System.Decimal Structure

```
[ILAsm]
.class public sequential sealed serializable Decimal extends
System.ValueType implements System.IComparable, System.IFormattable,
System.IComparable`1<valuetype System.Decimal>,
System.IEquatable`1<valuetype System.Decimal>

[C#]
public struct Decimal: IComparable, IFormattable, IComparable<Decimal>,
IEquatable<Decimal>
```

Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- *Version:* 2.0.x.x
- *Attributes:*
    - CLSCompliantAttribute(true)

Implements:

- **System.IFormattable**
- **System.IComparable**
- **System.IComparable<System.Decimal>**
- **System.IEquatable<System.Decimal>**

Summary

Represents a floating-point decimal data type with at least 28 significant digits, suitable for financial and commercial calculations.

Inherits From: System.ValueType

Library: ExtendedNumerics

Thread Safety: This type is not guaranteed to be safe for multithreaded operations.

Description

The `System.Decimal` type can represent values with at least 28 significant digits. The `System.Decimal` data type is ideally suited to financial calculations that require a large number of significant digits and no round-off errors.

The `System.Decimal` type shall represent values in at least the range -79228162514264337593543950335 to 79228162514264337593543950335, and having scale f such that 0 <= *f* <= at least 28.

A `System.Decimal` value occupies 128 bits; however, its representation is unspecified (see note below).

The result of an operation on values of type `System.Decimal` is that which would result from calculating an exact result (preserving scale, as defined for each operator) and then rounding to fit the representation. That is, results are exact to at least 28 digits, but not necessarily to more than 28 decimal places. A zero result has a sign of 0.

Results are rounded to the nearest representable value, and, when a result is equally close to two representable values, to the value that has an even number in the least significant digit position (banker's rounding).

The default initial value of an instance of type `System.Decimal` is zero with a scale of zero.

[*Note:* Unlike the `System.Single` and `System.Double` data types, decimal fractional numbers such as 0.1 can be represented exactly in the `System.Decimal` representation. In the `System.Single` and `System.Double` representations, such numbers are often infinite fractions, making those representations prone to round-off errors.

Further, the `System.Decimal` representation preserves scale, so that 1.23 + 1.27 will give the answer 2.50, not 2.5.

]

If a `System.Decimal` arithmetic operation produces a value that is too small for the `System.Decimal` format after rounding, the result of the operation is zero. If a `System.Decimal` arithmetic operation produces a result that is too large for the `System.Decimal` format, a `System.OverflowException` is thrown.

[*Note:* The `System.Decimal` class implements implicit conversions from the `System.SByte`, `System.Byte`, `System.Int16`, `System.UInt16`, `System.Int32`, `System.UInt32`, `System.Int64`, and `System.UInt64` types to `System.Decimal`. These implicit conversions never lose information and never throw exceptions. The `System.Decimal` class also implements explicit conversions from `System.Decimal` to `System.Byte`, `System.SByte`, `System.Int16`, `System.UInt16`, `System.Int32`, `System.UInt32`, `System.Int64`, and `System.UInt64`. These explicit conversions round the `System.Decimal` value towards zero to the nearest integer, and then convert that integer to the destination type. A `System.OverflowException` is thrown if the result is not within the range of the destination type.

The `System.Decimal` class provides narrowing conversions to and from the `System.Single` and `System.Double` types. A conversion from `System.Decimal` to `System.Single` or `System.Double` can lose precision. If the value being converted is not within the range of the destination type, a `System.OverflowException` is thrown. A conversion from `System.Single` or `System.Double` to `System.Decimal` throws a `System.OverflowException` if the value is not within the range of the `System.Decimal` type.

]

[*Note:* Although different implementations of `System.Decimal` can have different representations, interchange of a decimal value within the range of the internal format can still be achieved by converting it to a string, exporting it, and then converting it back to internal format.

]

[*Note:* In Version 1 of this standard, the representation of `System.Decimal` was well-defined, as follows.

When considered as an array of four `System.Int32`s, it contains the following four elements:

Index 0 (bits 0-31) contains the low-order 32 bits of the decimal's coefficient.

Index 1 (bits 32-63) contains the middle 32 bits of the decimal's coefficient.

Index 2 (bits 64-95) contains the high-order 32 bits of the decimal's coefficient.

Index 3 (bits 96-127) contains the sign bit and scale, as follows:

Bit Positions	**Name**	**Description**
0-15	(None.)	Zero.
16-23	Scale	Contains a value between 0 and 28.
24-30	(None.)	Zero.
31	Sign	0 (positive) or 1 (negative).

In order to allow alternate representations (such as in the update to the IEC floating-point standard, IEC-60559, currently in preparation), the representation has been made unspecified.

]

Decimal(System.Int32) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(int32 value)

[C#]
public Decimal(int value)
```

Summary

Constructs and initializes a new `System.Decimal` value.

Parameters

Parameter	**Description**
value	The `System.Int32` value used to initialize the new `System.Decimal`.

Description

This constructor initializes the new `System.Decimal` to the value specified by *value*. The scale of the new `System.Decimal` is 0.

Decimal(System.UInt32) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(unsigned int32 value)

[C#]
public Decimal(uint value)
```

Summary

Constructs and initializes a new `System.Decimal` value.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
value	The `System.UInt32` value used to initialize the new `System.Decimal`.

Description

This member is not CLS-compliant. For a CLS-compliant alternative, use the `System.Decimal`(`System.Int64`) constructor.

This constructor initializes the new `System.Decimal` to the value specified by *value*. The scale of the new `System.Decimal` is 0.

Decimal(System.Int64) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(int64 value)

[C#]
public Decimal(long value)
```

Summary

Constructs and initializes a new `System.Decimal` value.

Parameters

Parameter	Description
value	The `System.Int64` value used to initialize the new `System.Decimal`.

Description

This constructor initializes the new `System.Decimal` to the value specified by *value*. The scale of the new `System.Decimal` is 0.

Decimal(System.UInt64) Constructor

[ILAsm]
```
public rtspecialname specialname instance void .ctor(unsigned int64 value)
```

[C#]
```
public Decimal(ulong value)
```

Summary

Constructs and initializes a new `System.Decimal` value.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	**Description**
value	The `System.UInt64` value used to initialize the new `System.Decimal`.

Description

This constructor initializes the new `System.Decimal` to the value specified by *value*. The scale of the new `System.Decimal` is 0.

This member is not CLS-compliant. For a CLS-compliant alternative, use the `System.Decimal`(`System.Int64`) constructor.

Decimal(System.Single) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(float32 value)

[C#]
public Decimal(float value)
```

Summary

Constructs and initializes a new `System.Decimal` value.

Parameters

Parameter	Description
value	The `System.Single` value used to initialize the new `System.Decimal`.

Description

This constructor initializes the new `System.Decimal` to the value specified by *value*. This constructor rounds *value* to 7 significant digits using banker's rounding. The scale of the new `System.Decimal` is the same as that produced by `System.Decimal.Parse(`*value*`.ToString())`.

Exceptions

Exception	Condition
System.OverflowException	*value* is one of the following: greater than `System.Decimal.MaxValue` less than `System.Decimal.MinValue` equal to `System.Single.NaN`, but the Decimal representation does not support NaNs. equal to `System.Single.PositiveInfinity`, but the Decimal representation does not support infinities. equal to `System.Single.NegativeInfinity`, but the Decimal representation does not support infinities.

Decimal(System.Double) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(float64 value)

[C#]
public Decimal(double value)
```

Summary

Constructs and initializes a new `System.Decimal` value.

Parameters

Parameter	Description
value	The `System.Double` value used to initialize the new `System.Decimal`.

Description

This constructor initializes the new `System.Decimal` to the value specified by *value*. This constructor rounds *value* to 15 significant digits using banker's rounding. The scale of the new `System.Decimal` is the same as that produced by `System.Decimal.Parse(`*value*`.ToString())`.

Exceptions

Exception	Condition
System.OverflowException	*value* is one of the following: greater than `System.Decimal.MaxValue` less than `System.Decimal.MinValue` equal to `System.Double.NaN`, but the Decimal representation does not support NaNs. equal to `System.Double.PositiveInfinity`, but the Decimal representation does not support infinities. equal to `System.Double.NegativeInfinity`, but the Decimal representation does not support infinities.

Decimal(System.Int32[]) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(int32[] bits)

[C#]
public Decimal(int[] bits)
```

Summary

Constructs and initializes a new `System.Decimal` value.

