table used by the Pearson algorithm.

```
public byte[] Table { get; set; }
```

Property Value

[byte](#)[]

Exceptions

[CryptographicException](#)

Thrown if modified after hashing has begun.

[ArgumentException](#)

Thrown if the table is not 256 bytes or contains duplicate values.

Methods

Dispose(bool)

Releases the unmanaged resources used by the [HashAlgorithm](#) and optionally releases the managed resources.

```
protected override void Dispose(bool disposing)
```

Parameters

disposing [bool](#)

[true](#) to release both managed and unmanaged resources; [false](#) to release only unmanaged resources.

HashCore(byte[], int, int)

When overridden in a derived class, routes data written to the object into the hash algorithm for computing the hash.

```
protected override void HashCore(byte[] array, int ibStart, int cbSize)
```

Parameters

array `byte[]`

The input to compute the hash code for.

ibStart `int`

The offset into the byte array from which to begin using data.

cbSize `int`

The number of bytes in the byte array to use as data.

HashFinal()

When overridden in a derived class, finalizes the hash computation after the last data is processed by the cryptographic hash algorithm.

```
protected override byte[] HashFinal()
```

Returns

`byte[]`

The computed hash code.

Initialize()

Resets the hash algorithm to its initial state.

```
public override void Initialize()
```

Class SipHash

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

Provides the base implementation of the [SipHash](#) cryptographic hash algorithm—a fast, secure, and keyed pseudorandom function optimized for short input messages. See the [official SipHash specification](#) ↗ for details.

```
public abstract class SipHash : KeyedHashAlgorithm, ICryptoTransform, IDisposable
```

Inheritance

[object](#) ↗ ← [HashAlgorithm](#) ↗ ← [KeyedHashAlgorithm](#) ↗ ← SipHash

Implements

[ICryptoTransform](#) ↗ , [IDisposable](#) ↗

Derived

[SipHash128](#), [SipHash64](#)

Inherited Members

[KeyedHashAlgorithm.KeyValue](#) ↗ , [KeyedHashAlgorithm.Create\(\)](#) ↗ ,
[KeyedHashAlgorithm.Create\(string\)](#) ↗ , [HashAlgorithm.HashSizeValue](#) ↗ , [HashAlgorithm.HashValue](#) ↗ ,
[HashAlgorithm.State](#) ↗ , [HashAlgorithm.Clear\(\)](#) ↗ , [HashAlgorithm.ComputeHash\(byte\[\]\)](#) ↗ ,
[HashAlgorithm.ComputeHash\(byte\[\], int, int\)](#) ↗ , [HashAlgorithm.ComputeHash\(Stream\)](#) ↗ ,
[HashAlgorithm.ComputeHashAsync\(Stream, CancellationToken\)](#) ↗ , [HashAlgorithm.Dispose\(\)](#) ↗ ,
[HashAlgorithm.HashCore\(ReadOnlySpan<byte>\)](#) ↗ ,
[HashAlgorithm.TransformBlock\(byte\[\], int, int, byte\[\], int\)](#) ↗ ,
[HashAlgorithm.TransformFinalBlock\(byte\[\], int, int\)](#) ↗ ,
[HashAlgorithm.TryComputeHash\(ReadOnlySpan<byte>, Span<byte>, out int\)](#) ↗ ,
[HashAlgorithm.TryHashFinal\(Span<byte>, out int\)](#) ↗ , [HashAlgorithm.CanReuseTransform](#) ↗ ,
[HashAlgorithm.CanTransformMultipleBlocks](#) ↗ , [HashAlgorithm.Hash](#) ↗ , [HashAlgorithm.HashSize](#) ↗ ,
[object.Equals\(object\)](#) ↗ , [object.Equals\(object, object\)](#) ↗ , [object.GetHashCode\(\)](#) ↗ , [object.GetType\(\)](#) ↗ ,
[object.MemberwiseClone\(\)](#) ↗ , [object.ReferenceEquals\(object, object\)](#) ↗ , [object.ToString\(\)](#) ↗

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#) ,
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#) ,
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\], out bool\)](#) ,
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], string\)](#) ,

[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#) ,
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, string\)](#) ,
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#) ,
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#) ,
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], string, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, string, Encoding, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], string\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, string\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#)

Remarks

[SipHash](#) is a keyed hash function that requires a 128-bit (16-byte) secret key. It operates on short messages using a sequence of Add-Rotate-XOR (ARX) mixing steps across four 64-bit state variables (`v0` through `v3`). The algorithm is resistant to hash-flooding attacks and is particularly effective for securing hash tables.

This abstract base class defines the core SipHash logic and exposes configuration options such as the number of compression and finalization rounds. It is extended by:

- [SipHash64](#) — Produces a 64-bit hash output suitable for compact keyed checksums.
- [SipHash128](#) — Produces a 128-bit hash output offering increased collision resistance.

The algorithm proceeds in two primary phases:

1. **Compression:** Each 64-bit block of the input is mixed into the internal state using a configurable number of rounds ([CompressionRounds](#)).

2. **Finalization:** After processing all input, additional rounds ([FinalizationRounds](#)) are applied to produce the final hash output.

IMPORTANT

This algorithm is **not** suitable for cryptographic applications such as password hashing, digital signatures, or secure data integrity checks.

Constructors

SipHash(int)

Initializes a new instance of the [SipHash](#) class with a specified hash size.

```
protected SipHash(int hashSize)
```

Parameters

`hashSize` [int](#)

The desired size of the final hash in bits. Supported values are 64 or 128.

Exceptions

[ArgumentException](#)

Thrown if `hashSize` is not supported.

Fields

KeySize

The fixed key size in bytes (128 bits).

```
public const int KeySize = 16
```

Field Value

[int↗](#)

MinCompressionRounds

The minimum number of compression rounds required by SipHash.

```
public const int MinCompressionRounds = 2
```

Field Value

[int↗](#)

MinFinalizationRounds

The minimum number of finalization rounds required by SipHash.

```
public const int MinFinalizationRounds = 4
```

Field Value

[int↗](#)

Properties

CompressionRounds

Gets or sets the number of compression rounds performed per message block.

```
public int CompressionRounds { get; set; }
```

Property Value

[int↗](#)

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if the value is less than [MinCompressionRounds](#).

[ObjectDisposedException](#)

Instance has been disposed and its members are accessed.

[CryptographicUnexpectedOperationException](#)

The hash computation has already started.

