

System.Security Namespace

Reference

Provides the underlying structure of the common language runtime security system, including base classes for permissions.

Classes

[] Expand table

AllowPartiallyTrustedCallersAttribute	Allows an assembly to be called by partially trusted code. Without this declaration, only fully trusted callers are able to use the assembly. This class cannot be inherited.
PermissionSet	Represents a collection that can contain many different types of permissions.
SecureString	Represents text that should be kept confidential, such as by deleting it from computer memory when no longer needed. This class cannot be inherited.
SecureStringMarshal	Provides a collection of methods for allocating unmanaged memory and copying unmanaged memory blocks.
SecurityCriticalAttribute	Specifies that code or an assembly performs security-critical operations.
SecurityElement	Represents the XML object model for encoding security objects. This class cannot be inherited.
SecurityException	The exception that is thrown when a security error is detected.
SecurityRulesAttribute	Indicates the set of security rules the common language runtime should enforce for an assembly.
SecuritySafeCriticalAttribute	Identifies types or members as security-critical and safely accessible by transparent code.
SecurityTransparentAttribute	Specifies that an assembly cannot cause an elevation of privilege.
SecurityTreatAsSafeAttribute	Identifies which of the nonpublic SecurityCriticalAttribute members are accessible by transparent code within the assembly.
SuppressUnmanagedCodeSecurityAttribute	Allows managed code to call into unmanaged code without a stack walk. This class cannot be inherited.

UnverifiableCode Attribute	Marks modules containing unverifiable code. This class cannot be inherited.
VerificationException	The exception that is thrown when the security policy requires code to be type safe and the verification process is unable to verify that the code is type safe.

Interfaces

[\[+\] Expand table](#)

IPermission	Defines methods implemented by permission types.
ISecurityEncodable	Defines the methods that convert permission object state to and from XML element representation.
IStackWalk	Manages the stack walk that determines whether all callers in the call stack have the required permissions to access a protected resource.

Enums

[\[+\] Expand table](#)

PartialTrustVisibilityLevel	Specifies the default partial-trust visibility for code that is marked with the AllowPartiallyTrustedCallersAttribute (APTCA) attribute.
SecurityCriticalScope	Specifies the scope of a SecurityCriticalAttribute .
SecurityRuleSet	Identifies the set of security rules the common language runtime should enforce for an assembly.

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

AllowPartiallyTrustedCallersAttribute Class

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Allows an assembly to be called by partially trusted code. Without this declaration, only fully trusted callers are able to use the assembly. This class cannot be inherited.

C#

```
[System.AttributeUsage(System.AttributeTargets.Assembly,
    AllowMultiple=false, Inherited=false)]
public sealed class AllowPartiallyTrustedCallersAttribute : Attribute
```

Inheritance Object → [Attribute](#) → AllowPartiallyTrustedCallersAttribute

Attributes [AttributeUsageAttribute](#)

Examples

The following example shows how to use the [AllowPartiallyTrustedCallersAttribute](#) class.

C#

```
// The following HTML code can be used to call the user control in this
sample.
//
//      <OBJECT id="usercontrol"
//      classid="usercontrol.dll#UserControl.UserControl1" width="800"
//      height="300" style="font-size:12;">

// To run this test control you must create a strong name key, snkey.snk,
and
// a code group that gives full trust to assemblies signed with snkey.snk.

// The user control displays an OpenFileDialog box, then displays a text box
containing the name of
// the file selected and a list box that displays the contents of the file.
The selected file must
```

```
// contain text in order for the control to display the data properly.

// Caution This sample demonstrates the use of the Assert method. Calling
Assert removes the
// requirement that all code in the call chain must be granted permission to
access the specified
// resource, it can open up security vulnerabilities if used incorrectly or
inappropriately. Therefore,
// it should be used with great caution. Assert should always be followed
with a RevertAssert
// command to restore the security settings.

using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Windows.Forms;
using System.IO;
using System.Security;
using System.Security.Permissions;
using System.Reflection;
using System.Runtime.CompilerServices;

// This strong name key is used to create a code group that gives
permissions to this assembly.
[assembly: AssemblyKeyFile("snKey.snk")]
[assembly: AssemblyVersion("1.0.0.0")]

// The AllowPartiallyTrustedCallersAttribute requires the assembly to be
signed with a strong name key.
// This attribute is necessary since the control is called by either an
intranet or Internet
// Web page that should be running under restricted permissions.
[assembly:AllowPartiallyTrustedCallers]
namespace UserControl
{
    // The userControl1 displays an OpenFileDialog box, then displays a text
    box containing the name of
    // the file selected and a list box that displays the contents of the
    file. The selected file must
    // contain text in order for the control to display the data properly.
    public class UserControl1 : System.Windows.Forms.UserControl
    {
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.ListBox listBox1;
        // Required designer variable.
        private System.ComponentModel.Container components = null;

        // Demand the zone requirement for the calling application.
        [ZoneIdentityPermission(SecurityAction.Demand, Zone =
SecurityZone.Intranet)]
        public UserControl1()
        {
            // This call is required by the Windows.Forms Form Designer.
```

```
InitializeComponent();

        // The OpenFileDialog box should not require any special
permissions.
        OpenFileDialog fileDialog = new OpenFileDialog();
        if(fileDialog.ShowDialog() == DialogResult.OK)
        {
            // Reading the name of the selected file from the
OpenFileDialog box
                // and reading the file requires FileIOPermission. The user
control should
                    // have this permission granted through its code group; the
Web page that calls the
                        // control should not have this permission. The Assert
command prevents a stack walk
                            // that would fail because the caller does not have the
required FileIOPermission.
                                // The use of Assert can open up security vulnerabilities if
used incorrectly or
                                    // inappropriately. Therefore, it should be used with great
caution.
                                        // The Assert command should be followed by a RevertAssert
as soon as the file operation
                                            // is completed.
                                            new FileIOPermission(PermissionState.Unrestricted).Assert();
                                            textBox1.Text = fileDialog.FileName;
                                            // Display the contents of the file in the text box.
                                            FileStream fsIn = new FileStream(textBox1.Text,
 FileMode.Open, FileAccess.Read,
                                             FileShare.Read);
                                            StreamReader sr = new StreamReader(fsIn);

                                                // Process every line in the file
                                                for (String Line = sr.ReadLine(); Line != null; Line =
sr.ReadLine())
                                                {
                                                    listBox1.Items.Add(Line);
                                                }
                                                // It is very important to call RevertAssert to restore the
stack walk for
                                                    // file operations.
                                                    FileIOPermission.RevertAssert();
                                                }
                                            }

// Clean up any resources being used.
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if( components != null )
            components.Dispose();
    }
    base.Dispose( disposing );
}
```

```

#region Component Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.textBox1 = new System.Windows.Forms.TextBox();
    this.listBox1 = new System.Windows.Forms.ListBox();
    this.SuspendLayout();
    //
    // textBox1
    //
    this.textBox1.Location = new System.Drawing.Point(208, 112);
    this.textBox1.Name = "textBox1";
    this.textBox1.Size = new System.Drawing.Size(320, 20);
    this.textBox1.TabIndex = 0;
    this.textBox1.Text = "textBox1";
    this.textBox1.TextChanged += new
System.EventHandler(this.textBox1_TextChanged);
    //
    // listBox1
    //
    this.listBox1.Location = new System.Drawing.Point(200, 184);
    this.listBox1.Name = "listBox1";
    this.listBox1.Size = new System.Drawing.Size(336, 108);
    this.listBox1.TabIndex = 1;
    //
    // UserControl1
    //
    this.Controls.Add(this.listBox1);
    this.Controls.Add(this.textBox1);
    this.Name = "UserControl1";
    this.Size = new System.Drawing.Size(592, 400);
    this.Load += new System.EventHandler(this.UserControl1_Load);
    this.ResumeLayout(false);
}
#endregion

private void UserControl1_Load(object sender, System.EventArgs e)
{
}

private void textBox1_TextChanged(object sender, System.EventArgs e)
{
}
}