Parameters

Parameter	Description
bits	An array of four `System.Int32` containing an unspecified 128-bit representation of a `System.Decimal` in its raw form.

Description

This constructor initializes the new `System.Decimal` to the value represented by the elements of *bits*.

[*Note:* A numeric value can have several possible binary representations; they are numerically equal but have different scales. Also, the bit representation differentiates between -0, 0.00, and 0; these are all treated as 0 in operations, and any zero result will have a sign of 0.]

The format of *bits* is the same as that returned by `System.Decimal.GetBits(System.Decimal)`.

Exceptions

Exception	Condition
System.ArgumentNullException	*bits* is a null reference.
System.ArgumentException	*bits* does not contain four elements.

Decimal.MaxValue Field

[ILAsm]
```
.field public static initOnly valuetype System.Decimal MaxValue
```

[C#]
```
public static readonly decimal MaxValue
```

Summary

Contains the maximum positive value for the `System.Decimal` type.

Type Attributes:

- DecimalConstantAttribute [*Note:* This attribute requires the RuntimeInfrastructure library.]

Description

The value of this constant is implementation-specific, but shall be greater than or equal to 79228162514264337593543950335. The scale shall be 0.

This field is read-only.

Decimal.MinusOne Field

[ILAsm]
```
.field public static initOnly valuetype System.Decimal MinusOne
```

[C#]
```
public static readonly decimal MinusOne
```

Summary

Contains negative one (-1).

Type Attributes:

- DecimalConstantAttribute [*Note:* This attribute requires the RuntimeInfrastructure library.]

Description

The value of this constant is -1. The scale shall be 0.

This field is read-only.

Decimal.MinValue Field

```
[ILAsm]
.field public static initOnly valuetype System.Decimal MinValue

[C#]
public static readonly decimal MinValue
```

Summary

Contains the minimum (most negative) value for the `System.Decimal` type.

Type Attributes:

- DecimalConstantAttribute [*Note:* This attribute requires the RuntimeInfrastructure library.]

Description

The value of this constant is implementation-specific, but shall be less than or equal to -79228162514264337593543950335. The scale shall be 0.

This field is read-only.

Decimal.One Field

[ILAsm]
```
.field public static initOnly valuetype System.Decimal One
```

[C#]
```
public static readonly decimal One
```

Summary

Contains one (1).

Type Attributes:

- DecimalConstantAttribute [*Note:* This attribute requires the RuntimeInfrastructure library.]

Description

The value of this constant is 1. The scale shall be 0.

This field is read-only.

Decimal.Zero Field

[ILAsm]
```
.field public static initOnly valuetype System.Decimal Zero
```

[C#]
```
public static readonly decimal Zero
```

Summary

Contains zero (0).

Type Attributes:

- DecimalConstantAttribute [*Note:* This attribute requires the RuntimeInfrastructure library.]

Description

The value of this constant is 0. The scale shall be 0.

This field is read-only.

Decimal.Add(System.Decimal, System.Decimal) Method

```
[ILAsm]
.method public hidebysig static valuetype System.Decimal Add(valuetype
System.Decimal d1, valuetype System.Decimal d2)

[C#]
public static decimal Add(decimal d1, decimal d2)
```

Summary

Adds two `System.Decimal` values together.

Parameters

Parameter	**Description**
d1	The first addend.
d2	The second addend.

Return Value

A `System.Decimal` containing the sum of *d1* and *d2*. The scale of the result, before any rounding, is the larger of the scales of d1 and d2. For example, 1.1 + 2.22 gives 3.32, and 2.50 + 1 gives 3.50.

Exceptions

Exception	**Condition**
System.OverflowException	The sum of *d1* and *d2* is less than `System.Decimal.MinValue` or greater than `System.Decimal.MaxValue`.

Decimal.Compare(System.Decimal, System.Decimal) Method

```
[ILAsm]
.method public hidebysig static int32 Compare(valuetype System.Decimal d1,
valuetype System.Decimal d2)

[C#]
public static int Compare(decimal d1, decimal d2)
```

Summary

Compares the values of two `System.Decimal` values and returns sort order information.

Parameters

Parameter	Description
d1	The first value to compare.
d2	The second value to compare.

Return Value

The return value is a negative number, zero, or a positive number reflecting the sort order of *d1* as compared to *d2*. Trailing zero digits in the fractional part of are ignored. For non-zero return values, the exact value returned by this method is unspecified. The following table defines the return value. Each comparison compares the numerical values of *d1* and *d2*.

Return Value	Meaning
Any negative number	*d1*< *d2*
Zero	*d1* == *d2*
A positive number	*d1*>*d2*

Decimal.CompareTo(System.Decimal) Method

```
[ILAsm]
.method public final hidebysig virtual int32 CompareTo(valuetype
System.Decimal value)

[C#]
public int CompareTo(decimal value)
```

Summary

Returns the sort order of the current instance compared to the specified `System.Decimal`.

Parameters

Parameter	Description
value	The `System.Decimal` to compare to the current instance.

Return Value

The return value is a negative number, zero, or a positive number reflecting the sort order of the current instance as compared to *value*. Trailing zero digits in the fractional part of the current instance and *value*. For non-zero return values, the exact value returned by this method is unspecified. The following table defines the return value:

Return Value	Description
A negative number	Current instance < *value*.
Zero	Current instance == *value*.
A positive number	current instance > *value*.

Description

[*Note:* This method is implemented to support the `System.IComparable<Decimal>` interface.]

Decimal.CompareTo(System.Object) Method

```
[ILAsm]
.method public final hidebysig virtual int32 CompareTo(object value)

[C#]
public int CompareTo(object value)
```

Summary

Returns the sort order of the current instance compared to the specified `System.Object`.

Parameters

Parameter	Description
value	The `System.Object` to compare to the current instance.

Return Value

The return value is a negative number, zero, or a positive number reflecting the sort order of the current instance as compared to *value*. Trailing zero digits in the fractional part of the current instance and *value*. For non-zero return values, the exact value returned by this method is unspecified. The following table defines the return value:

Return Value	Description
A negative number	Current instance < *value*.
Zero	Current instance == *value*.
A positive number	current instance > *value*, or *value* is a null reference.

Description

[*Note:* This method is implemented to support the `System.IComparable` interface.]

Exceptions

Exception	Condition

System.ArgumentException	*value* is not a `System.Decimal` and is not a null reference.

Decimal.Divide(System.Decimal, System.Decimal) Method

```
[ILAsm]
.method public hidebysig static valuetype System.Decimal Divide(valuetype
System.Decimal d1, valuetype System.Decimal d2)

[C#]
public static valuetype System.Decimal Divide(valuetype System.Decimal d1,
valuetype System.Decimal d2)
```

Summary

Divides the value of one `System.Decimal` by another.

Parameters

Parameter	**Description**
d1	The dividend.
d2	The divisor.

Return Value

A `System.Decimal` containing the result of dividing *d1* by *d2*. The scale of the result, before any rounding, is the closest scale to the preferred scale which will preserve a result equal to the exact result. The preferred scale is the scale of *d1* less the scale of *d2*. For example, 2.20 / 2 gives 1.10.

Exceptions

Exception	**Condition**
System.DivideByZeroException	*d2* is zero.
System.OverflowException	The result is greater than `System.Decimal.MaxValue` or less than `System.Decimal.MinValue`.

Decimal.Equals(System.Decimal) Method

```
[ILAsm]
.method public hidebysig virtual bool Equals(valuetype System.Decimal
value)

[C#]
public override bool Equals(decimal value)
```

Summary

Determines whether the current instance and the specified `System.Decimal` have the same value. Trailing zero digits in the fractional part are ignored.