FinalizationRounds

Gets or sets the number of finalization rounds performed after all input has been processed.

```
public int FinalizationRounds { get; set; }
```

Property Value

[int](#)

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if the value is less than [MinFinalizationRounds](#).

[ObjectDisposedException](#)

Instance has been disposed and its members are accessed.

[CryptographicUnexpectedOperationException](#)

The hash computation has already started.

InputBlockSize

When overridden in a derived class, gets the input block size.

```
public override int InputBlockSize { get; }
```

Property Value

[int](#)

The input block size.

Key

Gets or sets the key to use in the hash algorithm.

```
public override byte[] Key { get; set; }
```

Property Value

[byte](#)[]

The key to use in the hash algorithm.

Exceptions

[CryptographicException](#)

An attempt was made to change the [Key](#) property after hashing has begun.

OutputBlockSize

When overridden in a derived class, gets the output block size.

```
public override int OutputBlockSize { get; }
```

Property Value

[int](#)

The output block size.

Methods

Dispose(bool)

Releases the unmanaged resources used by the [KeyedHashAlgorithm](#) and optionally releases the managed resources.

```
protected override void Dispose(bool disposing)
```

Parameters

disposing [bool](#)

[true](#) to release both managed and unmanaged resources; [false](#) to release only unmanaged resources.

HashCore(byte[], int, int)

When overridden in a derived class, routes data written to the object into the hash algorithm for computing the hash.

```
protected override void HashCore(byte[] buffer, int offset, int length)
```

Parameters

buffer [byte](#)[]

offset [int](#)

length [int](#)

HashFinal()

When overridden in a derived class, finalizes the hash computation after the last data is processed by the cryptographic hash algorithm.

```
protected override byte[] HashFinal()
```

Returns

[byte](#)[]

The computed hash code.

Initialize()

Resets the hash algorithm to its initial state.

```
public override void Initialize()
```

Class SipHash128

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

```
public sealed class SipHash128 : SipHash, ICryptoTransform, IDisposable
```

Inheritance

[object](#) ← [HashAlgorithm](#) ← [KeyedHashAlgorithm](#) ← [SipHash](#) ← SipHash128

Implements

[ICryptoTransform](#), [IDisposable](#)

Inherited Members

[SipHash.KeySize](#), [SipHash.MinCompressionRounds](#), [SipHash.MinFinalizationRounds](#),
[SipHash.CompressionRounds](#), [SipHash.FinalizationRounds](#), [SipHash.InputBlockSize](#),
[SipHash.OutputBlockSize](#), [SipHash.Key](#), [SipHash.Initialize\(\)](#), [KeyedHashAlgorithm.Create\(\)](#),
[KeyedHashAlgorithm.Create\(string\)](#), [HashAlgorithm.Clear\(\)](#), [HashAlgorithm.ComputeHash\(byte\[\]\)](#),
[HashAlgorithm.ComputeHash\(byte\[\], int, int\)](#), [HashAlgorithm.ComputeHash\(Stream\)](#),
[HashAlgorithm.ComputeHashAsync\(Stream, CancellationToken\)](#), [HashAlgorithm.Dispose\(\)](#),
[HashAlgorithm.TransformBlock\(byte\[\], int, int, byte\[\], int\)](#),
[HashAlgorithm.TransformFinalBlock\(byte\[\], int, int\)](#),
[HashAlgorithm.TryComputeHash\(ReadOnlySpan<byte>, Span<byte>, out int\)](#),
[HashAlgorithm.CanReuseTransform](#), [HashAlgorithm.CanTransformMultipleBlocks](#),
[HashAlgorithm.Hash](#), [HashAlgorithm.HashSize](#), [object.Equals\(object\)](#),
[object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),
[object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\], out bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], byte\[\], CancellationToken\)](#),

[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], string, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#) ,
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, string, Encoding, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], string\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, string\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#) ,
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#) ,
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#)

Constructors

SipHash128()

Initializes a new instance of the [SipHash128](#) class.

```
public SipHash128()
```

Remarks

This method initializes the protected fields of the [SipHash128](#) class to the default values listed in the following permutationTable.

Property-Default Value

Class SipHash64

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

Provides a sealed implementation of the [SipHash](#) cryptographic hash algorithm that produces a 64-bit hash output. This implementation uses a keyed Add-Rotate-XOR (ARX) construction optimized for short messages. See the [official SipHash specification](#) for details.

```
public sealed class SipHash64 : SipHash, ICryptoTransform, IDisposable
```

Inheritance

[object](#) ← [HashAlgorithm](#) ← [KeyedHashAlgorithm](#) ← [SipHash](#) ← SipHash64

Implements

[ICryptoTransform](#), [IDisposable](#)

Inherited Members

[SipHash.KeySize](#), [SipHash.MinCompressionRounds](#), [SipHash.MinFinalizationRounds](#),
[SipHash.CompressionRounds](#), [SipHash.FinalizationRounds](#), [SipHash.InputBlockSize](#),
[SipHash.OutputBlockSize](#), [SipHash.Key](#), [SipHash.Initialize\(\)](#), [KeyedHashAlgorithm.Create\(\)](#),
[KeyedHashAlgorithm.Create\(string\)](#), [HashAlgorithm.Clear\(\)](#), [HashAlgorithm.ComputeHash\(byte\[\]\)](#),
[HashAlgorithm.ComputeHash\(byte\[\], int, int\)](#), [HashAlgorithm.ComputeHash\(Stream\)](#),
[HashAlgorithm.ComputeHashAsync\(Stream, CancellationToken\)](#), [HashAlgorithm.Dispose\(\)](#),
[HashAlgorithm.TransformBlock\(byte\[\], int, int, byte\[\], int\)](#),
[HashAlgorithm.TransformFinalBlock\(byte\[\], int, int\)](#),
[HashAlgorithm.TryComputeHash\(ReadOnlySpan<byte>, Span<byte>, out int\)](#),
[HashAlgorithm.CanReuseTransform](#), [HashAlgorithm.CanTransformMultipleBlocks](#),
[HashAlgorithm.Hash](#), [HashAlgorithm.HashSize](#), [object.Equals\(object\)](#),
[object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),
[object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], byte\[\], out bool\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, byte\[\], string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, Stream, string\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#),

[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#),
[HashAlgorithmExtensions.TryVerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], byte\[\], CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, byte\[\], string, CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#),
[HashAlgorithmExtensions.TryVerifyHashAsync\(HashAlgorithm, string, Encoding, byte\[\], CancellationToken\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, byte\[\], string\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, Stream, string\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlyMemory<byte>, byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>\)](#),
[HashAlgorithmExtensions.VerifyHash\(HashAlgorithm, string, Encoding, byte\[\]\)](#),
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, byte\[\], CancellationToken\)](#),
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken\)](#),
[HashAlgorithmExtensions.VerifyHashAsync\(HashAlgorithm, Stream, string, CancellationToken\)](#).

