MD5 Class

Namespace: System.Security.Cryptography

Assemblies: System.Security.Cryptography.Algorithms.dll, mscorlib.dll, netstandard.dll

Represents the abstract class from which all implementations of the MD5 hash algorithm inherit.

In this article

C# Copy

```
[System.Runtime.InteropServices.ComVisible(true)]
public abstract class MD5 : System.Security.Cryptography.HashAlgorithm
```

Inheritance Object → HashAlgorithm → MD5

Derived System.Security.Cryptography.MD5Cng
System.Security.Cryptography.MD5CryptoServiceProvider

Attributes ComVisibleAttribute

Examples

The following code example computes the MD5 hash value of a string and returns the hash as a 32-character, hexadecimal-formatted string. The hash string created by this code example is compatible with any MD5 hash function (on any platform) that creates a 32-character, hexadecimal-formatted hash string.

C# Copy

```
using System;
using System.Security.Cryptography;
using System.Text;

namespace MD5Sample
{
    class Program
    {
        static void Main(string[] args)
        {
            string source = "Hello World!";
            using (MD5 md5Hash = MD5.Create())
            {
```

```
        {
            string hash = GetMd5Hash(md5Hash, source);

            Console.WriteLine("The MD5 hash of " + source + " is: " + hash + ".");

            Console.WriteLine("Verifying the hash...");

            if (VerifyMd5Hash(md5Hash, source, hash))
            {
                Console.WriteLine("The hashes are the same.");
            }
            else
            {
                Console.WriteLine("The hashes are not same.");
            }

        }



    }
    static string GetMd5Hash(MD5 md5Hash, string input)
    {

        // Convert the input string to a byte array and compute the hash.
        byte[] data = md5Hash.ComputeHash(Encoding.UTF8.GetBytes(input));

        // Create a new Stringbuilder to collect the bytes
        // and create a string.
        StringBuilder sBuilder = new StringBuilder();

        // Loop through each byte of the hashed data
        // and format each one as a hexadecimal string.
        for (int i = 0; i < data.Length; i++)
        {
            sBuilder.Append(data[i].ToString("x2"));
        }

        // Return the hexadecimal string.
```

```
            return sBuilder.ToString();
        }

        // Verify a hash against a string.
        static bool VerifyMd5Hash(MD5 md5Hash, string input, string hash)
        {
            // Hash the input.
            string hashOfInput = GetMd5Hash(md5Hash, input);

            // Create a StringComparer an compare the hashes.
            StringComparer comparer = StringComparer.OrdinalIgnoreCase;

            if (0 == comparer.Compare(hashOfInput, hash))
            {

                return true;
            }
            else
            {
                return false;
            }
        }

    }
}

// This code example produces the following output:
//
// The MD5 hash of Hello World! is: ed076287532e86365e841e92bfc50d8c.
// Verifying the hash...
// The hashes are the same.
```

Remarks

Hash functions map binary strings of an arbitrary length to small binary strings of a fixed length. A cryptographic hash function has the

property that it is computationally infeasible to find two distinct inputs that hash to the same value; that is, hashes of two sets of data should match if the corresponding data also matches. Small changes to the data result in large, unpredictable changes in the hash.

The hash size for the MD5 algorithm is 128 bits.

The ComputeHash methods of the MD5 class return the hash as an array of 16 bytes. Note that some MD5 implementations produce a 32-character, hexadecimal-formatted hash. To interoperate with such implementations, format the return value of the ComputeHash methods as a hexadecimal value.

> **Note**
>
> Newer hash functions, such as the Secure Hash Algorithms SHA-256 and SHA-512, are available. Consider using the **SHA256** class or the **SHA512** class instead of the **MD5** class. Use **MD5** only for compatibility with legacy applications and data.

Constructors

MD5()	Initializes a new instance of MD5.

Methods

Clear()	Releases all resources used by the HashAlgorithm class. (Inherited from HashAlgorithm)
ComputeHash(Byte[])	Computes the hash value for the specified byte array. (Inherited from HashAlgorithm)

ComputeHash(Byte[], Int32, Int32)	Computes the hash value for the specified region of the specified byte array. (Inherited from HashAlgorithm)
ComputeHash(Stream)	Computes the hash value for the specified Stream object. (Inherited from HashAlgorithm)
Create()	Creates an instance of the default implementation of the MD5 hash algorithm.
Create(String)	Creates an instance of the specified implementation of the MD5 hash algorithm.
Dispose()	Releases all resources used by the current instance of the HashAlgorithm class. (Inherited from HashAlgorithm)
Dispose(Boolean)	Releases the unmanaged resources used by the HashAlgorithm and optionally releases the managed resources. (Inherited from HashAlgorithm)
Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object)
GetHashCode()	Serves as the default hash function. (Inherited from Object)
GetType()	Gets the Type of the current instance. (Inherited from Object)
HashCore(Byte[], Int32, Int32)	When overridden in a derived class, routes data written to the object into the hash algorithm for computing the hash. (Inherited from HashAlgorithm)

HashCore(ReadOnlySpan<Byte>)	Inherited from HashAlgorithm
HashFinal()	When overridden in a derived class, finalizes the hash computation after the last data is processed by the cryptographic stream object. (Inherited from HashAlgorithm)
Initialize()	Initializes an implementation of the HashAlgorithm class. (Inherited from HashAlgorithm)
MemberwiseClone()	Creates a shallow copy of the current Object. (Inherited from Object)
ToString()	Returns a string that represents the current object. (Inherited from Object)
TransformBlock(Byte[], Int32, Int32, Byte[], Int32)	Computes the hash value for the specified region of the input byte array and copies the specified region of the input byte array to the specified region of the output byte array. (Inherited from HashAlgorithm)
TransformFinalBlock(Byte[], Int32, Int32)	Computes the hash value for the specified region of the specified byte array. (Inherited from HashAlgorithm)
TryComputeHash(ReadOnlySpan<Byte>, Span<Byte>, Int32)	Inherited from HashAlgorithm
TryHashFinal(Span<Byte>, Int32)	Inherited from HashAlgorithm
CanReuseTransform	Gets a value indicating whether the current transform can be reused. (Inherited from HashAlgorithm)

CanTransformMultipleBlocks	When overridden in a derived class, gets a value indicating whether multiple blocks can be transformed. (Inherited from HashAlgorithm)
Hash	Gets the value of the computed hash code. (Inherited from HashAlgorithm)
HashSize	Gets the size, in bits, of the computed hash code. (Inherited from HashAlgorithm)
InputBlockSize	When overridden in a derived class, gets the input block size. (Inherited from HashAlgorithm)
OutputBlockSize	When overridden in a derived class, gets the output block size. (Inherited from HashAlgorithm)
HashSizeValue	Represents the size, in bits, of the computed hash code. (Inherited from HashAlgorithm)
HashValue	Represents the value of the computed hash code. (Inherited from HashAlgorithm)
State	Represents the state of the hash computation. (Inherited from HashAlgorithm)
IDisposable.Dispose()	Releases the unmanaged resources used by the HashAlgorithm and optionally releases the managed resources. (Inherited from HashAlgorithm)

Applies to