System.Uri Class

[ILAsm]
```
.class public serializable Uri extends System.Object
```

[C#]
```
public class Uri
```

Assembly Info:

- *Name:* System
- *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)

Summary

Provides an object representation of a uniform resource identifier (URI) as defined by IETF RFC 2396.

Inherits From: System.Object

Library: Networking

Thread Safety: All public static members of this type are safe for multithreaded operations. No instance members are guaranteed to be thread safe.

Description

[*Note:* A Uniform Resource Identifier (URI) is a compact string of characters used to identify a resource located on a computer. A resource can be anything that has identity. Examples of resources that might be accessed using a URI include an electronic document, an image, a web service, and a collection of other resources. A URI is represented as a sequence of characters. While the exact format of a URI is determined by the protocol used to access the resource, many URI consist of four major components:

<scheme>://*<authority><path>*?*<query>*

Scheme - Indicates a protocol used to access the resource.

Authority - Indicates the naming authority (server or registry) that governs the namespace defined by the remainder of the URI. The authority component is composed of *userinfo*, *host* and *port* subcomponents in the form *<userinfo>*@*<host>*:*<port>*. Only the *host* subcomponent is required to be present in the *Authority* component. Authority information is stored in the `System.Uri.Authority` property.

Path - Identifies the resource within the scope of the scheme and, if present, the authority. This information is stored in the `System.Uri.AbsolutePath`, `System.Uri.PathAndQuery`, and `System.Uri.LocalPath` properties.

Query - Parameter information that is passed to the executable script identified by the URI. The query, if present, is the last element in a URI and begins with a "?". This information is stored in the `System.Uri.Query` property.

Userinfo - [Subcomponent of *Authority*] Consists of a user name and, optionally, scheme-specific authorization information used to access *Host*. The *userinfo*, if present, is separated from the *Host* component by the "@" character. Note that for some URI schemes, the format of the *userinfo* subcomponent is "username:password". Passing authorization information in this manner is strongly discouraged due to security issues. The *userinfo* information is stored in the `System.Uri.UserInfo` property.

Host - [Subcomponent of *Authority*] The Domain Name system (DNS) name or IP4 address of a machine that provides access to the resource. This information is stored in the `System.Uri.Host` property.

Port - [Subcomponent of *Authority*] The network port number used to connect to the host. If no port number is specified in the URI, most schemes designate protocols that have a default port number. This information is stored in the `System.Uri.Port` property.

Fragment - The fragment is not part of the URI, but is used in conjunction with the URI and is included here for completeness. This component contains resource-specific information that is used after a resource is retrieved. The *fragment*, if present, is separated from the URI by the "#" character. This information is stored in the `System.Uri.Fragment` property.

URIs include components consisting of or delimited by certain special (reserved) characters that have a special meaning in a URI component. If the reserved meaning is not intended, then the character is required to be escaped in the URI. An escaped character is encoded as a character triplet consisting of the percent character "%" followed by the US-ASCII character code specified as two hexadecimal digits. For example, "%20" is the escaped encoding for the US-ASCII space character. The URI represented by a `System.Uri` instance is always in "escaped" form. The following characters are reserved:

- Semi-colon (";")
- Forward slash ("/")
- Question mark ("?")
- Colon (":")
- At-sign ("@")
- Ampersand ("&")

- Equal sign ("=")
- Plus sign ("+")
- US Dollar sign ("$")
- Comma (",")

To transform the URI contained in a `System.Uri` instance from an escape encoded URI to a human-readable URI, use the `System.Uri.ToString` method.

]

URIs are stored as canonical URIs in escaped encoding, with all characters with ASCII values greater than 127 replaced with their hexadecimal equivalents. The `System.Uri` constructors do not escape URI strings if the string is a well-formed URI, including a scheme identifier, that contains escape sequences. To put the URI in canonical form, the `System.Uri` constructors perform the following steps.

- Converts the URI scheme to lowercase.
- Converts the host name to lowercase.
- Removes default and empty port numbers.
- Simplifies the URI by removing superfluous segments such as "/" and "/test" segments.

The `System.Uri` class stores only absolute URIs (for example, "http://www.contoso.com/index.htm"). Relative URIs (for example, "/new/index.htm") are expanded to absolute form using a specified base URI. The `System.Uri.MakeRelative` method converts absolute URIs to relative URIs.

The `System.Uri` class properties are read-only; to modify a `System.Uri` instance use the `System.UriBuilder` class.

Uri(System.String) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(string
uriString)

[C#]
public Uri(string uriString)
```

Summary

Constructs and initializes a new instance of the `System.Uri` class by parsing the specified URI.

Parameters

Parameter	Description
uriString	A `System.String` containing a URI.

Description

This constructor is equivalent to calling the `System.Uri` (`System.String`, `System.Boolean`) constructor, and specifying *uriString* and `false` as the arguments.

Exceptions

Exception	Condition
System.ArgumentNullException	*uriString* is `null`.
System.UriFormatException	*uriString* is a zero length string or contains only spaces. -or- *uriString* is in an invalid form and cannot be parsed.

Uri(System.String, System.Boolean) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(string
uriString, bool dontEscape)

[C#]
public Uri(string uriString, bool dontEscape)
```

Summary

Constructs and initializes a new instance of the `System.Uri` class by parsing the specified URI.

Parameters

Parameter	Description
uriString	A `System.String` containing a URI.
dontEscape	`true` if the URI in *uriString* is already escaped; otherwise, `false`.