Remarks

[SipHash64](#) computes a 64-bit keyed hash using a configurable number of compression and finalization rounds. The algorithm is parameterized as [SipHash-c-d](#), where [c](#) is the number of compression rounds and [d](#) is the number of finalization rounds (e.g., [SipHash-2-4](#)).

Each round consists of a series of ARX operations—four additions, four XORs, and six bitwise rotations—applied to four 64-bit state variables. Message blocks are integrated into the state across both the compression and finalization phases using the same round structure.

This implementation is intended for applications requiring fast keyed hashing over short messages, such as hash table protection or lightweight message authentication.

IMPORTANT

This algorithm is **not** suitable for cryptographic applications such as password hashing, digital signatures, or secure data integrity checks.

Constructors

SipHash64()

Initializes a new instance of the [SipHash64](#) class.

```
public SipHash64()
```

Remarks

This method initializes the protected fields of the [SipHash64](#) class to the default values listed in the following permutationTable.

Property-Default Value

Enum TransformMode

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

Defines the direction of a cryptographic transformation.

```
public enum TransformMode
```

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#)

Fields

`Decrypt = 1`

Indicates that the transform should decrypt ciphertext into plaintext.

`Encrypt = 0`

Indicates that the transform should encrypt plaintext into ciphertext.

Remarks

This enumeration is used to distinguish between encryption and decryption operations in cryptographic transform implementations.

Class TweakableSymmetricAlgorithm

Namespace: [Bodu.Security.Cryptography](#)

Assembly: Bodu.Security.Cryptography.dll

Represents a symmetric encryption algorithm that supports an additional tweak value in addition to the encryption key and initialization vector (IV).

```
public abstract class TweakableSymmetricAlgorithm : SymmetricAlgorithm, IDisposable
```

Inheritance

[object](#) ← [SymmetricAlgorithm](#) ← TweakableSymmetricAlgorithm

Implements

[IDisposable](#)

Inherited Members

[SymmetricAlgorithm.BlockSizeValue](#) , [SymmetricAlgorithm.FeedbackSizeValue](#) ,
[SymmetricAlgorithm.IVValue](#) , [SymmetricAlgorithm.KeySizeValue](#) , [SymmetricAlgorithm.KeyValue](#) ,
[SymmetricAlgorithm.LegalBlockSizesValue](#) , [SymmetricAlgorithm.LegalKeySizesValue](#) ,
[SymmetricAlgorithm.ModeValue](#) , [SymmetricAlgorithm.PaddingValue](#) , [SymmetricAlgorithm.Clear\(\)](#) ,
[SymmetricAlgorithm.Create\(\)](#) , [SymmetricAlgorithm.Create\(string\)](#) ,
[SymmetricAlgorithm.CreateDecryptor\(byte\[\], byte\[\]\)](#) ,
[SymmetricAlgorithm.CreateEncryptor\(byte\[\], byte\[\]\)](#) ,
[SymmetricAlgorithm.DecryptCbc\(byte\[\], byte\[\], PaddingMode\)](#) ,
[SymmetricAlgorithm.DecryptCbc\(ReadOnlySpan<byte>, ReadOnlySpan<byte>, PaddingMode\)](#) ,
[SymmetricAlgorithm.DecryptCbc\(ReadOnlySpan<byte>, ReadOnlySpan<byte>, Span<byte>, PaddingMode\)](#) ,
[SymmetricAlgorithm.DecryptCfb\(byte\[\], byte\[\], PaddingMode, int\)](#) ,
[SymmetricAlgorithm.DecryptCfb\(ReadOnlySpan<byte>, ReadOnlySpan<byte>, PaddingMode, int\)](#) ,
[SymmetricAlgorithm.DecryptCfb\(ReadOnlySpan<byte>, ReadOnlySpan<byte>, Span<byte>, PaddingMode, int\)](#) ,
[SymmetricAlgorithm.DecryptEcb\(byte\[\], PaddingMode\)](#) ,
[SymmetricAlgorithm.DecryptEcb\(ReadOnlySpan<byte>, PaddingMode\)](#) ,
[SymmetricAlgorithm.DecryptEcb\(ReadOnlySpan<byte>, Span<byte>, PaddingMode\)](#) ,
[SymmetricAlgorithm.Dispose\(\)](#) , [SymmetricAlgorithm.EncryptCbc\(byte\[\], byte\[\], PaddingMode\)](#) ,
[SymmetricAlgorithm.EncryptCbc\(ReadOnlySpan<byte>, ReadOnlySpan<byte>, PaddingMode\)](#) ,
[SymmetricAlgorithm.EncryptCbc\(ReadOnlySpan<byte>, ReadOnlySpan<byte>, Span<byte>, PaddingMode\)](#) ,