```

Remarks

ⓘ Important

Partially trusted code is no longer supported. This attribute has no effect in .NET Core.

ⓘ Note

The .NET Framework 4 introduces new security rules that affect the behavior of the [AllowPartiallyTrustedCallersAttribute](#) attribute (see [Security-Transparent Code, Level 2](#)). In the .NET Framework 4, all code defaults to security-transparent, that is, partially trusted. However, you can annotate individual types and members to assign them other transparency attributes. For this and other security changes, see [Security Changes](#).

.NET Framework version 2.0 () assemblies must be strong-named to effectively use the [AllowPartiallyTrustedCallersAttribute](#) (APCA) attribute. .NET Framework 4 () assemblies do not have to be strong-named for the APTCA attribute to be effective, and they can contain transparent, security-critical and security-safe-critical code. For more information about applying attributes at the assembly level, see [Applying Attributes](#).

By default, if a strong-named, assembly does not explicitly apply this attribute at the assembly level, it can be called only by other assemblies that are granted full trust. This restriction is enforced by placing a [LinkDemand](#) for [FullTrust](#) on every public or protected method on every publicly accessible class in the assembly. Assemblies that are intended to be called by partially trusted code can declare their intent through the use of [AllowPartiallyTrustedCallersAttribute](#). An example of the declaration in C# is

```
[assembly:AllowPartiallyTrustedCallers]; an example in Visual Basic is
```

```
<assembly:AllowPartiallyTrustedCallers>.
```

✖ Caution

The presence of this assembly-level attribute prevents the default behavior of placing [FullTrust](#) [LinkDemand](#) security checks, and makes the assembly callable from any other (partially or fully trusted) assembly.

When the APTCA attribute is present, all other security checks function as intended, including any class-level or method-level declarative security attributes that are present. This attribute blocks only the implicit, fully trusted caller demand.

This is not a declarative security attribute, but a regular attribute (it derives from [System.Attribute](#), not from [System.Security.Permissions.SecurityAttribute](#)).

For more information, see [Using Libraries from Partially Trusted Code](#).

Constructors

[+] Expand table

AllowPartiallyTrustedCallersAttribute()	Initializes a new instance of the AllowPartiallyTrustedCallersAttribute class.
---	--

Properties

[+] Expand table

PartialTrustVisibilityLevel	Gets or sets the default partial trust visibility for code that is marked with the AllowPartiallyTrustedCallersAttribute (APTCA) attribute.
TypeId	When implemented in a derived class, gets a unique identifier for this Attribute . (Inherited from Attribute)

Methods

[+] Expand table

Equals(Object)	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute)
GetHashCode()	Returns the hash code for this instance. (Inherited from Attribute)
GetType()	Gets the Type of the current instance. (Inherited from Object)
IsDefaultAttribute()	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute)
Match(Object)	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object.

(Inherited from [Attribute](#))

MemberwiseClone()	Creates a shallow copy of the current Object . (Inherited from Object)
ToString()	Returns a string that represents the current object. (Inherited from Object)

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0

 Collaborate with us on
[GitHub](#)

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



[.NET feedback](#)

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

AllowPartiallyTrustedCallersAttribute Constructor

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Initializes a new instance of the [AllowPartiallyTrustedCallersAttribute](#) class.

C#

```
public AllowPartiallyTrustedCallersAttribute ();
```

Examples

For an example of how to use this constructor, see the code example provided for the [AllowPartiallyTrustedCallersAttribute](#) class.

Remarks

This attribute should be applied only at the assembly level.

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

AllowPartiallyTrustedCallersAttribute. PartialTrustVisibilityLevel Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets or sets the default partial trust visibility for code that is marked with the [AllowPartiallyTrustedCallersAttribute](#) (APTCA) attribute.

C#

```
public System.Security.PartialTrustVisibilityLevel  
PartialTrustVisibilityLevel { get; set; }
```

Property Value

[PartialTrustVisibilityLevel](#)

One of the enumeration values. The default is [VisibleToAllHosts](#).

Remarks

The following examples demonstrate how to use this property.

- Default, unconditional APTCA:

```
[assembly: AllowPartiallyTrustedCallers]
```

Defaults to [VisibleToAllHosts](#).