Description

This constructor parses the URI, places its components into the appropriate properties, and puts the URI in canonical form. If the specified URI does not contain a scheme component, the URI is parsed using "file" as the scheme.

Exceptions

Exception	Condition
System.ArgumentNullException	*uriString* is `null`.
System.UriFormatException	*uriString* is a zero length string or contains only spaces. -or- The parsing routine detected a scheme in an invalid form. -or- The parser detected more than two consecutive slashes in a URI that does not use the "file" scheme.

	-or- *uriString* is in an invalid form and cannot be parsed.

Example

The following example creates a `System.Uri` instance for the URI "http://www.contoso.com/Hello%20World.htm". Because the URI contains escaped characters, the third parameter, *dontEscape*, is set to `true`.

[C#]

```
using System;

public class UriTest {
 public static void Main() {

 Uri myUri = new Uri("http://www.contoso.com/Hello%20World.htm", true);

 Console.WriteLine(myUri.ToString());
 }
}
```

The output is

```
http://www.contoso.com/Hello World.htm
```

Uri(System.Uri, System.String) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(class
System.Uri baseUri, string relativeUri)

[C#]
public Uri(Uri baseUri, string relativeUri)
```

Summary

Constructs and initializes a new instance of the `System.Uri` class by combining the specified base and relative URIs.

Parameters

Parameter	Description
baseUri	A `System.Uri` containing a base URI.
relativeUri	A `System.String` containing a relative URI.

Description

This constructor is equivalent to calling the `System.Uri` (`System.Uri`, `System.String`, `System.Boolean`) constructor, and specifying *baseUri*, *relativeUri*, and `false` as the arguments.

Exceptions

Exception	Condition
System.UriFormatException	*relativeUri* is in an invalid form.
System.NullReferenceException	*baseUri* is `null`.

Uri(System.Uri, System.String, System.Boolean) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(class
System.Uri baseUri, string relativeUri, bool dontEscape)

[C#]
public Uri(Uri baseUri, string relativeUri, bool dontEscape)
```

Summary

Constructs and initializes a new instance of the `System.Uri` class by combining the specified base and relative URIs.

Parameters

Parameter	Description
baseUri	A `System.Uri` containing the base URI. This parameter can, but is not required to contain a terminating slash ("/") character.
relativeUri	A `System.String` containing the relative URI to add to the base URI. This parameter can, but is not required to contain a leading slash ("/") character.
dontEscape	`true` if *baseUri* and *relativeUri* are already escaped; otherwise, `false`.

Description

This constructor compensates for the presence or absence of a terminating slash in *baseUri* and/or a leading slash in *relativeUri* to produce a well-formed URI.

If the relative URI contains a `System.Uri.Scheme` that is the same as the scheme of the base URI and the `System.Uri.SchemeDelimiter` is not present, or the relative URI does not contain a scheme, the new instance is composed of the relative URI (without its scheme component, if any) qualified by the scheme and authority information from the base URI.

If the relative URI contains a `System.Uri.Scheme` followed by the `System.Uri.SchemeDelimiter`, it is treated as an absolute URI and the base URI is ignored. If the relative URI contains a scheme that differs from the scheme of the base URI, the base URI is ignored. If the `System.Uri.SchemeDelimiter` is not present in the relative URI, it is assumed, and the new instance is constructed as though the relative URI were an absolute URI.

[*Note:* When the base URI is ignored, only the components of the relative URI are used to construct the new instance.]

Exceptions

Exception	**Condition**
System.UriFormatException	*relativeUri* is in an invalid form.
System.NullReferenceException	*baseUri* is `null`.

Example

The following example creates new instances of the `System.Uri` class by combining a `System.Uri` instance representing the base URI and a string containing a relative URI.

[C#]

```
using System;

public class UriTest {
 public static void Main() {

 // Typical base and relative URI constructor usage.

 Uri baseUri = new Uri("http://www.contoso.com", true);
 Uri myUri = new Uri(baseUri, "index.htm",true);
 Console.WriteLine("Typical usage: {0}",myUri.ToString());

 // Base and relative URI contain slashes.
 Uri baseUri2 = new Uri("http://www.contoso.com/", true);
 Uri myUri2 = new Uri(baseUri2, "/index.htm",true);
 Console.WriteLine("Slash example: {0}",myUri2.ToString());

 // Relative URI contains a different scheme than the base URI.
 Uri baseUri3 = new Uri("http://www.contoso.com/", true);
 Uri myUri3 = new Uri(baseUri3,
"ftp://www.contoso2.com/index.htm",true);
 Console.WriteLine("Different schemes: {0}", myUri3.ToString());


 // Relative URI contains the same scheme as the base URI.
 // The scheme delimiter is not present in the relative URI.
 Uri baseUri4 = new Uri("http://www.contoso.com/", true);
 Uri myUri4 = new Uri(baseUri4,
"http:www.contoso2.com/index.htm",true);
 Console.WriteLine("Same schemes - relative treated as relative:
{0}",myUri4.ToString());

 // Relative URI contains the same scheme as the base URI.
 // The scheme delimiter is present in the relative URI.
 Uri baseUri5 = new Uri("http://www.contoso.com/", true);
 Uri myUri5 = new Uri(baseUri5, "http://www.contoso2/index.htm",true);
 Console.WriteLine("Same schemes - relative treated as absolute:
{0}",myUri5.ToString());

 }
}
```

The output is

```
Typical usage: http://www.contoso.com/index.htm


Slash example: http://www.contoso.com/index.htm


Different schemes: ftp://www.contoso2.com/index.htm


Same schemes - relative treated as relative:
http://www.contoso.com/www.contoso2.com/index.htm


Same schemes - relative treated as absolute:
http://www.contoso2/index.htm
```

Uri.SchemeDelimiter Field

[ILAsm]
```
.field public static initOnly string SchemeDelimiter
```

[C#]
```
public static readonly string SchemeDelimiter
```

Summary

A `System.String` containing the characters that separate the scheme component from the remainder of a URI.