[SymmetricAlgorithm.EncryptCfb\(byte\[\], byte\[\], PaddingMode, int\)](#) ,
[SymmetricAlgorithm.EncryptCfb\(ReadOnlySpan<byte>, ReadOnlySpan<byte>, PaddingMode, int\)](#) ,
[SymmetricAlgorithm.EncryptCfb\(ReadOnlySpan<byte>, ReadOnlySpan<byte>, Span<byte>, PaddingMode, int\)](#) ,
[SymmetricAlgorithm.EncryptEcb\(byte\[\], PaddingMode\)](#) ,
[SymmetricAlgorithm.EncryptEcb\(ReadOnlySpan<byte>, PaddingMode\)](#) ,
[SymmetricAlgorithm.EncryptEcb\(ReadOnlySpan<byte>, Span<byte>, PaddingMode\)](#) ,
[SymmetricAlgorithm.GenerateIV\(\)](#) , [SymmetricAlgorithm.GenerateKey\(\)](#) ,
[SymmetricAlgorithm.GetCiphertextLengthCbc\(int, PaddingMode\)](#) ,
[SymmetricAlgorithm.GetCiphertextLengthCfb\(int, PaddingMode, int\)](#) ,
[SymmetricAlgorithm.GetCiphertextLengthEcb\(int, PaddingMode\)](#) ,
[SymmetricAlgorithm.TryDecryptCbc\(ReadOnlySpan<byte>, ReadOnlySpan<byte>, Span<byte>, out int, PaddingMode\)](#) ,
[SymmetricAlgorithm.TryDecryptCbcCore\(ReadOnlySpan<byte>, ReadOnlySpan<byte>, Span<byte>, PaddingMode, out int\)](#) ,
[SymmetricAlgorithm.TryDecryptCfb\(ReadOnlySpan<byte>, ReadOnlySpan<byte>, Span<byte>, out int, PaddingMode, int\)](#) ,
[SymmetricAlgorithm.TryDecryptCfbCore\(ReadOnlySpan<byte>, ReadOnlySpan<byte>, Span<byte>, PaddingMode, int, out int\)](#) ,
[SymmetricAlgorithm.TryDecryptEcb\(ReadOnlySpan<byte>, Span<byte>, PaddingMode, out int\)](#) ,
[SymmetricAlgorithm.TryDecryptEcbCore\(ReadOnlySpan<byte>, Span<byte>, PaddingMode, out int\)](#) ,
[SymmetricAlgorithm.TryEncryptCbc\(ReadOnlySpan<byte>, ReadOnlySpan<byte>, Span<byte>, out int, PaddingMode\)](#) ,
[SymmetricAlgorithm.TryEncryptCbcCore\(ReadOnlySpan<byte>, ReadOnlySpan<byte>, Span<byte>, PaddingMode, out int\)](#) ,
[SymmetricAlgorithm.TryEncryptCfb\(ReadOnlySpan<byte>, ReadOnlySpan<byte>, Span<byte>, out int, PaddingMode, int\)](#) ,
[SymmetricAlgorithm.TryEncryptCfbCore\(ReadOnlySpan<byte>, ReadOnlySpan<byte>, Span<byte>, PaddingMode, int, out int\)](#) ,
[SymmetricAlgorithm.TryEncryptEcb\(ReadOnlySpan<byte>, Span<byte>, PaddingMode, out int\)](#) ,
[SymmetricAlgorithm.TryEncryptEcbCore\(ReadOnlySpan<byte>, Span<byte>, PaddingMode, out int\)](#) ,
[SymmetricAlgorithm.ValidKeySize\(int\)](#) , [SymmetricAlgorithm.BlockSize](#) ,
[SymmetricAlgorithm.FeedbackSize](#) , [SymmetricAlgorithm.IV](#) , [SymmetricAlgorithm.Key](#) ,
[SymmetricAlgorithm.KeySize](#) , [SymmetricAlgorithm.LegalBlockSizes](#) ,
[SymmetricAlgorithm.LegalKeySizes](#) , [SymmetricAlgorithm.Mode](#) , [SymmetricAlgorithm.Padding](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Extension Methods

[NumericExtensions.GetBytes<T>\(T, bool\)](#)

Remarks

This abstract class extends the [SymmetricAlgorithm](#) base class to support tweakable ciphers, such as Threefish, which use an additional tweak input to influence encryption.

The tweak is provided as a byte array and may vary in size depending on the algorithm implementation. Valid tweak sizes are specified via the [LegalTweakSizes](#) property.

Derived classes must override the encryptor and decryptor creation methods to incorporate the tweak into the encryption or decryption process.

Fields

LegalTweakSizesValue

Specifies the tweak sizes, in bits, that are supported by the algorithm.

```
protected KeySizes[] LegalTweakSizesValue
```

Field Value

[KeySizes](#)[]

TweakSizeValue

Stores the currently configured tweak size in bits.

```
protected int TweakSizeValue
```

Field Value

[int](#)

TweakValue

Stores the current tweak value.

```
protected byte[] TweakValue
```

Field Value

[byte](#)[]

Properties

LegalTweakSizes

Gets the tweak sizes, in bits, that are supported by the symmetric algorithm.

```
public virtual KeySizes[] LegalTweakSizes { get; }
```

Property Value

[KeySizes](#)[]

An array of [KeySizes](#) values indicating valid tweak sizes.

Tweak

Gets or sets the tweak value for the algorithm.

```
public virtual byte[] Tweak { get; set; }
```

Property Value

[byte](#)[]

The tweak value as a byte array.

Remarks

The provided value must conform to one of the allowed sizes defined in [LegalTweakSizes](#). A defensive copy is made during assignment and retrieval.

Exceptions

[ArgumentNullException](#)

Thrown if the value is [null](#).

[CryptographicException](#)

Thrown if the specified tweak size is not valid.

TweakSize

Gets or sets the size, in bits, of the tweak used by the algorithm.

```
public virtual int TweakSize { get; set; }
```

Property Value

[int](#)

The configured tweak size, in bits.

Exceptions

[CryptographicException](#)

Thrown if the provided tweak size is not supported.

TweakSpan

Gets a read-only span view of the current tweak value.

```
protected ReadOnlySpan<byte> TweakSpan { get; }
```

Property Value

[ReadOnlySpan](#)<[byte](#)>

Remarks

This property exposes the current tweak as a [ReadOnlySpan<T>](#) for efficient access in span-based cryptographic operations. If no tweak has been set, this span is empty.

Methods

CreateDecryptor()

Creates a symmetric decryptor object with the current [Key](#) property and initialization vector ([IV](#)).

```
public override ICryptoTransform CreateDecryptor()
```

Returns

[ICryptoTransform](#)

A symmetric decryptor object.

CreateDecryptor(byte[], byte[], byte[])

Creates a symmetric decryptor object using the specified key, IV, and tweak.

```
public abstract ICryptoTransform CreateDecryptor(byte[] rgbKey, byte[] rgbIV, byte[] tweak)
```

Parameters

rgbKey [byte](#)[]

The encryption key.

rgbIV [byte](#)[]

The initialization vector.

tweak [byte](#)[]

The tweak value to use.

Returns

[ICryptoTransform](#)

An [ICryptoTransform](#) instance for decryption.

CreateEncryptor()

Creates a symmetric encryptor object with the current [Key](#) property and initialization vector ([IV](#)).

```
public override ICryptoTransform CreateEncryptor()
```

Returns

[ICryptoTransform](#)

A symmetric encryptor object.

CreateEncryptor(byte[], byte[], byte[])

Creates a symmetric encryptor object using the specified key, IV, and tweak.

```
public abstract ICryptoTransform CreateEncryptor(byte[] rgbKey, byte[] rgbIV, byte[] tweak)
```

Parameters

rgbKey [byte](#)[]

The encryption key.

rgbIV [byte](#)[]

The initialization vector.

tweak [byte](#)[]

The tweak value to use.

Returns

[ICryptoTransform](#)

An [ICryptoTransform](#) instance for encryption.