- Explicit, unconditional APTCA:

```
[assembly:  
AllowPartiallyTrustedCallers(PartialTrustVisibilityLevel=VisibleToAllHo
```

```
sts)]
```

The assembly can always be called by partial-trust code.

- Explicit, conditional APTCA:

```
[assembly:  
AllowPartiallyTrustedCallers(PartialTrustVisibilityLevel=NotVisibleByDefault)]
```

The assembly has been audited for partial trust, but it is not visible to partial-trust code by default. To make the assembly visible to partial-trust code, add it to the [AppDomainSetup.PartialTrustVisibleAssemblies](#) property.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

IPermission Interface

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

⊗ Caution

Code Access Security is not supported or honored by the runtime.

Defines methods implemented by permission types.

C#

```
[System.Obsolete("Code Access Security is not supported or honored by the  
runtime.", DiagnosticId="SYSLIB0003", UrlFormat="https://aka.ms/dotnet-  
warnings/{0}")]  
public interface IPermission : System.Security.ISecurityEncodable
```

Derived [System.IdentityModel.Services.ClaimsPrincipalPermission](#)

[System.Security.CodeAccessPermission](#)

[System.Security.Permissions.PrincipalPermission](#)

Attributes [ObsoleteAttribute](#)

Implements [ISecurityEncodable](#)

Examples

This example shows how to define a permission class for use with code access security. All of the necessary permission interfaces are implemented.

C#

```
using System;  
using System.Security;  
using System.Security.Permissions;
```

```

using System.Reflection;

// Enumerated type for permission states.
[Serializable]
public enum SoundPermissionState
{
    NoSound = 0,
    PlaySystemSounds = 1,
    PlayAnySound = 2
}

// Derive from CodeAccessPermission to gain implementations of the following
// sealed IStackWalk methods: Assert, Demand, Deny, and PermitOnly.
// Implement the following abstract IPermission methods: Copy, Intersect,
and IsSubSetOf.
// Implementing the Union method of the IPermission class is optional.
// Implement the following abstract ISecurityEncodable methods: FromXml and
ToXml.
// Making the class 'sealed' is optional.

public sealed class SoundPermission : CodeAccessPermission, IPermission,
    IUnrestrictedPermission, ISecurityEncodable, ICloneable
{
    private Boolean m_specifiedAsUnrestricted = false;
    private SoundPermissionState m_flags = SoundPermissionState.NoSound;

    // This constructor creates and initializes a permission with generic
    access.
    public SoundPermission(PermissionState state)
    {
        m_specifiedAsUnrestricted = (state == PermissionState.Unrestricted);
    }

    // This constructor creates and initializes a permission with specific
    access.
    public SoundPermission(SoundPermissionState flags)
    {
        if (!Enum.IsDefined(typeof(SoundPermissionState), flags))
            throw new ArgumentException
                ("flags value is not valid for the SoundPermissionState
enumrated type");
        m_specifiedAsUnrestricted = false;
        m_flags = flags;
    }

    // For debugging, return the state of this object as XML.
    public override String ToString() { return ToXml().ToString(); }

    // Private method to cast (if possible) an IPermission to the type.
    private SoundPermission VerifyTypeMatch(IPermission target)
    {
        if (GetType() != target.GetType())
            throw new ArgumentException(String.Format("target must be of the
{0} type",
                GetType().FullName));
    }
}

```

```

        return (SoundPermission)target;
    }

    // This is the Private Clone helper method.
    private SoundPermission Clone(Boolean specifiedAsUnrestricted,
SoundPermissionState flags)
{
    SoundPermission soundPerm = (SoundPermission)Clone();
    soundPerm.m_specifiedAsUnrestricted = specifiedAsUnrestricted;
    soundPerm.m_flags = specifiedAsUnrestricted ?
SoundPermissionState.PlayAnySound : m_flags;
    return soundPerm;
}

#region IPermission Members
// Return a new object that contains the intersection of 'this' and
'target'.
public override IPermission Intersect(IPermision target)
{
    // If 'target' is null, return null.
    if (target == null) return null;

    // Both objects must be the same type.
    SoundPermission soundPerm = VerifyTypeMatch(target);

    // If 'this' and 'target' are unrestricted, return a new
unrestricted permission.
    if (m_specifiedAsUnrestricted &&
soundPerm.m_specifiedAsUnrestricted)
        return Clone(true, SoundPermissionState.PlayAnySound);

    // Calculate the intersected permissions. If there are none, return
null.
    SoundPermissionState val = (SoundPermissionState)
        Math.Min((Int32)m_flags, (Int32)soundPerm.m_flags);
    if (val == 0) return null;

    // Return a new object with the intersected permission value.
    return Clone(false, val);
}

// Called by the Demand method: returns true if 'this' is a subset of
'target'.
public override Boolean IsSubsetOf(IPermision target)
{
    // If 'target' is null and this permission allows nothing, return
true.
    if (target == null) return m_flags == 0;

    // Both objects must be the same type.
    SoundPermission soundPerm = VerifyTypeMatch(target);

    // Return true if the permissions of 'this' is a subset of 'target'.
    return m_flags <= soundPerm.m_flags;
}

```

```

// Return a new object that matches 'this' object's permissions.
public sealed override IPermission Copy()
{
    return (IPermission)Clone();
}

// Return a new object that contains the union of 'this' and 'target'.
// Note: You do not have to implement this method. If you do not, the
version
// in CodeAccessPermission does this:
//   1. If target is not null, a NotSupportedException is thrown.
//   2. If target is null, then Copy is called and the new object is
returned.
public override IPermission Union(IPermission target)
{
    // If 'target' is null, then return a copy of 'this'.
    if (target == null) return Copy();

    // Both objects must be the same type.
    SoundPermission soundPerm = VerifyTypeMatch(target);

    // If 'this' or 'target' are unrestricted, return a new unrestricted
permission.
    if (m_specifiedAsUnrestricted ||
soundPerm.m_specifiedAsUnrestricted)
        return Clone(true, SoundPermissionState.PlayAnySound);

    // Return a new object with the calculated, unioned permission
value.
    return Clone(false, (SoundPermissionState)
        Math.Max((Int32)m_flags, (Int32)soundPerm.m_flags));
}

#endregion

#region ISecurityEncodable Members
// Populate the permission's fields from XML.
public override void FromXml(SecurityElement e)
{
    m_specifiedAsUnrestricted = false;
    m_flags = 0;

    // If XML indicates an unrestricted permission, make this permission
unrestricted.
    String s = (String)e.Attributes["Unrestricted"];
    if (s != null)
    {
        m_specifiedAsUnrestricted = Convert.ToBoolean(s);
        if (m_specifiedAsUnrestricted)
            m_flags = SoundPermissionState.PlayAnySound;
    }

    // If XML indicates a restricted permission, parse the flags.
    if (!m_specifiedAsUnrestricted)
    {