Parameters

Parameter	Description
value	The `System.Decimal` to compare to the current instance.

Return Value

`true` if *value* is numerically equal to (has the same value as) the current instance; otherwise, `false`.

Description

[*Note:* This method is implemented to support the `System.IEquatable<Decimal>` interface.]

Decimal.Equals(System.Object) Method

[ILAsm]
```
.method public hidebysig virtual bool Equals(object value)
```

[C#]
```
public override bool Equals(object value)
```

Summary

Determines whether the current instance and the specified `System.Object` have the same type and value. Trailing zero digits in the fractional part are ignored.

Parameters

Parameter	Description
value	The `System.Object` to compare to the current instance.

Return Value

`true` if *value* has the same type and is numerically equal to (has the same value as) the current instance. If *value* is a null reference or is not an instance of `System.Decimal`, returns `false`.

Description

[*Note:* This method overrides `System.Object.Equals`.]

Decimal.Equals(System.Decimal, System.Decimal) Method

```
[ILAsm]
.method public hidebysig static bool Equals(valuetype System.Decimal d1,
valuetype System.Decimal d2)

[C#]
public static bool Equals(decimal d1, decimal d2)
```

Summary

Determines whether two `System.Decimal` values have the same value. Trailing zero digits in the fractional part are ignored.

Parameters

Parameter	Description
d1	The first `System.Decimal` to compare.
d2	The second `System.Decimal` to compare.

Return Value

`true` if *d1* and *d2* are numerically equal (have the same value), otherwise `false`.

Decimal.Floor(System.Decimal) Method

[ILAsm]
```
.method public hidebysig static valuetype System.Decimal Floor(valuetype System.Decimal d)
```

[C#]
```
public static decimal Floor(decimal d)
```

Summary

Rounds a `System.Decimal` value to the closest integer towards negative infinity.

Parameters

Parameter	Description
d	The `System.Decimal` to round downward.

Return Value

A `System.Decimal` value *v* such that *v* is an integer and *d* - 1 < *v* <= *d*. The scale of the result will be zero.

Example

The following example demonstrates the `System.Decimal.Floor` method.

[C#]

```
using System;
class DecimalTest {
 public static void Main() {
   Console.WriteLine("floor {0} is {1}", 3.14159m, Decimal.Floor(3.14159m));
   Console.WriteLine("floor {0} is {1}", -3.9m, Decimal.Floor(-3.9m));
   Console.WriteLine("floor {0} is {1}", 3.0m, Decimal.Floor(3.0m));
 }
}
```

The output is

```
floor 3.14159 is 3


floor -3.9 is -4


floor 3.0 is 3
```

1

Decimal.GetBits(System.Decimal) Method

[ILAsm]

```
.method public hidebysig static class int32[] GetBits(valuetype
System.Decimal d)
```

[C#]

```
public static int[] GetBits(decimal d)
```

Summary

Returns a binary representation of the specified `System.Decimal` value.

Parameters

Parameter	Description
d	The `System.Decimal` value for which a binary representation is returned.

Return Value

An array of four `System.Int32` containing an unspecified 128-bit representation of a `System.Decimal` in its raw form.

Description

The format of the bits in the 4-element array returned is the same as that used by the *bits* parameter to `System.Decimal.Decimal(System.Int32)`.

A numeric value can have several possible binary representations; they are numerically equal but have different scales. Also, the bit representation differentiates between -0, 0.00, and 0; these are all treated as 0 in operations, and any zero result will have a sign of 0.

Example

The following example demonstrates the different representations of 1.00 and 1.

[C#]

```
using System;
public class Class1  {
    public static void Print (int [] bs) {
        foreach (int b in bs) {
            Console.Write (b+" ");
        }
    }
public static void Main () {
   decimal d = 1.00m;
```

```
        decimal d1 = 1;
        Console.Write (d);
        Console.Write (" - bits: ");
        Print (decimal.GetBits(d));
        Console.WriteLine();
        Console.Write (d1);
        Console.Write (" - bits: ");
        Print (decimal.GetBits(d1));
        Console.WriteLine();
        Console.WriteLine ("d1.CompareTo(d) == {0}", d1.CompareTo(d));
        Console.WriteLine ("d1 == d {0}", d1 == d);
    }
}
```

Decimal.GetHashCode() Method

[ILAsm]
```
.method public hidebysig virtual int32 GetHashCode()
```

[C#]
```
public override int GetHashCode()
```

Summary

Generates a hash code for the current instance. Trailing zero digits in the fractional part are ignored.

Return Value

A `System.Int32` containing the hash code for this instance.

Description

The algorithm used to generate the hash code value is unspecified.

[*Note:* This method overrides `System.Object.GetHashCode`.]

Decimal.Multiply(System.Decimal, System.Decimal) Method

```
[ILAsm]
.method public hidebysig static valuetype System.Decimal
Multiply(valuetype System.Decimal d1, valuetype System.Decimal d2)

[C#]
public static decimal Multiply(decimal d1, decimal d2)
```

Summary

Returns the result of multiplying two `System.Decimal` values.

Parameters

Parameter	Description
d1	The multiplier.
d2	The multiplicand.

Return Value

The result of multiplying *d1* and *d2*. The scale of the result, before any rounding, is the sum of the scales of d1 and d2.

For example, 123 x 3 gives 369, and 2.2 x 1.35 gives 2.970.

Exceptions

Exception	Condition
System.OverflowException	The result is greater than `System.Decimal.MaxValue` or less than `System.Decimal.MinValue`.

Decimal.Negate(System.Decimal) Method

[ILAsm]
```
.method public hidebysig static valuetype System.Decimal Negate(valuetype System.Decimal d)
```

[C#]
```
public static decimal Negate(decimal d)
```

Summary

Returns the result of multiplying a `System.Decimal` value by negative one.

Parameters

Parameter	Description
d	The value to negate.

Return Value

The negated value of *d*. If *d* is zero then zero is returned (with 0 sign); otherwise the scale of the result is the same as the scale of *d*.

Decimal.op_Addition(System.Decimal, System.Decimal) Method

```
[ILAsm]
.method public hidebysig static specialname valuetype System.Decimal
op_Addition(valuetype System.Decimal d1, valuetype System.Decimal d2)

[C#]
public static Decimal operator +(Decimal d1, Decimal d2)
```

Summary

Adds two `System.Decimal` values together.

Parameters

Parameter	**Description**
d1	The first addend.
d2	The second addend.

Return Value

The value returned by `System.Decimal.Add` (*d1*,*d2*).

Exceptions

Exception	**Condition**
System.OverflowException	The sum of *d1* and *d2* is greater than `System.Decimal.MaxValue` or less than `System.Decimal.MinValue`.

Decimal.op_Decrement(System.Decimal) Method

```
[ILAsm]
.method public hidebysig static specialname valuetype System.Decimal
op_Decrement(valuetype System.Decimal d)

[C#]
public static Decimal operator --(Decimal d)
```

Summary

Returns the specified value decremented by one.

Parameters

Parameter	Description
d	A `System.Decimal` value.

Return Value

The value returned by `System.Decimal.Subtract` (*d*, `System.Decimal.One`).

Exceptions

Exception	Condition
System.OverflowException	The result is greater than `System.Decimal.MaxValue` or less than `System.Decimal.MinValue`.

Decimal.op_Division(System.Decimal, System.Decimal) Method

```
[ILAsm]
.method public hidebysig static specialname valuetype System.Decimal
op_Division(valuetype System.Decimal d1, valuetype System.Decimal d2)

[C#]
public static Decimal operator /(Decimal d1, Decimal d2)
```

Summary

Divides one `System.Decimal` value by another `System.Decimal`.

Parameters

Parameter	Description
d1	The dividend.
d2	The divisor.

Return Value

The value returned by `System.Decimal.Divide` (*d1*, *d2*).

Exceptions

Exception	Condition
System.DivideByZeroException	The divisor is zero.
System.OverflowException	The result is greater than `System.Decimal.MaxValue` or less than `System.Decimal.MinValue`.