Dispose(bool)

Releases the unmanaged resources used by the [SymmetricAlgorithm](#) and optionally releases the managed resources.

```
protected override void Dispose(bool disposing)
```

Parameters

disposing [bool](#)

[true](#) to release both managed and unmanaged resources; [false](#) to release only unmanaged resources.

GenerateTweak()

Generates a new tweak value for use by the algorithm.

```
public abstract void GenerateTweak()
```

Remarks

This method should populate the [Tweak](#) property with a valid value based on the current [TweakSize](#). Implementations must ensure the generated tweak conforms to [LegalTweakSizes](#).

ThrowIfInvalidTweakSize(byte[], string?)

Throws if the specified **tweak** is [null](#) or its size is not valid.

```
protected void ThrowIfInvalidTweakSize(byte[] tweak, string? paramName = null)
```

Parameters

tweak [byte](#)[]

The tweak value to validate.

paramName [string](#)

The name of the parameter, automatically inferred if not specified.

Exceptions

[ArgumentException](#)

Thrown when `tweak` is [null](#).

[CryptographicException](#)

Thrown when `tweak` is not supported by [LegalTweakSizes](#).

ThrowIfInvalidTweakSize(int, string?)

Throws if the specified bit length is not a valid tweak size for this algorithm.

```
protected void ThrowIfInvalidTweakSize(int bitLength, string? paramName = null)
```

Parameters

`bitLength` [int](#)

The tweak size in bits.

`paramName` [string](#)

The name of the parameter, automatically inferred if not specified.

Exceptions

[CryptographicException](#)

Thrown when `bitLength` is not supported by [LegalTweakSizes](#).

ThrowIfTweakNotSet()

Throws a [CryptographicException](#) if the tweak has not been set or generated.

```
protected void ThrowIfTweakNotSet()
```

Remarks

This method ensures that the [Tweak](#) property is not accessed in an uninitialized state. It is recommended to call [GenerateTweak\(\)](#) or assign a value to [Tweak](#) before use.

Exceptions

[CryptographicException](#)

Thrown when the tweak value is empty, indicating it has not been initialized or generated.

TryCreateDecryptor(ReadOnlySpan<byte>, ReadOnlySpan<byte>, ReadOnlySpan<byte>, out ICryptoTransform)

Attempts to create a symmetric decryptor using span-based key, IV, and tweak values.

```
public virtual bool TryCreateDecryptor(ReadOnlySpan<byte> key, ReadOnlySpan<byte> iv,  
ReadOnlySpan<byte> tweak, out ICryptoTransform transform)
```

Parameters

key [ReadOnlySpan](#)<byte>

The encryption key as a read-only span.

iv [ReadOnlySpan](#)<byte>

The initialization vector as a read-only span.

tweak [ReadOnlySpan](#)<byte>

The tweak value as a read-only span.

transform [ICryptoTransform](#)

The resulting [ICryptoTransform](#) instance if successful.

Returns

[bool](#)

[true](#) if the decryptor was created successfully; otherwise, [false](#).

Remarks

This method provides a span-based alternative to [CreateDecryptor\(byte\[\], byte\[\], byte\[\]\)](#). The default implementation converts the inputs to arrays. Override for custom span handling.

TryCreateEncryptor(ReadOnlySpan<byte>, ReadOnlySpan<byte>, ReadOnlySpan<byte>, out ICryptoTransform)

Attempts to create a symmetric encryptor using span-based key, IV, and tweak values.

```
public virtual bool TryCreateEncryptor(ReadOnlySpan<byte> key, ReadOnlySpan<byte> iv,  
ReadOnlySpan<byte> tweak, out ICryptoTransform transform)
```

Parameters

key [ReadOnlySpan<byte>](#)

The encryption key as a read-only span.

iv [ReadOnlySpan<byte>](#)

The initialization vector as a read-only span.

tweak [ReadOnlySpan<byte>](#)

The tweak value as a read-only span.

transform [ICryptoTransform](#)

The resulting [ICryptoTransform](#) instance if successful.

Returns

[bool](#)

[true](#) if the encryptor was created successfully; otherwise, [false](#).

Remarks

This method provides a span-based alternative to [CreateEncryptor\(byte\[\], byte\[\], byte\[\]\)](#). The default implementation converts the inputs to arrays. Override for custom span handling.

ValidTweakSize(int)

Determines whether the specified tweak size, in bits, is valid for this algorithm.

```
public bool ValidTweakSize(int length)
```

Parameters

length [int](#)

The tweak size in bits.

Returns

[bool](#)

[true](#) if the size is valid; otherwise, [false](#).

Namespace Bodu.Security.Cryptography.Extensions

Classes

[HashAlgorithmExtensions](#)

Provides a set of [`static`](#) (`Shared` in Visual Basic) methods that extend the [HashAlgorithm](#) class.

Class HashAlgorithmExtensions

Namespace: [Bodu.Security.Cryptography.Extensions](#)

Assembly: Bodu.Security.Cryptography.dll

Provides a set of [static](#) ([Shared](#) in Visual Basic) methods that extend the [HashAlgorithm](#) class.

```
public static class HashAlgorithmExtensions
```

Inheritance

[object](#) ← HashAlgorithmExtensions

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

TryVerifyHash(HashAlgorithm, byte[], byte[])

Attempts to compute and verify the hash of a byte array against an expected hash value.

```
public static bool TryVerifyHash(this HashAlgorithm algorithm, byte[] input,  
byte[] expectedHash)
```

Parameters

algorithm [HashAlgorithm](#)

The [HashAlgorithm](#) instance used for hashing.

input [byte](#)[]

The input data as a byte array.

expectedHash [byte](#)[]

The expected hash value.

Returns

[bool](#)

`true` if the hash matches; otherwise, `false`.

Exceptions

[ArgumentNullException](#)

Thrown if `algorithm` or `expectedHash` is `null`.

TryVerifyHash(HashAlgorithm, byte[], byte[], out bool)

Attempts to compute the hash of the specified input and compare it against the expected hash.

```
public static bool TryVerifyHash(this HashAlgorithm algorithm, byte[] input, byte[]  
expectedHash, out bool result)
```

Parameters

`algorithm` [HashAlgorithm](#)

The [HashAlgorithm](#) instance used for computing the hash.

`input` [byte](#)[]

The byte array input to hash.

`expectedHash` [byte](#)[]

The expected hash value for comparison.

`result` [bool](#)

Outputs `true` if the computed hash matches `expectedHash`; otherwise, `false`.