```

```

        s = (String)e.Attributes["Flags"];
        if (s != null)
        {
            m_flags = (SoundPermissionState)
                Convert.ToInt32(Enum.Parse(typeof(SoundPermission), s,
true));
        }
    }

    // Produce XML from the permission's fields.
    public override SecurityElement ToXml()
    {
        // These first three lines create an element with the required
format.
        SecurityElement e = new SecurityElement("IPermission");
        // Replace the double quotation marks ("") with single quotation
marks ('')
        // to remain XML compliant when the culture is not neutral.
        e.AddAttribute("class",
GetType().AssemblyQualifiedName.Replace('\"', '\"'));
        e.AddAttribute("version", "1");

        if (!m_specifiedAsUnrestricted)
            e.AddAttribute("Flags",
Enum.Format(typeof(SoundPermissionState), m_flags, "G"));
        else
            e.AddAttribute("Unrestricted", "true");
        return e;
    }
    #endregion

    #region IUnrestrictedPermission Members
    // Returns true if permission is effectively unrestricted.
    public Boolean IsUnrestricted()
    {
        // This means that the object is unrestricted at runtime.
        return m_flags == SoundPermissionState.PlayAnySound;
    }
    #endregion

    #region ICloneable Members

    // Return a copy of the permission.
    public Object Clone() { return MemberwiseClone(); }

    #endregion
}

```

Remarks

Caution

Code Access Security (CAS) has been deprecated across all versions of .NET Framework and .NET. Recent versions of .NET do not honor CAS annotations and produce errors if CAS-related APIs are used. Developers should seek alternative means of accomplishing security tasks.

Permissions in the common language runtime are objects that describe sets of operations that can be secured for specified resources. A permission object describes operations or access that is subject to security control; it does not represent access or a right to perform operations. Permissions are used by both application code and the .NET Framework security system in the following ways:

- Code requests the permissions it needs in order to run.
- The security system policy grants permissions to code in order for it to run.
- Code demands that calling code has a permission.
- Code overrides the security stack using assert/deny/permit-only.

Note

If you write a new permission, you must implement this interface in your class.

Important

A permission can be accessed by multiple threads. When implementing this interface, you must guarantee that the [IsSubsetOf](#), [Intersect](#), [Union](#), and [Copy](#) method implementations are thread safe.

Methods

 Expand table

Copy()	Creates and returns an identical copy of the current permission.
Demand()	Throws a SecurityException at run time if the security requirement is not met.

FromXml(Security Element)	Reconstructs a security object with a specified state from an XML encoding. (Inherited from ISecurityEncodable)
Intersect(IPermission)	Creates and returns a permission that is the intersection of the current permission and the specified permission.
IsSubsetOf(IPermission)	Determines whether the current permission is a subset of the specified permission.
ToXml()	Creates an XML encoding of the security object and its current state. (Inherited from ISecurityEncodable)
Union(IPermission)	Creates a permission that is the union of the current permission and the specified permission.

Applies to

Product	Versions (<i>Obsolete</i>)
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1 (5, 6, 7, 8, 9)
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	(5, 6, 7, 8, 9)
.NET Standard	2.0, 2.1
Windows Desktop	(5, 6, 7, 8, 9)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

IPermission.Copy Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Creates and returns an identical copy of the current permission.

C#

```
public System.Security.IPermission Copy();
```

Returns

[IPermission](#)

A copy of the current permission.

Examples

The following code example demonstrates implementing the [Copy](#) method. This code example is part of a larger example provided for the [IPermission](#) interface.

C#

```
// Return a new object that matches 'this' object's permissions.
public sealed override IPermission Copy()
{
    return (IPermission)Clone();
}
```

Remarks

A copy of a permission represents the same access to resources as the original permission.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

IPermission.Demand Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Throws a [SecurityException](#) at run time if the security requirement is not met.

C#

```
public void Demand();
```

Remarks

This method is typically used by secure libraries to ensure that callers have permission to access a resource. For example, a file class in a secure class library calls [Demand](#) for the necessary [FileIOPermission](#) before performing a file operation requested by the caller.

Although the majority of the classes that implement this interface method satisfy the security criteria by performing a full stack walk, a stack walk is not necessarily performed. An example of an implementation that does not perform a stack walk is [PrincipalPermission.Demand](#).

When a stack walk is performed, the permissions of the code that calls this method are not examined; the check begins from the immediate caller of that code and proceeds up the stack. The call stack is typically represented as growing down, so that methods higher in the call stack call methods lower in the call stack. [Demand](#) succeeds only if no [SecurityException](#) is raised.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

IPermission.Intersect(IPermission) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Creates and returns a permission that is the intersection of the current permission and the specified permission.

C#

```
public System.Security.IPermission? Intersect (System.Security.IPermission?  
target);
```

Parameters

target [IPermission](#)

A permission to intersect with the current permission. It must be of the same type as the current permission.

Returns

[IPermission](#)

A new permission that represents the intersection of the current permission and the specified permission. This new permission is `null` if the intersection is empty.

Exceptions

[ArgumentException](#)

The `target` parameter is not `null` and is not an instance of the same class as the current permission.

Examples

The following code example demonstrates implementing the [Intersect](#) method. This code example is part of a larger example provided for the [IPermission](#) class.