Decimal.op_Equality(System.Decimal, System.Decimal) Method

```
[ILAsm]
.method public hidebysig static specialname bool op_Equality(valuetype System.Decimal d1, valuetype System.Decimal d2)

[C#]
public static bool operator ==(Decimal d1, Decimal d2)
```

Summary

Determines whether two decimals have the same value.

Parameters

Parameter	Description
d1	The first `System.Decimal` to compare.
d2	The second `System.Decimal` to compare.

Return Value

`true` if `System.Decimal.Compare` (*d1*, *d2*) returns zero; otherwise `false`.

Decimal.op_Explicit(System.Decimal) Method

[ILAsm]
```
.method public hidebysig static specialname float64 op_Explicit(valuetype
System.Decimal value)
```

[C#]
```
public static explicit operator double(Decimal value)
```

Summary

Perform an explicit conversion of a `System.Decimal` value to `System.Double`.

Parameters

Parameter	Description
value	The `System.Decimal` value to convert to `System.Double`.

Return Value

A `System.Double` with the specified value.

Description

[*Note:* This operation can produce round-off errors due to the fact that `System.Double` has fewer significant digits than, and has a different radix than, `System.Decimal`.]

Decimal.op_Explicit(System.Decimal) Method

[ILAsm]

```
.method public hidebysig static specialname float32 op_Explicit(valuetype
System.Decimal value)
```

[C#]

```
public static explicit operator float(Decimal value)
```

Summary

Perform an explicit conversion of a `System.Decimal` value to `System.Single`.

Parameters

Parameter	Description
value	The `System.Decimal` value to convert to `System.Single`.

Return Value

A `System.Single` with the specified value.

[*Note:* This operation can produce round-off errors due to the fact that `System.Single` has fewer significant digits than, and has a different radix than, `System.Decimal`.]

Decimal.op_Explicit(System.Decimal) Method

```
[ILAsm]
.method public hidebysig static specialname unsigned int64
op_Explicit(valuetype System.Decimal value)

[C#]
public static explicit operator ulong(Decimal value)
```

Summary

Perform an explicit conversion of a `System.Decimal` value to `System.UInt64`.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
value	The `System.Decimal` value to convert to `System.UInt64`.

Return Value

A `System.UInt64` containing *value* rounded towards zero to the nearest integer.

Description

This member is not CLS-compliant. For a CLS-compliant alternative to `System.UInt64`, use `System.Int64`.

Exceptions

Exception	Condition
System.OverflowException	The resulting integer value is greater than `System.UInt64.MaxValue` or less than `System.UInt64.MinValue`.

Decimal.op_Explicit(System.Decimal) Method

[ILAsm]
```
.method public hidebysig static specialname int64 op_Explicit(valuetype System.Decimal value)
```

[C#]
```
public static explicit operator long(Decimal value)
```

Summary

Perform an explicit conversion of a `System.Decimal` value to `System.Int64`.

Parameters

Parameter	**Description**
value	The `System.Decimal` value to convert to `System.Int64`.

Return Value

A `System.Int64` containing *value* rounded towards zero to the nearest integer.

Exceptions

Exception	**Condition**
System.OverflowException	The resulting integer value is greater than `System.Int64.MaxValue` or less than `System.Int64.MinValue`.

Decimal.op_Explicit(System.Decimal) Method

```
[ILAsm]
.method public hidebysig static specialname unsigned int32
op_Explicit(valuetype System.Decimal value)

[C#]
public static explicit operator uint(Decimal value)
```

Summary

Perform an explicit conversion of a `System.Decimal` value to `System.UInt32`.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
value	The `System.Decimal` value to convert to `System.UInt32`.

Return Value

A `System.UInt32` containing *value* rounded towards zero to the nearest integer.

Description

This member is not CLS-compliant. For a CLS-compliant alternative to `System.UInt32`, use `System.Int64`).

Exceptions

Exception	Condition
System.OverflowException	The resulting integer value is greater than `System.UInt32.MaxValue` or less than `System.UInt32.MinValue`.

Decimal.op_Explicit(System.Decimal) Method

```
[ILAsm]
.method public hidebysig static specialname int32 op_Explicit(valuetype
System.Decimal value)

[C#]
public static explicit operator int(Decimal value)
```

Summary

Perform an explicit conversion of a `System.Decimal` value to `System.Int32`.

Parameters

Parameter	Description
value	The `System.Decimal` value to convert to `System.Int32`.

Return Value

A `System.Int32` containing *value* rounded towards zero to the nearest integer.

Exceptions

Exception	Condition
System.OverflowException	The resulting integer value is greater than `System.Int32.MaxValue` or less than `System.Int32.MinValue`.

Decimal.op_Explicit(System.Decimal) Method

```
[ILAsm]
.method public hidebysig static specialname int8 op_Explicit(valuetype
System.Decimal value)

[C#]
public static explicit operator sbyte(Decimal value)
```

Summary

Perform an explicit conversion of a `System.Decimal` value to `System.SByte`.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
value	The `System.Decimal` value to convert to `System.SByte`.

Return Value

A `System.SByte` containing *value* rounded towards zero to the nearest integer.

Description

This member is not CLS-compliant. For a CLS-compliant alternative to `System.SByte`, use `System.Int16`.

Exceptions

Exception	Condition
System.OverflowException	The resulting integer value is greater than `System.SByte.MaxValue` or less than `System.SByte.MinValue`.

Decimal.op_Explicit(System.Decimal) Method

[ILAsm]
```
.method public hidebysig static specialname unsigned int8
op_Explicit(valuetype System.Decimal value)
```

[C#]
```
public static explicit operator byte(Decimal value)
```

Summary

Perform an explicit conversion of a `System.Decimal` value to `System.Byte`.

Parameters

Parameter	Description
value	The `System.Decimal` value to convert to `System.Byte`.

Return Value

A `System.Byte` containing *value* rounded towards zero to the nearest integer.

Exceptions

Exception	Condition
System.OverflowException	The resulting integer value is greater than `System.Byte.MaxValue` or less than `System.Byte.MinValue`.

Decimal.op_Explicit(System.Double) Method

```
[ILAsm]
.method public hidebysig static specialname valuetype System.Decimal
op_Explicit(float64 value)

[C#]
public static explicit operator Decimal(double value)
```

Summary

Perform an explicit conversion of a `System.Double` value to `System.Decimal`.

Parameters

Parameter	**Description**
value	The `System.Double` value to convert to `System.Decimal`.

Return Value

A `System.Decimal` with the specified value.

Exceptions

Exception	**Condition**
System.OverflowException	*value* is one of the following: greater than `System.Decimal.MaxValue` less than `System.Decimal.MinValue` equal to `System.Double.NaN`, but the Decimal representation does not support NaNs. equal to `System.Double.PositiveInfinity`, but the Decimal representation does not support infinities. equal to `System.Double.NegativeInfinity`, but the Decimal representation does not support infinities.

Decimal.op_Explicit(System.Single) Method

```
[ILAsm]
.method public hidebysig static specialname valuetype System.Decimal
op_Explicit(float32 value)

[C#]
public static explicit operator Decimal(float value)
```

Summary

Perform an explicit conversion of a `System.Single` value to `System.Decimal`.

Parameters

Parameter	**Description**
value	The `System.Single` value to convert to `System.Decimal`.

Return Value

A `System.Decimal` with the specified value.

Exceptions

Exception	**Condition**
System.OverflowException	*value* is one of the following: greater than `System.Decimal.MaxValue` less than `System.Decimal.MinValue` equal to `System.Single.NaN`, but the Decimal representation does not support NaNs. equal to `System.Single.PositiveInfinity`, but the Decimal representation does not support infinities. equal to `System.Single.NegativeInfinity`, but the Decimal representation does not support infinities.

Decimal.op_Explicit(System.Decimal) Method

[ILAsm]
```
.method public hidebysig static specialname valuetype System.Char
op_Explicit(valuetype System.Decimal value)
```

[C#]
```
public static explicit operator char(Decimal value)
```

Summary

Perform an explicit conversion of a `System.Decimal` value to `System.Char`.