Returns

[bool](#)

`true` if hashing and comparison completed without exception; otherwise, `false`.

Remarks

This method is useful for defensive validation when inputs may be malformed or optional. Unlike [VerifyHash](#), it avoids exceptions during failure.

Exceptions

[ArgumentNullException](#)

Thrown if `algorithm` is `null`.

TryVerifyHash(HashAlgorithm, byte[], string)

Attempts to verify the computed hash of the byte array input against a hexadecimal hash string.

```
public static bool TryVerifyHash(this HashAlgorithm algorithm, byte[] input,  
    string expectedHex)
```

Parameters

`algorithm` [HashAlgorithm](#)

The hashing algorithm.

`input` [byte](#)[]

The data to hash.

`expectedHex` [string](#)

The expected hexadecimal hash string.

Returns

[bool](#)

`true` if the computed hash matches the expected hex string; otherwise, `false`.

Remarks

This is useful for verifying known test vectors stored in hexadecimal format.

Exceptions

[ArgumentNullException](#)

Thrown if `algorithm` or `expectedHash` is `null`.

TryVerifyHash(HashAlgorithm, Stream, byte[])

Attempts to verify that the hash of a stream matches the expected byte array.

```
public static bool TryVerifyHash(this HashAlgorithm algorithm, Stream stream,
byte[] expectedHash)
```

Parameters

`algorithm` [HashAlgorithm](#)

The hashing algorithm used to compute the hash.

`stream` [Stream](#)

The input stream to hash. The stream must be readable and ideally seekable.

`expectedHash` [byte](#)[]

The expected hash value.

Returns

`bool`

`true` if the stream produces a matching hash; otherwise, `false`.

Exceptions

[ArgumentNullException](#)

Thrown if `algorithm` or `expectedHash` is `null`.

TryVerifyHash(HashAlgorithm, Stream, string)

Attempts to verify that the hash of a stream matches the expected hexadecimal hash value.

```
public static bool TryVerifyHash(this HashAlgorithm algorithm, Stream stream,
string expectedHex)
```

Parameters

algorithm [HashAlgorithm](#)

The hashing algorithm used to compute the hash.

stream [Stream](#)

The input stream to hash. The stream must be readable and ideally seekable.

expectedHex [string](#)

The expected hash value in hexadecimal format.

Returns

[bool](#)

true if the stream hash matches the expected hex; otherwise, **false**.

Exceptions

[ArgumentNullException](#)

Thrown if **algorithm** or **expectedHex** is **null**.

TryVerifyHash(HashAlgorithm, ReadOnlyMemory<byte>, byte[])

Attempts to verify that the hash of a memory buffer matches the expected byte array value.

```
public static bool TryVerifyHash(this HashAlgorithm algorithm, ReadOnlyMemory<byte> input,
byte[] expectedHash)
```

Parameters

algorithm [HashAlgorithm](#)

The hashing algorithm instance.

input [ReadOnlyMemory](#)<[byte](#)>

The memory buffer containing input data.

expectedHash [byte](#)[]

The expected hash result.

Returns

[bool](#)

true if the memory contents produce a matching hash; otherwise, **false**.

Exceptions

[ArgumentNullException](#)

Thrown if **algorithm** or **expectedHash** is **null**.

TryVerifyHash(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>)

Attempts to verify that the hash computed from a span of bytes matches the expected span value.

```
public static bool TryVerifyHash(this HashAlgorithm algorithm, ReadOnlySpan<byte> input,  
ReadOnlySpan<byte> expectedHash)
```

Parameters

algorithm [HashAlgorithm](#)

The algorithm used for hashing.

input [ReadOnlySpan](#)<[byte](#)>

The span of input bytes.

expectedHash [ReadOnlySpan](#)<[byte](#)>

The expected hash as a span.

Returns

[bool](#)

`true` if the spans match; otherwise, `false`.

Exceptions

[ArgumentNullException](#)

Thrown if `algorithm` is `null`.

TryVerifyHash(HashAlgorithm, string, Encoding, byte[])

Attempts to verify that the computed hash of a UTF-encoded string matches the expected hash.

```
public static bool TryVerifyHash(this HashAlgorithm algorithm, string input, Encoding encoding, byte[] expectedHash)
```

Parameters

`algorithm` [HashAlgorithm](#)

The hash algorithm instance.

`input` [string](#)

The plain text input string.

`encoding` [Encoding](#)

The encoding used to convert the string into bytes.

`expectedHash` [byte](#)[]

The expected hash byte array.

Returns

[bool](#)

`true` if the hash matches the expected value; otherwise, `false`.

Exceptions

[ArgumentNullException](#)

Thrown if `algorithm`, `input`, `encoding`, or `expectedHash` is `null`.

TryVerifyHashAsync(HashAlgorithm, byte[], byte[], CancellationToken)

Attempts to asynchronously verify that the computed hash of a byte array matches the expected hash.

```
public static Task<bool> TryVerifyHashAsync(this HashAlgorithm algorithm, byte[] input,
    byte[] expectedHash, CancellationToken cancellationToken = default)
```

Parameters

`algorithm` [HashAlgorithm](#)

The [HashAlgorithm](#) used to compute the hash.

`input` [byte](#)[]

The input data as a byte array.

`expectedHash` [byte](#)[]

The expected hash to compare against.

`cancellationToken` [CancellationToken](#)

A token to cancel the asynchronous operation.

Returns

[Task](#)<[bool](#)>

`true` if the computed hash matches the expected value; otherwise, `false`.

Remarks

Converts the input to a stream internally for compatibility with async hash APIs.

Exceptions

[ArgumentNullException](#)

Thrown if any argument is `null`.

TryVerifyHashAsync(HashAlgorithm, byte[], string, CancellationToken)

Attempts to asynchronously verify that the computed hash of a byte array matches the expected hexadecimal hash.

```
public static Task<bool> TryVerifyHashAsync(this HashAlgorithm algorithm, byte[] input,
    string expectedHex, CancellationToken cancellationToken = default)
```

Parameters

`algorithm` [HashAlgorithm](#)

The [HashAlgorithm](#) used for hashing.

`input` [byte](#)[]

The input data to hash.

`expectedHex` [string](#)

The expected hash in hexadecimal format.

`cancellationToken` [CancellationToken](#)

Optional cancellation token.

Returns

[Task](#)<[bool](#)>

`true` if the hash matches the hex string; otherwise, `false`.

Remarks

This method enables secure comparison against external sources or stored hashes in hex format.

Exceptions

[ArgumentNullException](#)

Thrown if any argument is `null`.