Description

This field is read-only. The value of this field is "://".

Uri.UriSchemeFile Field

[ILAsm]
```
.field public static initOnly string UriSchemeFile
```

[C#]
```
public static readonly string UriSchemeFile
```

Summary

A `System.String` containing the characters that indicate that a URI identifies a file.

Description

This field is read-only. The value of this field is "file".

Uri.UriSchemeFtp Field

[ILAsm]
```
.field public static initOnly string UriSchemeFtp
```

[C#]
```
public static readonly string UriSchemeFtp
```

Summary

A `System.String` containing the characters that indicate that a URI is accessed through the File Transfer Protocol (FTP).

Description

This field is read-only. The value of this field is "ftp".

Uri.UriSchemeGopher Field

[ILAsm]
```
.field public static initOnly string UriSchemeGopher
```

[C#]
```
public static readonly string UriSchemeGopher
```

Summary

A `System.String` containing the characters that indicate that a URI is accessed through the Gopher protocol.

Description

This field is read-only. The value of this field is "gopher".

Uri.UriSchemeHttp Field

```
[ILAsm]
.field public static initOnly string UriSchemeHttp

[C#]
public static readonly string UriSchemeHttp
```

Summary

A `System.String` containing the characters that indicate that a URI is accessed through the Hypertext Transfer Protocol (HTTP).

Description

This field is read-only. The value of this field is "http".

Uri.UriSchemeHttps Field

[ILAsm]
```
.field public static initOnly string UriSchemeHttps
```

[C#]
```
public static readonly string UriSchemeHttps
```

Summary

A `System.String` containing the characters that indicate that a URI is accessed through the Secure Hypertext Transfer Protocol (HTTPS).

Description

This field is read-only. The value of this field is "https".

Uri.UriSchemeMailto Field

[ILAsm]
```
.field public static initOnly string UriSchemeMailto
```

[C#]
```
public static readonly string UriSchemeMailto
```

Summary

A `System.String` containing the characters that indicate that a URI is an email address and is accessed through the Simple Network Mail Protocol (SNMP).

Description

This field is read-only. The value of this field is "mailto".

Uri.UriSchemeNews Field

[ILAsm]
```
.field public static initOnly string UriSchemeNews
```

[C#]
```
public static readonly string UriSchemeNews
```

Summary

A `System.String` containing the characters that indicate that a URI is an Internet news group and is accessed through the Network News Transport Protocol (NNTP).

Description

This field is read-only. The value of this field is "news".

Uri.UriSchemeNntp Field

[ILAsm]
```
.field public static initOnly string UriSchemeNntp
```

[C#]
```
public static readonly string UriSchemeNntp
```

Summary

A `System.String` containing the characters that indicate that a URI is an Internet news group and is accessed through the Network News Transport Protocol (NNTP).

Description

This field is read-only. The value of this field is "nntp".

Uri.Canonicalize() Method

[ILAsm]
```
.method family hidebysig virtual void Canonicalize()
```

[C#]
```
protected virtual void Canonicalize()
```

Summary

Converts the components of the URI represented by the current instance to canonical form.

Behaviors

This method converts the URI to a format suitable for machine interpretation according to the scheme of the current instance. The conversions are required to preserve all information that could, if removed or altered, change the URI represented by the current instance.

Default

This method performs the following conversions:

- Converts file references to the format of the current platform, for example on a Windows system, file://c|/AFile.txt is converted to "file:///c:/AFile.txt".
- Converts any backslash characters ('\') to forward slashes ('/').
- Compresses multiple consecutive forward slashes ('/') in the path component to a single forward slash.
- Compresses any path meta sequences ("/." and "/..").

How and When to Override

Override this method to canonicalize the type derived from `System.Uri`.

Usage

Applications do not call this method; it is called by constructors after parsing the URI and escaping the components.

Uri.CheckHostName(System.String) Method

[ILAsm]

```
.method public hidebysig static valuetype System.UriHostNameType
CheckHostName(string name)
```

[C#]

```
public static UriHostNameType CheckHostName(string name)
```

Summary

Returns a value that describes the format of a host name string.

Parameters

Parameter	Description
name	A `System.String` containing the host name to validate.

Return Value

A `System.UriHostNameType` that indicates the type of the host name. If the type of the host name cannot be determined, or the host name is `null` or a zero-length string, returns `System.UriHostNameType.Unknown`.

Example

The following example demonstrates using the `System.Uri.CheckHostName` method.

[C#]

```
using System;

public class UriTest {
 public static void Main() {

 Console.WriteLine(Uri.CheckHostName("www.contoso.com"));
 }
}
```

The output is

```
Dns
```

Uri.CheckSchemeName(System.String) Method

```
[ILAsm]
.method public hidebysig static bool CheckSchemeName(string
schemeName)

[C#]
public static bool CheckSchemeName(string schemeName)
```

Summary

Returns a `System.Boolean` value indicating whether the specified scheme name is valid.