C#

```
// Return a new object that contains the intersection of 'this' and
// 'target'.
public override IPermission Intersect(IPermission target)
{
    // If 'target' is null, return null.
    if (target == null) return null;

    // Both objects must be the same type.
    SoundPermission soundPerm = VerifyTypeMatch(target);

    // If 'this' and 'target' are unrestricted, return a new unrestricted
    permission.
    if (m_specifiedAsUnrestricted && soundPerm.m_specifiedAsUnrestricted)
        return Clone(true, SoundPermissionState.PlayAnySound);

    // Calculate the intersected permissions. If there are none, return
    null.
    SoundPermissionState val = (SoundPermissionState)
        Math.Min((Int32)m_flags, (Int32)soundPerm.m_flags);
    if (val == 0) return null;

    // Return a new object with the intersected permission value.
    return Clone(false, val);
}
```

Remarks

The intersection of two permissions is a permission that describes the set of operations they both describe in common. Only a demand that passes both original permissions will pass the intersection.

The following statements are required to be true for all implementations of the [Intersect](#) method. `x` and `y` represent [IPermission](#) object references that are not `null`.

- `x.Intersect(x)` returns a value equal to `x`.
- `x.Intersect(y)` returns the same value as `y.Intersect(x)`.
- `x.Intersect(null)` returns `null`.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

IPermission.IsSubsetOf(IPermission) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Determines whether the current permission is a subset of the specified permission.

C#

```
public bool IsSubsetOf (System.Security.IPermission? target);
```

Parameters

target [IPermission](#)

A permission that is to be tested for the subset relationship. This permission must be of the same type as the current permission.

Returns

[Boolean](#)

`true` if the current permission is a subset of the specified permission; otherwise, `false`.

Exceptions

[ArgumentException](#)

The `target` parameter is not `null` and is not of the same type as the current permission.

Examples

The following code example demonstrates implementing the [IsSubsetOf](#) method. This code example is part of a larger example provided for the [IPermission](#) class.

C#

```

// Called by the Demand method: returns true if 'this' is a subset of
// 'target'.
public override Boolean IsSubsetOf(IPermission target)
{
    // If 'target' is null and this permission allows nothing, return true.
    if (target == null) return m_flags == 0;

    // Both objects must be the same type.
    SoundPermission soundPerm = VerifyTypeMatch(target);

    // Return true if the permissions of 'this' is a subset of 'target'.
    return m_flags <= soundPerm.m_flags;
}

```

Remarks

The current permission is a subset of the specified permission if the current permission specifies a set of operations that is wholly contained by the specified permission. For example, a permission that represents access to C:\example.txt is a subset of a permission that represents access to C:\. If this method returns `true`, the current permission represents no more access to the protected resource than does the specified permission.

The following statements are required to be true for all implementations of the `IsSubsetOf` method. `x`, `y`, and `z` represent `IPermission` objects that are not `null`.

- `x.IsSubsetOf(x)` returns `true`.
- `x.IsSubsetOf(y)` returns the same value as `y.IsSubsetOf(x)` if and only if `x` and `y` represent the same set of permissions.
- If `x.IsSubsetOf(y)` and `y.IsSubsetOf(z)` both return `true`, `x.IsSubsetOf(z)` returns `true`.

If `x` represents an empty `IPermission` object with a permission state of `None` and `y` represents an `IPermission` object that is `null`, `x.IsSubsetOf(y)` returns `true`. If `z` is also an empty permission, the compound set operation `x.Union(Z).IsSubsetOf(Y)` also returns `true` because the union of two empty permissions is an empty permission.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

IPermission.Union(IPermission) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Creates a permission that is the union of the current permission and the specified permission.

C#

```
public System.Security.IPermission? Union (System.Security.IPermission?  
target);
```

Parameters

target [IPermission](#)

A permission to combine with the current permission. It must be of the same type as the current permission.

Returns

[IPermission](#)

A new permission that represents the union of the current permission and the specified permission.

Exceptions

[ArgumentException](#)

The `target` parameter is not `null` and is not of the same type as the current permission.

Examples

The following code example demonstrates implementing the `Union` method. This code example is part of a larger example provided for the [IPermission](#) class.

C#

```

// Return a new object that contains the union of 'this' and 'target'.
// Note: You do not have to implement this method. If you do not, the
version
// in CodeAccessPermission does this:
//    1. If target is not null, a NotSupportedException is thrown.
//    2. If target is null, then Copy is called and the new object is
returned.
public override IPermission Union(IPermission target)
{
    // If 'target' is null, then return a copy of 'this'.
    if (target == null) return Copy();

    // Both objects must be the same type.
    SoundPermission soundPerm = VerifyTypeMatch(target);

    // If 'this' or 'target' are unrestricted, return a new unrestricted
permission.
    if (m_specifiedAsUnrestricted || soundPerm.m_specifiedAsUnrestricted)
        return Clone(true, SoundPermissionState.PlayAnySound);

    // Return a new object with the calculated, unioned permission value.
    return Clone(false, (SoundPermissionState)
        Math.Max((Int32)m_flags, (Int32)soundPerm.m_flags));
}

```

Remarks

The result of a call to [Union](#) is a permission that represents all the operations represented by both the current permission and the specified permission. Any demand that passes either permission passes their union.

The following statements are required to be true for all implementations of the [Union](#) method. `X` and `Y` represent [IPermission](#) objects that are not `null`.

- `X.Union(X)` returns an object that has the same value as `X`.
- `X.Union(Y)` returns an object that has the same value as the object returned by `Y.Union(X)`.
- `X.Union(null)` returns an object that has the same value as `X`.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9

Product	Versions
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

ISecurityEncodable Interface

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Defines the methods that convert permission object state to and from XML element representation.

C#

```
public interface ISecurityEncodable
```

Derived [System.IdentityModel.Services.ClaimsPrincipalPermission](#)

[System.Security.CodeAccessPermission](#)

[System.Security.IPermission](#)

[System.Security.Permissions.PrincipalPermission](#)

[System.Security.PermissionSet](#)

[More...](#)

Examples

This example shows how to define a permission class for use with code access security. All of the necessary permission interfaces are implemented.