`TryVerifyHashAsync(HashAlgorithm, Stream, byte[], CancellationToken)`

Attempts to asynchronously verify that the computed hash of a stream matches the expected hash value.

```
public static Task<bool> TryVerifyHashAsync(this HashAlgorithm algorithm, Stream stream,
byte[] expectedHash, CancellationToken cancellationToken = default)
```

Parameters

`algorithm` [HashAlgorithm](#)

The [HashAlgorithm](#) instance used to compute the hash.

`stream` [Stream](#)

The stream to read and hash. Must be readable.

`expectedHash` [byte](#)[]

The expected hash value as a byte array.

`cancellationToken` [CancellationToken](#)

Token used to cancel the operation.

Returns

[Task](#)<[bool](#)>

`true` if the computed hash matches; otherwise, `false`.

Remarks

This method safely validates a stream against a known hash, handling any internal errors gracefully.

Exceptions

[ArgumentNullException](#)

Thrown if `algorithm` is `null`.

TryVerifyHashAsync(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken)

Attempts to asynchronously verify that the computed hash of a stream matches the expected memory buffer.

```
public static Task<bool> TryVerifyHashAsync(this HashAlgorithm algorithm, Stream stream,  
    ReadOnlyMemory<byte> expectedHash, CancellationToken cancellationToken = default)
```

Parameters

`algorithm` [HashAlgorithm](#)

The [HashAlgorithm](#) used to compute the hash.

`stream` [Stream](#)

The stream to read and hash asynchronously.

`expectedHash` [ReadOnlyMemory](#)<byte>

The expected hash value as a memory buffer.

`cancellationToken` [CancellationToken](#)

Token to cancel the operation.

Returns

[Task](#)<bool>

`true` if the computed hash matches; otherwise, `false`.

Remarks

This overload supports memory-friendly comparison of hashes from stream input.

Exceptions

[ArgumentNullException](#)

Thrown if `algorithm` is `null`.

TryVerifyHashAsync(HashAlgorithm, Stream, string, CancellationToken)

Attempts to asynchronously verify that the computed hash of a stream matches the expected hexadecimal string.

```
public static Task<bool> TryVerifyHashAsync(this HashAlgorithm algorithm, Stream stream,
    string expectedHex, CancellationToken cancellationToken = default)
```

Parameters

`algorithm` [HashAlgorithm](#)

The [HashAlgorithm](#) used to compute the hash.

`stream` [Stream](#)

The readable stream to hash.

`expectedHex` [string](#)

The expected hash as a hexadecimal string.

`cancellationToken` [CancellationToken](#)

Token used to cancel the operation.

Returns

[Task](#) <bool>

`true` if the computed hash matches the expected hex; otherwise, `false`.

Remarks

Used for verifying hashes from test vectors or external sources represented as hex strings.

Exceptions

[ArgumentNullException](#)

Thrown if `algorithm` or `expectedHex` is `null`.

TryVerifyHashAsync(HashAlgorithm, string, Encoding, byte[], CancellationToken)

Attempts to asynchronously verify that the computed hash of a string (after encoding) matches the expected value.

```
public static Task<bool> TryVerifyHashAsync(this HashAlgorithm algorithm, string input, Encoding encoding, byte[] expectedHash, CancellationToken cancellationToken = default)
```

Parameters

`algorithm` [HashAlgorithm](#)

The [HashAlgorithm](#) instance.

`input` [string](#)

The input string to encode and hash.

`encoding` [Encoding](#)

The character encoding used to convert the string to bytes.

`expectedHash` [byte](#)[]

The expected hash byte array.

`cancellationToken` [CancellationToken](#)

A token to cancel the asynchronous operation.

Returns

[Task](#)<[bool](#)>

`true` if the computed hash matches `expectedHash`; otherwise, `false`.

Remarks

Used when comparing user-entered or stored strings after encoding to binary form for hashing.

Exceptions

[ArgumentNullException](#)

Thrown if any argument is `null`.

VerifyHash(HashAlgorithm, byte[], byte[])

Verifies that the computed hash of the input data matches the expected hash value.

```
public static bool VerifyHash(this HashAlgorithm algorithm, byte[] input,  
    byte[] expectedHash)
```

Parameters

`algorithm` [HashAlgorithm](#)

The [HashAlgorithm](#) instance used to compute the hash.

`input` [byte](#)[]

The input byte array whose hash will be computed.

`expectedHash` [byte](#)[]

The expected hash value as a byte array.

Returns

[bool](#)

`true` if the computed hash is equal to `expectedHash`; otherwise, `false`.

Remarks

Computes a hash from the input and compares it to `expectedHash` using byte-wise equality.

Exceptions

[ArgumentNullException](#)

Thrown if `algorithm` or `expectedHash` is `null`.

VerifyHash(HashAlgorithm, byte[], string)

Verifies that the computed hash of the input data matches the expected hash value as a hexadecimal string.

```
public static bool VerifyHash(this HashAlgorithm algorithm, byte[] input,
    string expectedHex)
```

Parameters

`algorithm` [HashAlgorithm](#)

The [HashAlgorithm](#) instance used to compute the hash.

`input` [byte](#)[]

The input byte array whose hash will be computed.

`expectedHex` [string](#)

The expected hash value as a hexadecimal string.

Returns

[bool](#)

`true` if the computed hash matches `expectedHex`; otherwise, `false`.

Remarks

Converts the computed hash to a hexadecimal string and performs a case-insensitive comparison to `expectedHex`.

Exceptions

[ArgumentNullException](#)

Thrown if `algorithm` or `expectedHex` is `null`.

VerifyHash(HashAlgorithm, Stream, byte[])

Verifies that the computed hash of the stream matches the expected hash value.

```
public static bool VerifyHash(this HashAlgorithm algorithm, Stream stream,
byte[] expectedHash)
```

Parameters

`algorithm` [HashAlgorithm](#)

The [HashAlgorithm](#) instance used to compute the hash.

`stream` [Stream](#)

The input stream to read and hash. The stream must be readable.

`expectedHash` [byte](#)[]

The expected hash value as a byte array.

Returns

[bool](#)

`true` if the hash matches `expectedHash`; otherwise, `false`.

Remarks

Computes the hash of the stream and compares it to `expectedHash` byte-by-byte.

Exceptions

[ArgumentNullException](#)

Thrown if `algorithm` or `expectedHash` is `null`.