Parameters

Parameter	Description
schemeName	A `System.String` containing the scheme name to validate.

Return Value

`true` if the scheme name is valid; otherwise, `false`. If *schemeName* is `null` or is a zero-length string, returns `false`.

Description

[*Note:* The scheme name is required to begin with a letter, and contain only letters, digits, and the characters '.', '+' or '-'.]

Uri.CheckSecurity() Method

[ILAsm]
```
.method family hidebysig virtual void CheckSecurity()
```

[C#]
```
protected virtual void CheckSecurity()
```

Summary

Checks the current instance for character sequences that can result in unauthorized access to resources, and removes them.

Behaviors

This method checks for invalid or dangerous character sequences in the components of the current instance, and removes them. The semantics that determine whether a character sequence presents a security risk are determined by the scheme of the current instance.

Default

The default implementation does nothing.

How and When to Override

Override this method to provide security checks for types derived from `System.Uri`.

Usage

Invoke this method on instances of types derived from `System.Uri` to remove any URI content that allows unauthorized access to resources.

Uri.Equals(System.Object) Method

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

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

Summary

Compares the current instance and the specified object for equality.

Parameters

Parameter	Description
comparand	The `System.Uri` instance to compare with the current instance. This argument can be a `System.String` or a `System.Uri`.

Return Value

`true` if *comparand* represents the same URI (ignoring any fragment or query information) as the current instance; otherwise, `false`. If *comparand* is `null`, a zero-length string, or is not an instance of `System.String` or `System.Uri`, returns false.

Description

If *comparand* is a `System.String`, it is converted to a `System.Uri` by calling `System.Uri`(*comparand*).

The `System.Uri.Scheme`, `System.Uri.Host` and unescaped version of the `System.Uri.AbsolutePath` of the current instance and *comparand* are compared for equality.

If the scheme of the current instance is the `System.Uri.UriSchemeFile` scheme, the absolute paths are compared in accordance with the case sensitivity of the current platform.

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

Uri.Escape() Method

```
[ILAsm]
.method family hidebysig virtual void Escape()

[C#]
protected virtual void Escape()
```

Summary

Converts any unsafe or reserved characters in the `System.Uri.AbsolutePath` component to equivalent escaped hexadecimal sequences.

Description

Behaviors

Converts any unsafe or reserved characters in the `System.Uri.AbsolutePath` component to a character sequence consisting of a "%" followed by the hexadecimal value of the character as described by IETF 2396.

If the path component of the current instance is `null`, the escaped path is `System.String.Empty`.

Default

As described above.

How and When to Override

Override this method to customize the escaping behavior provided by the `System.Uri` type.

Usage

Applications typically do not call this method; it is intended for use by the constructors.

[*Note:* For additional information on escaping URI, see section 2 of RFC 2396.]

Uri.EscapeString(System.String) Method

[ILAsm]
```
.method family hidebysig static string EscapeString(string str)
```

[C#]
```
protected static string EscapeString(string str)
```

Summary

Converts a string to its escaped representation.

Parameters

Parameter	Description
str	A `System.String` to convert to its escaped representation.

Return Value

A `System.String` containing the escaped representation of *str*.

Description

The string is escaped in accordance with RFC 2396.

Uri.FromHex(System.Char) Method

```
[ILAsm]
.method public hidebysig static int32 FromHex(valuetype System.Char
digit)

[C#]
public static int FromHex(char digit)
```

Summary

Returns the decimal value of a hexadecimal digit.

Parameters

Parameter	Description
digit	The hexadecimal digit (0-9, a-f, A-F) to convert.

Return Value

A `System.Int32` containing an integer from 0 - 15 that corresponds to the specified hexadecimal digit.

Exceptions

Exception	Condition
System.ArgumentException	*digit* is not a valid hexadecimal digit (0-9, a-f, A-F).

Uri.GetHashCode() Method

[ILAsm]

```
.method public hidebysig virtual int32 GetHashCode()
```

[C#]

```
public override int GetHashCode()
```

Summary

Generates a hash code for the current instance.

Return Value

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

Description

The hash code is generated without the fragment component. For example, the URIs "http://www.contoso.com/index.htm#search" and "http://www.contoso.com/index.htm" produce the same hash code.

The algorithm used to generate the hash code is unspecified.

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

Uri.GetLeftPart(System.UriPartial) Method

[ILAsm]
```
.method public hidebysig instance string GetLeftPart(valuetype
System.UriPartial part)
```

[C#]
```
public string GetLeftPart(UriPartial part)
```

Summary

Returns the specified portion of the URI represented by the current instance.

Parameters

Parameter	Description
part	A `System.UriPartial` value that specifies the component to return.

Return Value

A `System.String` containing all components up to the specified portion of the URI, or `System.String.Empty` if the current instance does not contain the component identified by *part*.

Description

The `System.Uri.GetLeftPart` method returns a string containing the URI components starting with the left-most component of the URI and ending with the component specified by *part*. The returned string does not include fragment or query information.

`System.Uri.GetLeftPart` includes delimiters as follows:

- `System.UriPartial.Scheme` has the scheme delimiter added.
- `System.UriPartial.Authority` does not have the path delimiter added.
- `System.UriPartial.Path` includes any delimiters in the original URI up to the query or fragment delimiter.

Exceptions

Exception	Condition

System.ArgumentException	The *part* parameter is not a valid `System.UriPartial` value.