Parameters

Parameter	Description
value	The `System.Decimal` value to convert to `System.Char`.

Return Value

A `System.Char` containing *value* rounded towards zero to the nearest integer.

Exceptions

Exception	Condition
System.OverflowException	The resulting integer value is greater than `System.Char.MaxValue` or less than `System.Char.MinValue`.

Decimal.op_Explicit(System.Decimal) Method

```
[ILAsm]
.method public hidebysig static specialname unsigned int16
op_Explicit(valuetype System.Decimal value)

[C#]
public static explicit operator ushort(Decimal value)
```

Summary

Perform an explicit conversion of a `System.Decimal` value to `System.UInt16`.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	**Description**
value	The `System.Decimal` value to convert to `System.UInt16`.

Return Value

A `System.UInt16` containing *value* rounded towards zero to the nearest integer.

Description

This member is not CLS-compliant. For a CLS-compliant alternative to `System.UInt16`, use `System.Int32`.

Exceptions

Exception	**Condition**
System.OverflowException	The resulting integer value is greater than `System.UInt16.MaxValue` or less than `System.UInt16.MinValue`.

Decimal.op_Explicit(System.Decimal) Method

```
[ILAsm]
.method public hidebysig static specialname int16 op_Explicit(valuetype
System.Decimal value)

[C#]
public static explicit operator short(Decimal value)
```

Summary

Perform an explicit conversion of a `System.Decimal` value to `System.Int16`.

Parameters

Parameter	Description
value	The `System.Decimal` value to convert to `System.Int16`.

Return Value

A `System.Int16` containing *value* rounded towards zero to the nearest integer.

Exceptions

Exception	Condition
System.OverflowException	The resulting integer value is greater than `System.Int16.MaxValue` or less than `System.Int16.MinValue`.

Decimal.op_GreaterThan(System.Decimal, System.Decimal) Method

```
[ILAsm]
.method public hidebysig static specialname bool op_GreaterThan(valuetype
System.Decimal d1, valuetype System.Decimal d2)

[C#]
public static bool operator >(Decimal d1, Decimal d2)
```

Summary

Determines whether one `System.Decimal` value is greater than another `System.Decimal` value.

Parameters

Parameter	Description
d1	The first `System.Decimal` to compare.
d2	The second `System.Decimal` to compare.

Return Value

`true` if `System.Decimal.Compare` (*d1*, *d2*) returns a value that is greater than zero; otherwise `false`.

Decimal.op_GreaterThanOrEqual(System.Decimal, System.Decimal) Method

[ILAsm]
```
.method public hidebysig static specialname bool
op_GreaterThanOrEqual(valuetype System.Decimal d1, valuetype
System.Decimal d2)
```

[C#]
```
public static bool operator >=(Decimal d1, Decimal d2)
```

Summary

Determines whether one `System.Decimal` value is greater than or equal to another `System.Decimal` value.

Parameters

Parameter	Description
d1	The first `System.Decimal` to compare.
d2	The second `System.Decimal` to compare.

Return Value

`true` if `System.Decimal.Compare` (*d1*, *d2*) returns a value that is greater than or equal to zero; otherwise `false`.

Decimal.op_Implicit(System.Byte) Method

[ILAsm]
```
.method public hidebysig static specialname valuetype System.Decimal
op_Implicit(unsigned int8 value)
```

[C#]
```
public static implicit operator Decimal(byte value)
```

Summary

Perform an implicit conversion of a `System.Byte` value to `System.Decimal`.

Parameters

Parameter	Description
value	The `System.Byte` value to convert to `System.Decimal`.

Return Value

A `System.Decimal` with the specified value.

Decimal.op_Implicit(System.SByte) Method

```
[ILAsm]
.method public hidebysig static specialname valuetype System.Decimal
op_Implicit(int8 value)

[C#]
public static implicit operator Decimal(sbyte value)
```

Summary

Perform an implicit conversion of a `System.SByte` value to `System.Decimal`.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
value	The `System.SByte` value to convert to `System.Decimal`.

Return Value

A `System.Decimal` with the specified value.

Description

This member is not CLS-compliant.

Decimal.op_Implicit(System.Int16) Method

[ILAsm]
```
.method public hidebysig static specialname valuetype System.Decimal
op_Implicit(int16 value)
```

[C#]
```
public static implicit operator Decimal(short value)
```

Summary

Perform an implicit conversion of a `System.Int16` value to `System.Decimal`.

Parameters

Parameter	Description
value	The `System.Int16` value to convert to `System.Decimal`.

Return Value

A `System.Decimal` with the specified value.

Decimal.op_Implicit(System.UInt16) Method

```
[ILAsm]
.method public hidebysig static specialname valuetype System.Decimal
op_Implicit(unsigned int16 value)

[C#]
public static implicit operator Decimal(ushort value)
```

Summary

Perform an implicit conversion of a `System.UInt16` value to `System.Decimal`.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
value	The `System.UInt16` value to convert to `System.Decimal`.

Return Value

A `System.Decimal` with the specified value.

Description

This member is not CLS-compliant.

Decimal.op_Implicit(System.Char) Method

```
[ILAsm]
.method public hidebysig static specialname valuetype System.Decimal
op_Implicit(valuetype System.Char value)

[C#]
public static implicit operator Decimal(char value)
```

Summary

Perform an implicit conversion of a `System.Char` value to `System.Decimal`.

Parameters

Parameter	Description
value	The `System.Char` value to convert to `System.Decimal`.

Return Value

A `System.Decimal` with the specified value.

Decimal.op_Implicit(System.Int32) Method

[ILAsm]
```
.method public hidebysig static specialname valuetype System.Decimal
op_Implicit(int32 value)
```

[C#]
```
public static implicit operator Decimal(int value)
```

Summary

Perform an implicit conversion of a `System.Int32` value to `System.Decimal`.

Parameters

Parameter	Description
value	The `System.Int32` value to convert to `System.Decimal`.

Return Value

A `System.Decimal` with the specified value.

Decimal.op_Implicit(System.UInt32) Method

```
[ILAsm]
.method public hidebysig static specialname valuetype System.Decimal
op_Implicit(unsigned int32 value)

[C#]
public static implicit operator Decimal(uint value)
```

Summary

Perform an implicit conversion of a `System.UInt32` value to `System.Decimal`.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
value	The `System.UInt32` value to convert to `System.Decimal`.

Return Value

A `System.Decimal` with the specified value.

Description

This member is not CLS-compliant.

Decimal.op_Implicit(System.Int64) Method

[ILAsm]
```
.method public hidebysig static specialname valuetype System.Decimal
op_Implicit(int64 value)
```

[C#]
```
public static implicit operator Decimal(long value)
```

Summary

Perform an implicit conversion of a `System.Int64` value to `System.Decimal`.

Parameters

Parameter	Description
value	The `System.Int64` value to convert to `System.Decimal`.

Return Value

A `System.Decimal` with the specified value.

Decimal.op_Implicit(System.UInt64) Method

```
[ILAsm]
.method public hidebysig static specialname valuetype System.Decimal
op_Implicit(unsigned int64 value)

[C#]
public static implicit operator Decimal(ulong value)
```

Summary

Perform an implicit conversion of a `System.UInt64` value to `System.Decimal`.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
value	The `System.UInt64` value to convert to `System.Decimal`.

Return Value

A `System.Decimal` with the specified value.

Description

This member is not CLS-compliant.

Decimal.op_Increment(System.Decimal) Method

```
[ILAsm]
.method public hidebysig static specialname valuetype System.Decimal
op_Increment(valuetype System.Decimal d)

[C#]
public static Decimal operator ++(Decimal d)
```

Summary

Returns the specified value incremented by one.

Parameters

Parameter	Description
d	A `System.Decimal` value.

Return Value

The value returned by `System.Decimal.Add` (*d*, `System.Decimal.One`).

Exceptions

Exception	Condition
System.OverflowException	The result is greater than `System.Decimal.MaxValue` or less than `System.Decimal.MinValue`.