C#

```
using System;
using System.Security;
using System.Security.Permissions;
using System.Reflection;

// Enumerated type for permission states.
[Serializable]
public enum SoundPermissionState
{
    NoSound = 0,
    PlaySystemSounds = 1,
    PlayAnySound = 2
}
```

```

// Derive from CodeAccessPermission to gain implementations of the following
// sealed IStackWalk methods: Assert, Demand, Deny, and PermitOnly.
// Implement the following abstract IPermission methods: Copy, Intersect,
and IsSubSetOf.
// Implementing the Union method of the IPermission class is optional.
// Implement the following abstract ISecurityEncodable methods: FromXml and
ToXml.
// Making the class 'sealed' is optional.

public sealed class SoundPermission : CodeAccessPermission, IPermission,
    IUnrestrictedPermission, ISecurityEncodable, ICloneable
{
    private Boolean m_specifiedAsUnrestricted = false;
    private SoundPermissionState m_flags = SoundPermissionState.NoSound;

        // This constructor creates and initializes a permission with generic
access.
    public SoundPermission(PermissionState state)
    {
        m_specifiedAsUnrestricted = (state == PermissionState.Unrestricted);
    }

        // This constructor creates and initializes a permission with specific
access.
    public SoundPermission(SoundPermissionState flags)
    {
        if (!Enum.IsDefined(typeof(SoundPermissionState), flags))
            throw new ArgumentException
                ("flags value is not valid for the SoundPermissionState
enumrated type");
        m_specifiedAsUnrestricted = false;
        m_flags = flags;
    }

        // For debugging, return the state of this object as XML.
    public override String ToString() { return ToXml().ToString(); }

        // Private method to cast (if possible) an IPermission to the type.
    private SoundPermission VerifyTypeMatch(IPermission target)
    {
        if (GetType() != target.GetType())
            throw new ArgumentException(String.Format("target must be of the
{0} type",
                GetType().FullName));
        return (SoundPermission)target;
    }

        // This is the Private Clone helper method.
    private SoundPermission Clone(Boolean specifiedAsUnrestricted,
SoundPermissionState flags)
    {
        SoundPermission soundPerm = (SoundPermission)Clone();
        soundPerm.m_specifiedAsUnrestricted = specifiedAsUnrestricted;
        soundPerm.m_flags = specifiedAsUnrestricted ?
SoundPermissionState.PlayAnySound : m_flags;
    }
}

```

```

        return soundPerm;
    }

    #region IPermission Members
    // Return a new object that contains the intersection of 'this' and
    'target'.
    public override IPermission Intersect(IPermision target)
    {
        // If 'target' is null, return null.
        if (target == null) return null;

        // Both objects must be the same type.
        SoundPermission soundPerm = VerifyTypeMatch(target);

        // If 'this' and 'target' are unrestricted, return a new
        unrestricted permission.
        if (m_specifiedAsUnrestricted &&
soundPerm.m_specifiedAsUnrestricted)
            return Clone(true, SoundPermissionState.PlayAnySound);

        // Calculate the intersected permissions. If there are none, return
        null.
        SoundPermissionState val = (SoundPermissionState)
            Math.Min((Int32)m_flags, (Int32)soundPerm.m_flags);
        if (val == 0) return null;

        // Return a new object with the intersected permission value.
        return Clone(false, val);
    }

    // Called by the Demand method: returns true if 'this' is a subset of
    'target'.
    public override Boolean IsSubsetOf(IPermision target)
    {
        // If 'target' is null and this permission allows nothing, return
        true.
        if (target == null) return m_flags == 0;

        // Both objects must be the same type.
        SoundPermission soundPerm = VerifyTypeMatch(target);

        // Return true if the permissions of 'this' is a subset of 'target'.
        return m_flags <= soundPerm.m_flags;
    }

    // Return a new object that matches 'this' object's permissions.
    public sealed override IPermission Copy()
    {
        return (IPermision)Clone();
    }

    // Return a new object that contains the union of 'this' and 'target'.
    // Note: You do not have to implement this method. If you do not, the
    version
    // in CodeAccessPermission does this:

```

```

//      1. If target is not null, a NotSupportedException is thrown.
//      2. If target is null, then Copy is called and the new object is
returned.
public override IPermission Union(IPermission target)
{
    // If 'target' is null, then return a copy of 'this'.
    if (target == null) return Copy();

    // Both objects must be the same type.
    SoundPermission soundPerm = VerifyTypeMatch(target);

    // If 'this' or 'target' are unrestricted, return a new unrestricted
permission.
    if (m_specifiedAsUnrestricted ||
soundPerm.m_specifiedAsUnrestricted)
        return Clone(true, SoundPermissionState.PlayAnySound);

    // Return a new object with the calculated, unioned permission
value.
    return Clone(false, (SoundPermissionState)
Math.Max((Int32)m_flags, (Int32)soundPerm.m_flags));
}

#endregion

#region ISecurityEncodable Members
// Populate the permission's fields from XML.
public override void FromXml(SecurityElement e)
{
    m_specifiedAsUnrestricted = false;
    m_flags = 0;

    // If XML indicates an unrestricted permission, make this permission
unrestricted.
    String s = (String)e.Attributes["Unrestricted"];
    if (s != null)
    {
        m_specifiedAsUnrestricted = Convert.ToBoolean(s);
        if (m_specifiedAsUnrestricted)
            m_flags = SoundPermissionState.PlayAnySound;
    }

    // If XML indicates a restricted permission, parse the flags.
    if (!m_specifiedAsUnrestricted)
    {
        s = (String)e.Attributes["Flags"];
        if (s != null)
        {
            m_flags = (SoundPermissionState)
Convert.ToInt32(Enum.Parse(typeof(SoundPermission), s,
true));
        }
    }
}

// Produce XML from the permission's fields.