Example

The following example demonstrates the `System.Uri.GetLeftPart` method.

[C#]

```
using System;

public class UriTest {
  public static void Main() {
    string[] myUri  = {
            "http://www.contoso.com/index.htm",
            "http:www.contoso.com/index.htm#mark",
                    "mailto:user@contoso.com?subject=uri",
                    "nntp://news.contoso.com/123456@contoso.com"
    };
    foreach (string s in myUri) {
      Uri aUri = new Uri(s);
      Console.WriteLine("URI: {0}", aUri.ToString());
      Console.WriteLine("Scheme:
{0}",aUri.GetLeftPart(UriPartial.Scheme));
      Console.WriteLine("Authority:
{0}",aUri.GetLeftPart(UriPartial.Authority));
      Console.WriteLine("Path: {0}",aUri.GetLeftPart(UriPartial.Path));
    }
  }
}
```

The output is

```
URI: http://www.contoso.com/index.htm


Scheme: http://


Authority: http://www.contoso.com


Path: http://www.contoso.com/index.htm


URI: http://www.contoso.com/index.htm#mark


Scheme: http://


Authority: http://www.contoso.com


Path: http://www.contoso.com/index.htm
```

```
URI: mailto:user@contoso.com?subject=uri

Scheme: mailto:

Authority:

Path: mailto:user@contoso.com

URI: nntp://news.contoso.com/123456@contoso.com

Scheme: nntp://

Authority: nntp://news.contoso.com

Path: nntp://news.contoso.com/123456@contoso.com
```

Uri.HexEscape(System.Char) Method

```
[ILAsm]
.method public hidebysig static string HexEscape(valuetype
System.Char character)

[C#]
public static string HexEscape(char character)
```

Summary

Converts a specified ASCII character into its escaped hexadecimal equivalent.

Parameters

Parameter	Description
character	A `System.Char` containing the character to convert to escaped hexadecimal representation.

Return Value

A `System.String` containing the escaped hexadecimal representation of the specified character.

Description

The returned string is in the form "%XX", where X represents a hexadecimal digit (0-9, A-F).

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	The numerical value of *character* is greater than 255.

Uri.HexUnescape(System.String, System.Int32&) Method

```
[ILAsm]
.method public hidebysig static valuetype System.Char
HexUnescape(string pattern, int32& index)

[C#]
public static char HexUnescape(string pattern, ref int index)
```

Summary

Converts a specified escaped hexadecimal representation of a character to the character.

Parameters

Parameter	**Description**
pattern	A `System.String` containing the hexadecimal representation of a character.
index	A `System.Int32` containing the location in *pattern* where the hexadecimal representation of a character begins.

Return Value

A `System.Char` containing a character. If the character pointed to by *index* is a "%" and there are at least two characters following the "%", and the two characters are valid hexadecimal digits, the hexadecimal digits are converted to `System.Char`. Otherwise, the character at *index* is returned. Valid hexadecimal digits are: 0-9, a-f, A-F.

On return, the value of *index* contains the index of the character following the one returned.

Exceptions

Exception	**Condition**
System.ArgumentOutOfRangeException	*index* < 0, or *index* >= the number of characters in pattern.

Uri.IsBadFileSystemCharacter(System.Char) Method

[ILAsm]
```
.method family hidebysig virtual bool
IsBadFileSystemCharacter(valuetype System.Char character)
```

[C#]
```
protected virtual bool IsBadFileSystemCharacter(char character)
```

Summary

Returns a `System.Boolean` value that indicates whether the specified character would be an invalid character if used in a file system name.

Parameters

Parameter	Description
character	A `System.Char` containing the character to check.

Return Value

`true` if the specified character is not acceptable for use in a file system name; otherwise, `false`.

The value returned by this method is implementation-specific.

Behaviors

This method returns `false` if the specified character cannot be used in a URI that identifies a file, as defined by the current file system on the current platform.

Default

As described above.

How and When to Override

Override this method to provide a check for invalid characters as defined by the current file system on the current platform.

Usage

Use this method to determine if a character can be used in a file name.

Uri.IsExcludedCharacter(System.Char) Method

```
[ILAsm]
.method family hidebysig static bool IsExcludedCharacter(valuetype
System.Char character)

[C#]
protected static bool IsExcludedCharacter(char character)
```

Summary

Returns a `System.Boolean` value that indicates whether the specified character is excluded from use or is unwise in URIs, as defined by IETF RFC 2396.

Parameters

Parameter	**Description**
character	A `System.Char` containing the character to check.

Return Value

`true` if the specified character is required to be escaped; otherwise, `false`.

Description

This method returns `true` for the following characters:

Character(s)	**Description**
character < 0x0020	Any character with the ASCII value less than hexadecimal 0x20 (32).
character < 0x007f	Any character with the ASCII value greater than hexadecimal 0x7f (127).
<	Less than sign.
>	Greater than sign.
#	Number sign (crosshatch, pound sign).
%	Percent.
"	Quotation mark.
{	Left curly brace.
}	Right curly brace.
|	Pipe sign (vertical bar).
\	Backward slash.

^	Circumflex (caret).
[	Left square bracket.
]	Right square bracket.
`	Grave accent.

Uri.IsHexDigit(System.Char) Method

[ILAsm]
```
.method public hidebysig static bool IsHexDigit(valuetype
System.Char character)
```

[C#]
```
public static bool IsHexDigit(char character)
```

Summary

Returns a `System.Boolean` value that indicates whether the specified character is a valid hexadecimal digit.