VerifyHash(HashAlgorithm, Stream, string)

Verifies that the computed hash of the stream matches the expected hash value as a hexadecimal string.

```
public static bool VerifyHash(this HashAlgorithm algorithm, Stream stream,
string expectedHex)
```

Parameters

algorithm [HashAlgorithm](#)

The [HashAlgorithm](#) instance used to compute the hash.

stream [Stream](#)

The stream to hash. Must be readable.

expectedHex [string](#)

The expected hash value as a hexadecimal string.

Returns

[bool](#)

true if the hash matches **expectedHex**; otherwise, **false**.

Remarks

Converts the computed hash of the stream to a hex string and compares it to **expectedHex**.

Exceptions

[ArgumentNullException](#)

Thrown if **algorithm** or **expectedHex** is **null**.

VerifyHash(HashAlgorithm, ReadOnlyMemory<byte>, byte[])

Verifies that the computed hash of the input memory block matches the expected hash.

```
public static bool VerifyHash(this HashAlgorithm algorithm, ReadOnlyMemory<byte> input,
byte[] expectedHash)
```

Parameters

algorithm [HashAlgorithm](#)

The [HashAlgorithm](#) used to compute the hash.

input [ReadOnlyMemory](#)<[byte](#)>

The memory buffer to hash.

expectedHash [byte](#)[]

The expected hash value.

Returns

[bool](#)

true if the hash matches; otherwise, **false**.

Remarks

This overload supports memory-friendly verification against a byte array.

Exceptions

[ArgumentNullException](#)

Thrown if **algorithm** or **expectedHash** is **null**.

VerifyHash(HashAlgorithm, ReadOnlySpan<byte>, ReadOnlySpan<byte>)

Verifies that the computed hash of the input span matches the expected span.

```
public static bool VerifyHash(this HashAlgorithm algorithm, ReadOnlySpan<byte> input,  
    ReadOnlySpan<byte> expectedHash)
```

Parameters

algorithm [HashAlgorithm](#)

The [HashAlgorithm](#) used to compute the hash.

`input` [ReadOnlySpan<byte>](#)

The input span of bytes to hash.

`expectedHash` [ReadOnlySpan<byte>](#)

The expected hash as a byte span.

Returns

[bool](#)

`true` if the computed hash matches the expected hash; otherwise, `false`.

Remarks

Converts the span to a buffer and computes its hash using [ComputeHash\(byte\[\]\)](#).

Exceptions

[ArgumentNullException](#)

Thrown if `algorithm` is `null`.

VerifyHash(HashAlgorithm, string, Encoding, byte[])

Verifies that the hash of the specified string (after encoding) matches the expected value.

```
public static bool VerifyHash(this HashAlgorithm algorithm, string text, Encoding encoding,
byte[] expectedHash)
```

Parameters

`algorithm` [HashAlgorithm](#)

The hashing algorithm.

`text` [string](#)

The input string to encode.

`encoding` [Encoding](#)

The encoding used to convert the string to bytes.

`expectedHash` [byte\[\]](#)

The expected hash as a byte array.

Returns

[bool](#)

`true` if the computed hash matches; otherwise, `false`.

Remarks

This overload allows direct verification of encoded strings against known hash values.

Exceptions

[ArgumentNullException](#)

Thrown if `algorithm`, `text`, `encoding`, or `expectedHash` is `null`.

VerifyHashAsync(HashAlgorithm, Stream, byte[], CancellationToken)

Asynchronously verifies that the computed hash of a stream matches the expected hash value.

```
public static Task<bool> VerifyHashAsync(this HashAlgorithm algorithm, Stream stream, byte[] expectedHash, CancellationToken cancellationToken = default)
```

Parameters

`algorithm` [HashAlgorithm](#)

The [HashAlgorithm](#) instance used to compute the hash.

`stream` [Stream](#)

The stream to read and hash asynchronously. The stream must be readable.

`expectedHash` [byte\[\]](#)

The expected hash value as a byte array.

cancellationToken [CancellationToken](#)

A token to cancel the asynchronous operation.

Returns

[Task](#) <bool>

A task that evaluates to `true` if the computed hash matches `expectedHash`; otherwise, `false`.

Remarks

This method uses [TransformBlock\(byte\[\], int, int, byte\[\], int\)](#) and [TransformFinalBlock\(byte\[\], int, int\)](#) to compute the hash in buffered blocks and compare it to `expectedHash`.

Exceptions

[ArgumentNullException](#)

Thrown if `algorithm` or `expectedHash` is `null`.

VerifyHashAsync(HashAlgorithm, Stream, ReadOnlyMemory<byte>, CancellationToken)

Asynchronously verifies that the computed hash of a stream matches the expected hash value in memory.

```
public static Task<bool> VerifyHashAsync(this HashAlgorithm algorithm, Stream stream,  
    ReadOnlyMemory<byte> expectedHash, CancellationToken cancellationToken = default)
```

Parameters

algorithm [HashAlgorithm](#)

The [HashAlgorithm](#) instance used to compute the hash.

stream [Stream](#)

The readable stream to hash asynchronously.

`expectedHash` [ReadOnlyMemory<byte>](#)

The expected hash value as a [ReadOnlyMemory<T>](#).

`cancellationToken` [CancellationToken](#)

A token to cancel the asynchronous operation.

Returns

[Task<bool>](#)

A task that evaluates to `true` if the computed hash matches `expectedHash`; otherwise, `false`.

Remarks

This overload enables hash verification using memory buffers to reduce allocations in memory-sensitive scenarios.

Exceptions

[ArgumentNullException](#)

Thrown if `algorithm` is `null`.

VerifyHashAsync(HashAlgorithm, Stream, string, CancellationToken)

Asynchronously verifies that the computed hash of a stream matches the expected hexadecimal hash string.

```
public static Task<bool> VerifyHashAsync(this HashAlgorithm algorithm, Stream stream, string expectedHex, CancellationToken cancellationToken = default)
```

Parameters

`algorithm` [HashAlgorithm](#)

The [HashAlgorithm](#) instance used to compute the hash.

`stream` [Stream](#)

The readable stream to hash asynchronously.

expectedHex [string](#)

The expected hash as a hexadecimal string.

cancellationToken [CancellationToken](#)

A token to cancel the asynchronous operation.

Returns

[Task](#) <[bool](#)>

A task that evaluates to `true` if the computed hash matches `expectedHex`; otherwise, `false`.

Remarks

This method computes the hash, converts it to a hexadecimal string, and compares it using case-insensitive ordinal comparison.

Exceptions

[ArgumentNullException](#)

Thrown if `algorithm` or `expectedHex` is `null`.