```

```

public override SecurityElement ToXml()
{
    // These first three lines create an element with the required
    // format.
    SecurityElement e = new SecurityElement("IPermission");
    // Replace the double quotation marks ("") with single quotation
    // marks ('')
    // to remain XML compliant when the culture is not neutral.
    e.AddAttribute("class",
    GetType().AssemblyQualifiedName.Replace('\"', '\"'));
    e.AddAttribute("version", "1");

    if (!m_specifiedAsUnrestricted)
        e.AddAttribute("Flags",
    Enum.Format(typeof(SoundPermissionState), m_flags, "G"));
    else
        e.AddAttribute("Unrestricted", "true");
    return e;
}

#endregion

#region IUnrestrictedPermission Members
// Returns true if permission is effectively unrestricted.
public Boolean IsUnrestricted()
{
    // This means that the object is unrestricted at runtime.
    return m_flags == SoundPermissionState.PlayAnySound;
}
#endregion

#region ICloneable Members

// Return a copy of the permission.
public Object Clone() { return MemberwiseClone(); }

#endregion
}

```

Remarks

The XML representation of permissions is used to describe instances of permissions for code requests, declarative security permission sets, and security policy configuration.

Note

You must implement this interface for any new permission object.

Methods

FromXml(SecurityElement)	Reconstructs a security object with a specified state from an XML encoding.
ToXml()	Creates an XML encoding of the security object and its current state.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

ISecurityEncodable.FromXml(SecurityElement) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Reconstructs a security object with a specified state from an XML encoding.

C#

```
public void FromXml (System.Security.SecurityElement e);
```

Parameters

e [SecurityElement](#)

The XML encoding to use to reconstruct the security object.

Examples

The following code example demonstrates implementing the [FromXml](#) method. This code example is part of a larger example provided for the [ISecurityEncodable](#) class.

C#

```
// Populate the permission's fields from XML.
public override void FromXml(SecurityElement e)
{
    m_specifiedAsUnrestricted = false;
    m_flags = 0;

    // If XML indicates an unrestricted permission, make this permission
    unrestricted.
    String s = (String)e.Attributes["Unrestricted"];
    if (s != null)
    {
        m_specifiedAsUnrestricted = Convert.ToBoolean(s);
        if (m_specifiedAsUnrestricted)
            m_flags = SoundPermissionState.PlayAnySound;
    }
}
```

```

// If XML indicates a restricted permission, parse the flags.
if (!m_specifiedAsUnrestricted)
{
    s = (String)e.Attributes["Flags"];
    if (s != null)
    {
        m_flags = (SoundPermissionState)
            Convert.ToInt32(Enum.Parse(typeof(SoundPermission), s, true));
    }
}

```

Remarks

Custom code that extends security objects needs to implement the [ToXml](#) and [FromXml](#) methods to make the objects security-encodable.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

ISecurityEncodable.Xml Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Creates an XML encoding of the security object and its current state.

C#

```
public System.Security.SecurityElement? ToXml();
```

Returns

[SecurityElement](#)

An XML encoding of the security object, including any state information.

Examples

The following code example demonstrates implementing the [FromXml](#) method. This code example is part of a larger example provided for the [ISecurityEncodable](#) class.

C#

```
// Produce XML from the permission's fields.
public override SecurityElement ToXml()
{
    // These first three lines create an element with the required format.
    SecurityElement e = new SecurityElement("IPermission");
    // Replace the double quotation marks ("") with single quotation marks
    ('')
    // to remain XML compliant when the culture is not neutral.
    e.AddAttribute("class", GetType().AssemblyQualifiedName.Replace('\"', 
'\''));
    e.AddAttribute("version", "1");

    if (!m_specifiedAsUnrestricted)
        e.AddAttribute("Flags", Enum.Format(typeof(SoundPermissionState),
m_flags, "G"));
    else
        e.AddAttribute("Unrestricted", "true");
```

```
    return e;  
}
```

Remarks

Custom code that extends security objects needs to implement the [ToXml](#) and [FromXml](#) methods to make the objects security-encodable.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

IStackWalk Interface

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

⊗ Caution

Code Access Security is not supported or honored by the runtime.

Manages the stack walk that determines whether all callers in the call stack have the required permissions to access a protected resource.

C#

```
[System.Obsolete("Code Access Security is not supported or honored by the  
runtime.", DiagnosticId="SYSLIB0003", UrlFormat="https://aka.ms/dotnet-  
warnings/{0}")]  
public interface IStackWalk
```

Derived [System.Security.CodeAccessPermission](#)
[System.Security.PermissionSet](#)

Attributes [ObsoleteAttribute](#)

Remarks

⊗ Caution

Code Access Security (CAS) has been deprecated across all versions of .NET Framework and .NET. Recent versions of .NET do not honor CAS annotations and produce errors if CAS-related APIs are used. Developers should seek alternative means of accomplishing security tasks.

Partially trusted code always presents a security risk. It can sometimes be manipulated to perform actions on behalf of malicious code that does not have permission to access

a resource. In this way, malicious code can achieve higher security access than it should be allowed.

The common language runtime helps protect managed code from these attacks by running a stack walk on all calls. The stack walk requires that all code in the call stack has permission to access a protected resource. Because the code attempting the attack will always be somewhere in the call stack, it will be unable to exceed its own security permissions.

Methods

[\[+\] Expand table](#)

Assert()	Asserts that the calling code can access the resource identified by the current permission object, even if callers higher in the stack have not been granted permission to access the resource.
Demand()	Determines at run time whether all callers in the call stack have been granted the permission specified by the current permission object.
Deny()	Causes every Demand() for the current object that passes through the calling code to fail.
Permit Only()	Causes every Demand() for all objects except the current one that passes through the calling code to fail, even if code higher in the call stack has been granted permission to access other resources.

Applies to

Product	Versions (<i>Obsolete</i>)
.NET	Core 3.0, Core 3.1 (5, 6, 7, 8, 9)
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2 (5, 6, 7, 8, 9)
Windows Desktop	(5, 6, 7, 8, 9)



Collaborate with us on



.NET feedback

GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

IStackWalk.Assert Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Asserts that the calling code can access the resource identified by the current permission object, even if callers higher in the stack have not been granted permission to access the resource.

C#

```
public void Assert();
```

Exceptions

[SecurityException](#)

The calling code does not have [Assertion](#).

Remarks

Calling [Assert](#) stops the permission check on callers higher in the call stack. Therefore, even if these callers do not have the requisite permissions, they can still access resources. An assertion is effective only if the code that calls [Assert](#) passes the security check for the permission that it is asserting.