Parameters

Parameter	Description
character	A `System.Char` containing the character to validate.

Return Value

`true` if the character is a valid hexadecimal digit (0-9, A-F, a-f); otherwise `false`.

Uri.IsHexEncoding(System.String, System.Int32) Method

```
[ILAsm]
.method public hidebysig static bool IsHexEncoding(string pattern,
int32 index)

[C#]
public static bool IsHexEncoding(string pattern, int index)
```

Summary

Returns a `System.Boolean` value that indicates whether a substring of the specified string is in escaped hexadecimal encoding format ("%" followed by two hexadecimal characters).

Parameters

Parameter	Description
pattern	The `System.String` to check.
index	A `System.Int32` containing the location in *pattern* to check for hex encoding.

Return Value

`true` if the specified location in *pattern* contains a substring in escaped hexadecimal encoding format; otherwise, `false`.

Description

The `System.Uri.IsHexEncoding` method checks for hexadecimal digits case-insensitively.

Uri.IsReservedCharacter(System.Char) Method

```
[ILAsm]
.method family hidebysig virtual bool IsReservedCharacter(valuetype
System.Char character)

[C#]
protected virtual bool IsReservedCharacter(char character)
```

Summary

Returns a `System.Boolean` value that indicates whether a character is part of the URI reserved set.

Return Value

`true` if *character* is a URI reserved character as defined by IETF RFC 2396; otherwise, `false`.

Description

The following characters are reserved for the use in URI:

Character	Description
;	Semi-colon.
/	Forward slash.
:	Colon.
@	At sign (commercial at).
&	Ampersand.
=	Equals sign.
+	Plus sign.
$	US Dollar sign.
,	Comma.

Behaviors

As described above.

How and When to Override

Override this method to customize the escaping behavior provided by the `System.Uri` type.

Usage

Use this method to determine if a character is reserved.

Uri.MakeRelative(System.Uri) Method

[ILAsm]

```
.method public hidebysig instance string MakeRelative(class
System.Uri toUri)
```

[C#]

```
public string MakeRelative(Uri toUri)
```

Summary

Returns the specified `System.Uri` as a relative URI.

Parameters

Parameter	Description
toUri	The URI to compare to the current URI.

Return Value

A `System.String` with the difference between the current instance and *toUri* if the two URIs are the same except for the path information. If the two URIs differ in more than the `System.Uri.AbsolutePath`, this method returns the `System.String` representation of *toUri*.

Example

The following example demonstrates the `System.Uri.MakeRelative` method.

[C#]

```
using System;
public class UriTest {
  public static void Main() {
    Uri myUri = new Uri("http://www.contoso.com/Hello%20World.htm",
true);
    Console.WriteLine(myUri.ToString());
    Console.WriteLine(myUri.MakeRelative(new Uri
("http://www.contoso.com/index.htm")));
  }
}
```

The output is

```
http://www.contoso.com/Hello World.htm


index.htm
```

Uri.Parse() Method

[ILAsm]
```
.method family hidebysig virtual void Parse()
```

[C#]
```
protected virtual void Parse()
```

Summary

Parses the URI into its constituent components.

Behaviors

This method parses the `System.Uri.AbsolutePath` property, separates it into various URI components, and stores the components in the appropriate `System.Uri` properties.

Default

This method parses path components as defined in IETF RFC 2396.

How and When to Override

Override this method to provide parsing for URIs in formats that are not defined in IETF RFC 2396.

Usage

Applications typically do not call this method; it is intended for use by the constructors.

Exceptions

Exception	Condition
System.UriFormatException	The scheme of the URI is in an invalid format. -or-

	The URI is in an invalid form and cannot be parsed.

Uri.ToString() Method

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

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

Summary

Returns the unescaped, canonical form of the URI information used to construct the current instance.

Return Value

A `System.String` containing the unescaped, canonical form of the URI represented by the current instance.

Description

The string returned by this method includes the `System.Uri.Query` and `System.Uri.Fragment` components.

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

Uri.Unescape(System.String) Method

[ILAsm]
```
.method family hidebysig virtual string Unescape(string path)
```

[C#]
```
protected virtual string Unescape(string path)
```

Summary

Converts escape sequences in the specified `System.String` into their unescaped equivalents.

Parameters

Parameter	Description
path	The `System.String` to unescape.

Return Value

A `System.String` containing *path* with its escaped characters converted to their unescaped equivalents. If path is `null` or a zero-length string, returns `System.String.Empty`.

Description

[*Note:* Escape sequences can be hex-encoded reserved characters (for example "%40") or hex-encoded UTF-8 sequences (for example "%C4%D2").

]

Uri.AbsolutePath Property

[ILAsm]

```
.property string AbsolutePath { public hidebysig specialname
instance string get_AbsolutePath() }
```

[C#]

```
public string AbsolutePath { get; }
```

Summary

Gets the absolute path of the resource identified by the current instance.

Property Value

A `System.String` containing the absolute path to the resource.

Description

This property is read-only.

The `System.Uri.AbsolutePath` property contains the path to the resource identified by the current instance. The `System.Uri.AbsolutePath` property always returns at least a slash ('/').

If, when the current instance was constructed, the URI was already escaped or the constructor's *dontEscape* parameter was set to `false`, the value returned by this property is escaped.

[*Note:* The path information does not include the scheme, host name, query, or fragment components of the URI.]

Example

The following example outputs the absolute path of a URI.