Decimal.op_Inequality(System.Decimal, System.Decimal) Method

```
[ILAsm]
.method public hidebysig static specialname bool op_Inequality(valuetype System.Decimal d1, valuetype System.Decimal d2)

[C#]
public static bool operator !=(Decimal d1, Decimal d2)
```

Summary

Determines whether two decimals do not have the same value.

Parameters

Parameter	Description
d1	The first `System.Decimal` to compare.
d2	The second `System.Decimal` to compare.

Return Value

`true` if `System.Decimal.Compare` (*d1*, *d2*) does not return zero; otherwise `false`.

Decimal.op_LessThan(System.Decimal, System.Decimal) Method

```
[ILAsm]
.method public hidebysig static specialname bool op_LessThan(valuetype
System.Decimal d1, valuetype System.Decimal d2)

[C#]
public static bool operator <(Decimal d1, Decimal d2)
```

Summary

Determines whether one `System.Decimal` value is less than another `System.Decimal` value.

Parameters

Parameter	Description
d1	The first `System.Decimal` to compare.
d2	The first `System.Decimal` to compare.

Return Value

`true` if `System.Decimal.Compare` (*d1*, *d2*) returns a value that is less than zero; otherwise `false`.

Decimal.op_LessThanOrEqual(System.Decimal, System.Decimal) Method

```
[ILAsm]
.method public hidebysig static specialname bool
op_LessThanOrEqual(valuetype System.Decimal d1, valuetype System.Decimal
d2)

[C#]
public static bool operator <=(Decimal d1, Decimal d2)
```

Summary

Determines whether one `System.Decimal` value is less than or equal to another `System.Decimal` value.

Parameters

Parameter	Description
d1	The first `System.Decimal` to compare.
d2	The second `System.Decimal` to compare.

Return Value

`true` if `System.Decimal.Compare` (*d1*, *d2*) returns a value that is less than or equal to zero; otherwise `false`.

Decimal.op_Modulus(System.Decimal, System.Decimal) Method

```
[ILAsm]
.method public hidebysig static specialname valuetype System.Decimal
op_Modulus(valuetype System.Decimal d1, valuetype System.Decimal d2)

[C#]
public static Decimal operator %(Decimal d1, Decimal d2)
```

Summary

Divides one `System.Decimal` value by another `System.Decimal` and returns the remainder.

Parameters

Parameter	Description
d1	The dividend.
d2	The divisor.

Return Value

The value returned by `System.Decimal.Remainder` (*d1*, *d2*).

Exceptions

Exception	Condition
System.DivideByZeroException	*d2* is zero.
System.OverflowException	*d1* divided by *d2* is greater than `System.Decimal.MaxValue` or less than `System.Decimal.MinValue`.

Decimal.op_Multiply(System.Decimal, System.Decimal) Method

```
[ILAsm]
.method public hidebysig static specialname valuetype System.Decimal
op_Multiply(valuetype System.Decimal d1, valuetype System.Decimal d2)

[C#]
public static Decimal operator *(Decimal d1, Decimal d2)
```

Summary

Returns the result of multiplying two `System.Decimal` values.

Parameters

Parameter	**Description**
d1	The multiplier.
d2	The multiplicand.

Return Value

The value returned by `System.Decimal.Multiply` (*d1*, *d2*).

Exceptions

Exception	**Condition**
System.OverflowException	The result is greater than `System.Decimal.MaxValue` or less than `System.Decimal.MinValue`.

Decimal.op_Subtraction(System.Decimal, System.Decimal) Method

```
[ILAsm]
.method public hidebysig static specialname valuetype System.Decimal
op_Subtraction(valuetype System.Decimal d1, valuetype System.Decimal d2)

[C#]
public static Decimal operator -(Decimal d1, Decimal d2)
```

Summary

Subtracts one `System.Decimal` value from another.

Parameters

Parameter	**Description**
d1	The minuend.
d2	The subtrahend.

Return Value

The value returned by `System.Decimal.Subtract` (*d1*, *d2*).

Exceptions

Exception	**Condition**
System.OverflowException	The result is greater than `System.Decimal.MaxValue` or less than `System.Decimal.MinValue`.

Decimal.op_UnaryNegation(System.Decimal) Method

[ILAsm]
```
.method public hidebysig static specialname valuetype System.Decimal
op_UnaryNegation(valuetype System.Decimal d)
```

[C#]
```
public static Decimal operator -(Decimal d)
```

Summary

Returns the specified value multiplied by negative one (-1).

Parameters

Parameter	Description
d	A `System.Decimal` value.

Return Value

The value returned by `System.Decimal.Negate` (*d*).

Decimal.op_UnaryPlus(System.Decimal) Method

```
[ILAsm]
.method public hidebysig static specialname valuetype System.Decimal
op_UnaryPlus(valuetype System.Decimal d)

[C#]
public static Decimal operator +(Decimal d)
```

Summary

Returns the specified value.

Parameters

Parameter	Description
d	A `System.Decimal` value.

Return Value

Returns *d*.

Decimal.Parse(System.String) Method

```
[ILAsm]
.method public hidebysig static valuetype System.Decimal Parse(string s)

[C#]
public static valuetype System.Decimal Parse(string s)
```

Summary

Returns the specified `System.String` converted to a `System.Decimal` value.

Parameters

Parameter	Description
s	A `System.String` containing the value to convert. The string is interpreted using the `System.Globalization.NumberStyles.Number` style, preserving scale.

Return Value

The `System.Decimal` value obtained from *s*.

Description

This version of `System.Decimal.Parse` is equivalent to `System.Decimal.Parse` (*s*, `System.Globalization.NumberStyles.Number`, `null`).

The string *s* is parsed using the formatting information in a `System.Globalization.NumberFormatInfo` initialized for the current system culture. [*Note:* For more information, see `System.Globalization.NumberFormatInfo.CurrentInfo`.]

If necessary, the value of s is rounded using banker's rounding. Any scale apparent in the string s is preserved unless the value is rounded. If the value is zero, the sign will be 0. Hence the string "2.900" will be parsed to form the decimal with sign 0, coefficient 2900, and scale 3.

Exceptions

Exception	Condition
System.ArgumentNullException	*s* is a null reference.

System.FormatException	*s* is not in the correct format.
System.OverflowException	*s* represents a number greater than `System.Decimal.MaxValue` or less than `System.Decimal.MinValue`.

Example

The following example demonstrates the `System.Decimal.Parse` method.

[C#]

```
using System;
using System.Globalization;
class DecimalParseClass {
  public static void Main() {
    string s1 = " -1.001  ";
    string s2 = "+1,000,111.99";
    string s3 = "2.900";
    Console.WriteLine("String: {0} (decimal) {1}",s1,Decimal.Parse(s1));
    Console.WriteLine("String: {0} (decimal) {1}",s2,Decimal.Parse(s2));
    Console.WriteLine("String: {0} (decimal) {1}",s3,Decimal.Parse(s3));
  }
}
```

The output is

```
String: -1.001 (decimal) -1.001


String: +1,000,111.99 (decimal) 1000111.99


String: 2.900 (decimal) 2.900
```

Decimal.Parse(System.String, System.Globalization.NumberStyles) Method

```
[ILAsm]
.method public hidebysig static valuetype System.Decimal Parse(string s,
valuetype System.Globalization.NumberStyles style)

[C#]
public static valuetype System.Decimal Parse(string s, NumberStyles style)
```

Summary

Returns the specified `System.String` converted to a `System.Decimal` value.

Parameters

Parameter	Description
s	A `System.String` containing the value to convert. The string is interpreted using the style specified by *style*, preserving scale.
style	Zero or more `System.Globalization.NumberStyles` values that specify the style of *s*. Specify multiple values for *style* using the bitwise OR operator. If *style* is a null reference, the string is interpreted using the `System.Globalization.NumberStyles.Number` style.

Return Value

The `System.Decimal` value obtained from *s*.

Description

This version of `System.Decimal.Parse` is equivalent to `System.Decimal.Parse` (*s*, *style*, `null`).