[C#]

```
using System;

public class UriTest {
 public static void Main() {
   Uri myUri = new Uri
("http://www.contoso.com/URI/Hello%20World.htm?date=today", true);
   Console.WriteLine(myUri.AbsolutePath);
 }
}
```

The output is

```
/URI/Hello%20World.htm
```

Uri.AbsoluteUri Property

[ILAsm]
```
.property string AbsoluteUri { public hidebysig specialname instance
string get_AbsoluteUri() }
```

[C#]
```
public string AbsoluteUri { get; }
```

Summary

Gets the absolute URI of the resource identified by the current instance in canonical form.

Property Value

A `System.String` containing the URI used to construct the current instance, in canonical format.

Description

This property is read-only.

The `System.Uri.AbsoluteUri` property includes the entire URI stored in the current `System.Uri` instance, including any fragment or query information. If, when the current instance was constructed, the URI was already escaped or the constructor's *dontEscape* parameter was set to `false`, the value returned by this property is escaped.

Uri.Authority Property

[ILAsm]
```
.property string Authority { public hidebysig specialname instance
string get_Authority() }
```

[C#]
```
public string Authority { get; }
```

Summary

Gets the authority component of the URI used to construct the current instance.

Property Value

A `System.String` containing the authority component of the current instance. The value returned by this property is composed of the values returned by the `System.Uri.Host` and `System.Uri.Port` properties.

Description

This property is read-only.

The `System.Uri.Authority` property returns the `System.Uri.Host` and `System.Uri.Port` information specified in the URI used to construct the current instance. The value of this property includes the port information only if the URI specified a port that is not the default for the current scheme. When port information is included in the value returned by this property, the host and port are separated by a colon (":").

Uri.Fragment Property

[ILAsm]
`.property string Fragment { public hidebysig specialname instance string get_Fragment() }`

[C#]
`public string Fragment { get; }`

Summary

Gets the fragment component of the URI used to construct the current instance.

Property Value

A `System.String` containing any fragment information contained in the URI used to construct the current instance.

Description

This property is read-only.

The `System.Uri.Fragment` property gets any text following a fragment marker ('#') in the URI, including the fragment marker itself. If, when the current instance was constructed, the URI was already escaped or the constructor's *dontEscape* parameter was set to `false`, the value returned by this property is escaped.

[*Note:* The `System.Uri.Fragment` property is not considered in a `System.Uri.Equals` comparison.

]

Example

The following example demonstrates the use of the `System.Uri.Fragment` property.

[C#]

```
using System;

public class UriTest {
 public static void Main() {

 Uri baseUri = new Uri("http://www.contoso.com/");
 Uri myUri = new Uri(baseUri, "index.htm#main");

 Console.WriteLine(myUri.Fragment);
 }
}
```

The output is

```
#main
```

Uri.Host Property

[ILAsm]

```
.property string Host { public hidebysig specialname instance string get_Host() }
```

[C#]

```
public string Host { get; }
```

Summary

Gets the host component of the URI used to construct the current instance.

Property Value

A `System.String` containing the DNS host name or IP address of the host server. If the host information was not specified to the constructor, the value of this property is `System.String.Empty`.

Description

This property is read-only.

If the host information is an IP6 address, the information is enclosed in square brackets ("[" and "]").

Example

The following example demonstrates using the `System.Uri.Host` property.

[C#]

```
using System;

public class UriTest {
 public static void Main() {

 Uri baseUri = new Uri("http://www.contoso.com:8080/");
 Uri myUri = new Uri(baseUri, "shownew.htm?date=today");

 Console.WriteLine(myUri.Host);
 }
}
```

The output is

```
www.contoso.com
```

Uri.HostNameType Property

[ILAsm]

```
.property valuetype System.UriHostNameType HostNameType { public
hidebysig specialname instance valuetype System.UriHostNameType
get_HostNameType() }
```

[C#]

```
public UriHostNameType HostNameType { get; }
```

Summary

Gets the format of the host address in the URI used to construct the current instance.

Property Value

A `System.UriHostNameType` that indicates the format of the host address information in the current instance.

Description

This property is read-only.

If `System.Uri.Host` is `null`, the value of this property is `System.UriHostNameType.Unknown`.

Uri.IsDefaultPort Property

[ILAsm]
```
.property bool IsDefaultPort { public hidebysig specialname instance
bool get_IsDefaultPort() }
```

[C#]
```
public bool IsDefaultPort { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the `System.Uri.Port` value of the current instance is the default port for the scheme of the current instance.

Property Value

`true` if the value in the `System.Uri.Port` property is the default port for the `System.Uri.Scheme`; otherwise, `false`.

Description

This property is read-only.

[*Note:* For a list of default port values, see the `System.Uri.Port` property.]

Uri.IsFile Property

```
[ILAsm]
.property bool IsFile { public hidebysig specialname instance bool
get_IsFile() }

[C#]
public bool IsFile { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the current instance identifies a file.

Property Value

`true` if the resource identified by the current `System.Uri` is a file; otherwise, `false`.

Description

This property is read-only.

The `System.Uri.IsFile` property is `true` when the `System.Uri.Scheme` property equals `System.Uri.UriSchemeFile`.