The string *s* is parsed using the formatting information in a `System.Globalization.NumberFormatInfo` initialized for the current system culture. [*Note:* For more information, see `System.Globalization.NumberFormatInfo.CurrentInfo`.]

If necessary, the value of *s* is rounded using banker's rounding.

Exceptions

Exception	Condition
System.ArgumentNullException	*s* is a null reference.
System.FormatException	*s* is not in the correct style.
System.OverflowException	*s* represents a number greater than `System.Decimal.MaxValue` or less than `System.Decimal.MinValue`.

Example

The following example demonstrates supplying `System.Globalization.NumberStyles` values to the `System.Decimal.Parse` method to allow for a symbol separating groups of digits, and a decimal separator. This example uses the symbols from the U.S. English culture, namely a comma and a decimal point.

[C#]

```
using System;
using System.Globalization;
class DecimalParseClass {
public static void Main() {
 string s = "1,000,111.99";
 NumberStyles ns = NumberStyles.AllowThousands |
NumberStyles.AllowDecimalPoint;
 decimal d = Decimal.Parse(s,ns);
 Console.WriteLine("{0} parsed to decimal {1}",s,d);
}
}
```

The output is

```
1,000,111.99 parsed to decimal 1000111.99
```

Decimal.Parse(System.String, System.IFormatProvider) Method

```
[ILAsm]
.method public hidebysig static valuetype System.Decimal Parse(string s,
class System.IFormatProvider provider)

[C#]
public static decimal Parse(string s, IFormatProvider provider)
```

Summary

Returns the specified `System.String` converted to a `System.Decimal` value.

Parameters

Parameter	Description
s	A `System.String` containing the value to convert. The `System.String` is interpreted using the `System.Globalization.NumberStyles.Number` style, preserving scale.
provider	A `System.IFormatProvider` that supplies a `System.Globalization.NumberFormatInfo` containing culture-specific formatting information about *s*.

Return Value

The `System.Decimal` value obtained from *s*.

Description

This version of `System.Decimal.Parse` is equivalent to `System.Decimal.Parse` (*s*, `System.Globalization.NumberStyles.Number`, *provider*).

The string *s* is parsed using the culture-specific formatting information from the `System.Globalization.NumberFormatInfo` instance supplied by *provider*. If *provider* is `null` or a `System.Globalization.NumberFormatInfo` cannot be obtained from *provider*, the formatting information for the current system culture is used.

If necessary, the value of s is rounded using banker's rounding. Any scale apparent in the string s is preserved unless the value is rounded. If the value is zero, the sign scale will be 0. Hence the string "2.900" will be parsed to form the decimal with sign 0, coefficient 2900, and scale 3.

Exceptions

Exception	Condition
System.FormatException	*s* is not in the correct style.
System.OverflowException	*s* represents a number greater than `System.Decimal.MaxValue` or less than `System.Decimal.MinValue`.
System.ArgumentNullException	*s* is a null reference.

Example

The following example demonstrates supplying a `System.IFormatProvider` to the `System.Decimal.Parse` method to allow a decimal point, and commas separating groups of digits.

[C#]

```
using System;
using System.Globalization;
class DecimalParseClass {
public static void Main() {
 string s = "1,000,111.99";
 //Get the default formatting symbols.
 NumberFormatInfo nfi = new NumberFormatInfo();
 // Default group separator is ','
 // Default decimal separator is '.'
 decimal d = Decimal.Parse(s,nfi);
 Console.WriteLine("{0} parsed to decimal {1}",s,d);
}
}
```

The output is

```
1,000,111.99 parsed to decimal 1000111.99
```

Decimal.Parse(System.String, System.Globalization.NumberStyles, System.IFormatProvider) Method

[ILAsm]
```
.method public hidebysig static valuetype System.Decimal Parse(string s,
valuetype System.Globalization.NumberStyles style, class
System.IFormatProvider provider)
```

[C#]
```
public static decimal Parse(string s, NumberStyles style, IFormatProvider
provider)
```

Summary

Returns the specified `System.String` converted to a `System.Decimal` value.

Parameters

Parameter	Description
s	A `System.String` containing the value to convert. The string is interpreted using the style specified by *style*, preserving scale.
style	Zero or more `System.Globalization.NumberStyles` values that specify the style of *s*. Specify multiple values for *style* using the bitwise OR operator. If *style* is a null reference, the string is interpreted using the `System.Globalization.NumberStyles.Number` style.
provider	A `System.IFormatProvider` that supplies a `System.Globalization.NumberFormatInfo` containing culture-specific formatting information about *s*.

Return Value

The `System.Decimal` value obtained from *s*.

Description

The string *s* is parsed using the culture-specific formatting information from the `System.Globalization.NumberFormatInfo` instance supplied by *provider*. If *provider* is `null` or if a `System.Globalization.NumberFormatInfo` cannot be obtained from *provider*, the formatting information for the current system culture is used.

If necessary, the value of *s* is rounded using banker's rounding.

Exceptions

Exception	Condition
System.ArgumentNullException	*s* is a null reference.
System.FormatException	*s* is not in the correct style.
System.OverflowException	*s* represents a number greater than `System.Decimal.MaxValue` or less than `System.Decimal.MinValue`.

Example

The following example demonstrates supplying `System.Globalization.NumberStyles` values and a `System.IFormatProvider` to the `System.Decimal.Parse` method to allow colons separating groups of digits, and a decimal point.

[C#]

```
using System;
using System.Globalization;
class DecimalParseClass {
public static void Main() {
 string s = "1:000:111.99";
 NumberStyles ns = NumberStyles.AllowThousands |
NumberStyles.AllowDecimalPoint;
 NumberFormatInfo nfi = new NumberFormatInfo();
 //Change the format info to separate digit groups using a colon.
 nfi.NumberGroupSeparator = ":";
 decimal d = Decimal.Parse(s,ns,nfi);
 Console.WriteLine("{0} parsed to decimal {1}",s,d);
}
}
```

The output is

```
1:000:111.99 parsed to decimal 1000111.99
```

Decimal.Remainder(System.Decimal, System.Decimal) Method

```
[ILAsm]
.method public hidebysig static valuetype System.Decimal
Remainder(valuetype System.Decimal d1, valuetype System.Decimal d2)

[C#]
public static decimal Remainder(decimal d1, decimal d2)
```

Summary

Computes the remainder after dividing two `System.Decimal` values.

Parameters

Parameter	**Description**
d1	The dividend.
d2	The divisor.

Return Value

The remainder after dividing d1 by d2 to give an integer result. The sign of the result, if non-zero, is the same as the sign of d1, and the scale of the result is the larger of the scales of d1 and d2.

For example, -10 % 3 gives -1, and 3.6 % 1.3 gives 1.0 (where % indicates the remainder operation).

Exceptions

Exception	**Condition**
System.DivideByZeroException	*d2* is zero.
System.OverflowException	*d1* divided by *d2* is greater than `System.Decimal.MaxValue` or less than `System.Decimal.MinValue`.

Decimal.Round(System.Decimal, System.Int32) Method

```
[ILAsm]
.method public hidebysig static valuetype System.Decimal Round(valuetype
System.Decimal d, int32 decimals)

[C#]
public static decimal Round(decimal d, int decimals)
```

Summary

Rounds a `System.Decimal` value to a specified number of decimal places.

Parameters

Parameter	Description
d	The `System.Decimal` to round.
decimals	The number of decimal places to round to. 0 <= *decimals* <= 28.

Return Value

The `System.Decimal` result of rounding *d* to *decimals* decimal places.

Description

When *d* is exactly half way between two rounded values, the result is the rounded value that has an even digit in the rightmost decimal position. For example, when rounded to two decimals, the value 2.345 becomes 2.34 and the value 2.355 becomes 2.36. [*Note:* This process is known as rounding half towards even, or banker's rounding.]

The scale of the result will be the smaller of *decimals* and the scale of d.

[*Note:* The scale of *d* is never increased, so `System.Decimal.Round` cannot cause overflow.]

Exceptions

Exception	Condition

System.ArgumentOutOfRangeException	*decimals* is not between 0 and 28, inclusive.

Example

The following example demonstrates the `System.Decimal.Round` method.

[C#]

```
using System;
class MyClass {
public static void Main() {
 decimal d1 = 2.5m;
 decimal d2 = 3.5m;
 decimal d3 = 2.98765432m;
 decimal d4 = 2.18765432m;
 Console.WriteLine("Rounding to 0 places...");
 Console.WriteLine("round {0} = {1}",d1, Decimal.Round(d1,0));
 Console.WriteLine("round {0} = {1}",d2, Decimal.Round(d2,0));
 Console.WriteLine("round {0} = {1}",d3, Decimal.Round(d3,0));
 Console.WriteLine("round {0} = {1}",d4, Decimal.Round(d4,0));
 Console.WriteLine("Rounding to 2 places...");
 Console.WriteLine("round {0} = {1}",d1, Decimal.Round(d1,2));
 Console.WriteLine("round {0} = {1}",d2, Decimal.Round(d2,2));
 Console.WriteLine("round {0} = {1}",d3, Decimal.Round(d3,2));
 Console.WriteLine("round {0} = {1}",d4, Decimal.Round(d4,2));
}
}
```

The output is

```
Rounding to 0 places...


round 2.5 = 2


round 3.5 = 4


round 2.98765432 = 3


round 2.18765432 = 2


Rounding to 2 places...


round 2.5 = 2.5
round 3.5 = 3.5


round 2.98765432 = 2.99


round 2.18765432 = 2.19


```

Decimal.Subtract(System.Decimal, System.Decimal) Method

```
[ILAsm]
.method public hidebysig static valuetype System.Decimal
Subtract(valuetype System.Decimal d1, valuetype System.Decimal d2)

[C#]
public static decimal Subtract(decimal d1, decimal d2)
```

Summary

Subtracts one `System.Decimal` value from another.

Parameters

Parameter	Description
d1	The left-side operand.
d2	The right-side operand.

Return Value

A `System.Decimal` containing the result of subtracting *d2* from *d1*. The scale of the result, before any rounding, is the larger of the scales of d1 and d2.

For example, 1.1 - 2.22 gives -1.12, and 2.50 - 1 gives 1.50.

Exceptions

Exception	Condition
System.OverflowException	The result is greater than `System.Decimal.MaxValue` or less than `System.Decimal.MinValue`.

Decimal.ToString(System.String) Method

```
[ILAsm]
.method public hidebysig instance string ToString(string format)

[C#]
public string ToString(string format)
```

Summary

Returns a `System.String` representation of the value of the current instance.

Parameters

Parameter	Description
format	A `System.String` that specifies the format of the returned string. [*Note:* For a list of valid values, see `System.Decimal.ToString` (`System.String`, `System.IFormatProvider`).]

Return Value

A `System.String` representation of the current instance formatted as specified by *format*. The string takes into account the current system culture.

Description

This version of `System.Decimal.ToString` is equivalent to `System.Decimal.ToString` (*format*, `null`).

If *format* is a null reference, the general format specifier "G" is used.

Exceptions

Exception	Condition
System.FormatException	*format* is invalid.

Example

The following example shows the effects of various formats on the string returned by `System.Decimal.ToString`.

[C#]

```
using System;
```

```
class test {
 public static void Main() {
 decimal d = 1234.56789m;
 Console.WriteLine(d);
 string[] fmts = {"C","E","F","G","N","P"};
 for (int i=0;i<fmts.Length;i++)
 Console.WriteLine("{0}: {1}",
 fmts[i],d.ToString(fmts[i]));
 }
}
```

The output is

```
1234.56789


C: $1,234.57


E: 1.234568E+003


F: 1234.57


G: 1234.56789


N: 1,234.57


P: 123,456.79 %
```

Decimal.ToString() Method

```
[ILAsm]
.method public hidebysig virtual string ToString()

[C#]
public override string ToString()
```

Summary

Returns a canonical `System.String` representation of the value of the current instance.

Return Value

A `System.String` representation of the current instance formatted using the general format specifier, ("G"). The string takes into account the current system culture and preserves the scale of the number.

Description

This version of `System.Decimal.ToString` is equivalent to `System.Decimal.ToString` (`null`, `null`).

[*Note:* The general format specifier formats the number in either fixed-point or exponential notation form, preserving the scale of the number. For a detailed description of the general format, see the `System.IFormattable` interface.

This method overrides `System.Object.ToString`.

]

Decimal.ToString(System.String, System.IFormatProvider) Method

```
[ILAsm]
.method public final hidebysig virtual string ToString(string format,
class System.IFormatProvider provider)

[C#]
public string ToString(string format, IFormatProvider provider)
```

Summary

Returns a `System.String` representation of the value of the current instance.

Parameters

Parameter	Description
format	A `System.String` containing a character that specifies the format of the returned string, optionally followed by a non-negative integer that specifies the precision of the number in the returned `System.String`.
provider	A `System.IFormatProvider` that supplies a `System.Globalization.NumberFormatInfo` instance containing culture-specific formatting information.

Return Value

A `System.String` representation of the current instance formatted as specified by *format*. The string takes into account the information in the `System.Globalization.NumberFormatInfo` instance supplied by *provider*.

Description

If *provider* is `null` or if a `System.Globalization.NumberFormatInfo` cannot be obtained from *provider*, the formatting information for the current system culture is used.

The following table lists the characters that are valid for the *format* parameter.

Format Characters	Description
"C", "c"	Currency format.
"E", "e"	Exponential notation format.

"F", "f"	Fixed-point format.
"G", "g"	General format.
"N", "n"	Number format.
"P", "p"	Percent format.

If *format* is a null reference, the general format specifier "G" is used.

[*Note:* For a detailed description of formatting, see the `System.IFormattable` interface.

This method is implemented to support the `System.IFormattable` interface.

]

Exceptions

Exception	**Condition**
System.FormatException	*format* is invalid.

Decimal.ToString(System.IFormatProvider) Method

```
[ILAsm]
.method public final hidebysig virtual string ToString(class
System.IFormatProvider provider)

[C#]
public string ToString(IFormatProvider provider)
```

Summary

Returns a `System.String` representation of the value of the current instance.

Parameters

Parameter	Description
provider	A `System.IFormatProvider` that supplies a `System.Globalization.NumberFormatInfo` containing culture-specific formatting information.

Return Value

A `System.String` representation of the current instance formatted using the general format specifier, ("G"). The string takes into account the formatting information in the `System.Globalization.NumberFormatInfo` instance supplied by *provider*.

Description

This version of `System.Decimal.ToString` is equivalent to `System.Decimal.ToString` (`null`, *provider*).

If *provider* is `null` or if a `System.Globalization.NumberFormatInfo` cannot be obtained from *provider*, the formatting information for the current system culture is used.

[*Note:* The general format specifier formats the number in either fixed-point or exponential notation form. For a detailed description of the general format, see the `System.IFormattable` interface.]

Decimal.Truncate(System.Decimal) Method

[ILAsm]
```
.method public hidebysig static valuetype System.Decimal
Truncate(valuetype System.Decimal d)
```

[C#]
```
public static decimal Truncate(decimal d)
```

Summary

Rounds a `System.Decimal` value towards zero, to the closest integer value.

Parameters

Parameter	Description
d	The `System.Decimal` to truncate.

Return Value

The `System.Decimal` result of truncating *d*. the scale of the result is 0.

Example

The following example demonstrates using the `System.Decimal.Truncate` method.

[C#]

```
using System;
class MyClass {
public static void Main() {
 decimal d1 = 1234.56789m;
 decimal d2 = -1234.56789m;
 Console.WriteLine("{0} truncated is {1}", d1, Decimal.Truncate(d1));
 Console.WriteLine("{0} truncated is {1}", d2, Decimal.Truncate(d2));
}
}
```

The output is

```
1234.56789 truncated is 1234


-1234.56789 truncated is -1234
```