Uri.IsLoopback Property

[ILAsm]

```
.property bool IsLoopback { public hidebysig specialname instance
bool get_IsLoopback() }
```

[C#]

```
public bool IsLoopback { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the host information of the current instance is the current computer.

Property Value

`true` if the host of the current instance is the reserved hostname "localhost" or the loop-back IP address (127.0.0.1); otherwise, `false`.

Description

This property is read-only.

Example

The following example demonstrates the `System.Uri.IsLoopback` property.

[C#]

```
using System;

public class UriTest {
 public static void Main() {
 Uri myUri = new Uri("http://127.0.0.1/index.htm", true);
 Console.WriteLine("{0} is loopback? {1}", myUri.ToString(),
myUri.IsLoopback);

 myUri = new Uri("http://localhost/index.htm", true);
 Console.WriteLine("{0} is loopback? {1}", myUri.ToString(),
myUri.IsLoopback);

 }
}
```

The output is

```
http://127.0.0.1/index.htm is loopback? True

http://localhost/index.htm is loopback? True
```

Uri.LocalPath Property

[ILAsm]
```
.property string LocalPath { public hidebysig specialname instance
string get_LocalPath() }
```

[C#]
```
public string LocalPath { get; }
```

Summary

Gets the local operating-system representation of the resource identified by the current instance.

Property Value

A `System.String` containing the local representation of the resource identified by the current instance.

Description

This property is read-only.

If the `System.Uri.Scheme` of the current instance is not equal to `System.Uri.UriSchemeFile`, this property returns the same value as `System.Uri.AbsolutePath`.

If the scheme is equal to `System.Uri.UriSchemeFile`, this property returns an unescaped platform-dependent local representation of the file name.

Uri.PathAndQuery Property

[ILAsm]

```
.property string PathAndQuery { public hidebysig specialname
instance string get_PathAndQuery() }
```

[C#]

```
public string PathAndQuery { get; }
```

Summary

Gets the `System.Uri.AbsolutePath` and `System.Uri.Query` components of the URI used to construct the current instance.

Property Value

A `System.String` that contains the values of the `System.Uri.AbsolutePath` and `System.Uri.Query` properties.

Description

This property is read-only.

Example

The following example uses the `System.Uri.PathAndQuery` property to extract the path and query information from a `System.Uri` instance.

[C#]

```
using System;

public class UriTest {
 public static void Main() {

 Uri baseUri = new Uri("http://www.contoso.com/");
 Uri myUri = new Uri(baseUri, "catalog/shownew.htm?date=today");

 Console.WriteLine(myUri.PathAndQuery);
 }
}
```

The output is

```
/catalog/shownew.htm?date=today
```

Uri.Port Property

[ILAsm]
```
.property int32 Port { public hidebysig specialname instance int32
get_Port() }
```

[C#]
```
public int Port { get; }
```

Summary

Gets the port number used to connect to the `System.Uri.Host` referenced by the current instance.

Property Value

A `System.Int32` containing the port number, or -1 if no port is used by the URI `System.Uri.Scheme`. If no port was specified as part of the URI used to construct the current instance, the `System.Uri.Port` property returns the default port for the URI scheme.

Description

This property is read-only.

[*Note:* The following table lists the default port number for each supported scheme.

Scheme	**Port**
file	-1
ftp	21
gopher	70
http	80
https	43
mailto	25
news	119
nntp	119

]

Uri.Query Property

[ILAsm]
```
.property string Query { public hidebysig specialname instance
string get_Query() }
```

[C#]
```
public string Query { get; }
```

Summary

Gets the query component of the URI used to construct the current instance.

Property Value

A `System.String` containing the query information included in the specified URI, or `System.String.Empty`.

Description

This property is read-only.

If, when the current instance was constructed, the URI was already escaped or the constructor's *dontEscape* parameter was set to `false`, the value returned by this property is escaped.

[*Note:* Query information is separated from the path information by a question mark ('?') and is located at the end of a URI. The query information includes the leading question mark.]

Example

The following example uses the `System.Uri.Query` property to extract the query from a URI.

[C#]

```
using System;

public class UriTest {
 public static void Main() {

 Uri baseUri = new Uri("http://www.contoso.com/");
 Uri myUri = new Uri(baseUri, "catalog/shownew.htm?date=today");

 Console.WriteLine(myUri.Query);
 }
}
```

The output is

```
?date=today
```

Uri.Scheme Property

[ILAsm]
```
.property string Scheme { public hidebysig specialname instance
string get_Scheme() }
```

[C#]
```
public string Scheme { get; }
```

Summary

Gets the scheme component of the URI used to construct the current instance.

Property Value

A `System.String` containing the URI scheme.

Description

This property is read-only.

Uri.UserEscaped Property

[ILAsm]
```
.property bool UserEscaped { public hidebysig specialname instance
bool get_UserEscaped() }
```

[C#]
```
public bool UserEscaped { get; }
```

Summary

Gets a `System.Boolean` value that indicates whether the URI information used to construct the current instance was escaped before the current instance was created.

Property Value

`true` if the *dontEscape* parameter of the constructor for the current instance was set to `true;` otherwise, `false`.

Description

This property is read-only.

Uri.UserInfo Property

[ILAsm]

```
.property string UserInfo { public hidebysig specialname instance
string get_UserInfo() }
```

[C#]

```
public string UserInfo { get; }
```

Summary

Gets the userinfo component of the URI used to construct the current instance.

Property Value

A `System.String` containing any user information included in the URI used to construct the current instance, or `System.String.Empty` if no user information was included.

Description

This property is read-only.

[*Note:* For details on the userinfo component of a URI, see IETF RFC 2396, 3.2.2.]