A call to [Assert](#) is effective until the calling code returns to its caller or until a subsequent call to [Assert](#) renders the previous assertion ineffective. Also, [RevertAssert](#) or [RevertAll](#) removes a pending [Assert](#).

[Assert](#) is ignored for a permission not granted because a demand for that permission will not succeed. However, if code lower on the call stack calls [Demand](#) for that permission, a [SecurityException](#) is thrown when the stack walk reaches the code that tried to call [Assert](#). This happens because the code that called [Assert](#) has not been granted the permission, even though it tried to [Assert](#) it.

⊗ Caution

Because calling **[Assert](#)** removes the requirement that all code in the call chain must be granted permission to access the specified resource, it can open up security vulnerabilities if used incorrectly or inappropriately. Therefore, it should be used with great caution.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

See also

- [Using the Assert Method](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

IStackWalk.Demand Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Determines at run time whether all callers in the call stack have been granted the permission specified by the current permission object.

C#

```
public void Demand ();
```

Exceptions

[SecurityException](#)

A caller higher in the call stack does not have the permission specified by the current permission object.

-or-

A caller in the call stack has called [Deny\(\)](#) on the current permission object.

Remarks

This method is typically used by secure libraries to ensure that callers have permission to access a resource. For example, a file class in a secure class library calls [Demand](#) for the necessary [FileIOPermission](#) before performing a file operation requested by the caller.

The permissions of the code that calls this method are not examined; the check begins from the immediate caller of that code and proceeds up the stack. [Demand](#) succeeds only if no [SecurityException](#) is raised.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9

Product	Versions
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

IStackWalk.Deny Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Causes every [Demand\(\)](#) for the current object that passes through the calling code to fail.

C#

```
public void Deny();
```

Remarks

This method prevents callers higher in the call stack from accessing the protected resource through the code that calls this method, even if those callers have been granted permission to access it. The call stack is typically represented as growing down, so that methods higher in the call stack call methods lower in the call stack.

[Deny](#) can limit the liability of the programmer or help prevent accidental security vulnerabilities because it helps prevent the method that calls [Deny](#) from being used to access the resource protected by the denied permission. If a method calls [Deny](#) on a permission, and if a [Demand](#) for that permission is invoked by a caller lower in the call stack, that security check will fail when it reaches the [Deny](#).

[Deny](#) is ignored for a permission not granted because a demand for that permission will not succeed.

Notes to Implementers

You cannot override this method.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

IStackWalk.PermitOnly Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Causes every [Demand\(\)](#) for all objects except the current one that passes through the calling code to fail, even if code higher in the call stack has been granted permission to access other resources.

C#

```
public void PermitOnly();
```

Remarks

[PermitOnly](#) is similar to [Deny](#), in that both cause stack walks to fail when they would otherwise succeed. The difference is that [Deny](#) specifies permissions that will cause the stack walk to fail, but [PermitOnly](#) specifies the only permissions that do not cause the stack walk to fail. Call this method to ensure that your code can be used to access only the specified resources.

[PermitOnly](#) is ignored for a permission not granted because a demand for that permission will not succeed. However, if code lower on the call stack later calls [Demand](#) for that permission, a [SecurityException](#) is thrown when the stack walk reaches the code that tried to call [PermitOnly](#). This is because the code that called [PermitOnly](#) has not been granted the permission, even though it called [PermitOnly](#) for that permission. The call stack is typically represented as growing down, so that methods higher in the call stack call methods lower in the call stack.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1

Product	Versions
.NET Platform Extensions	2.1, 2.2

 **Collaborate with us on GitHub**

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 .NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PartialTrustVisibilityLevel Enum

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Specifies the default partial-trust visibility for code that is marked with the [AllowPartiallyTrustedCallersAttribute](#) (APTCA) attribute.

C#

```
public enum PartialTrustVisibilityLevel
```

Inheritance Object → [ValueType](#) → [Enum](#) → PartialTrustVisibilityLevel

Fields

[] [Expand table](#)

NotVisibleByDefault	1	The assembly has been audited for partial trust, but it is not visible to partial-trust code in all hosts. To make the assembly visible to partial-trust code, add it to the PartialTrustVisibleAssemblies property.
VisibleToAllHosts	0	The assembly can always be called by partial-trust code.

Remarks

[PartialTrustVisibilityLevel](#) is passed as a property setting parameter to the [AllowPartiallyTrustedCallersAttribute.AllowPartiallyTrustedCallersAttribute](#) constructor. If no parameter is passed to the constructor, the default is [VisibleToAllHosts](#).

You enable partially trusted assemblies that are identified as [NotVisibleByDefault](#) by adding them to the [PartialTrustVisibleAssemblies](#) property of their application domain. If you enable an assembly that references (directly or indirectly) other partially trusted assemblies that are [NotVisibleByDefault](#), those other assemblies should be enabled as well.

When an APTCA library that specifies a `PartialTrustVisibilityLevel` and that is eligible for code sharing is loaded for the first time, it is loaded into the shared domain. Whenever that assembly is loaded with the same `PartialTrustVisibilityLevel` into another domain, it will be shared. However, if the assembly is loaded with a different `PartialTrustVisibilityLevel`, it will not be shared.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet Class

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

⊗ Caution

Code Access Security is not supported or honored by the runtime.

Represents a collection that can contain many different types of permissions.

C#

```
[System.Obsolete("Code Access Security is not supported or honored by the  
runtime.", DiagnosticId="SYSLIB0003", UrlFormat="https://aka.ms/dotnet-  
warnings/{0}")]  
public class PermissionSet : System.Collections.ICollection,  
System.Runtime.Serialization.IDeserializationCallback,  
System.Security.ISecurityEncodable, System.Security.IStackWalk
```

Inheritance Object → PermissionSet

Derived [System.Security.NamedPermissionSet](#)
[System.Security.ReadOnlyPermissionSet](#)

Attributes [ObsoleteAttribute](#)

Implements [ICollection](#) , [IEnumerable](#) , [IDeserializationCallback](#) , [ISecurityEncodable](#) ,
[IStackWalk](#)

Examples

The following code example demonstrates the use of the [PermissionSet](#) class and members.

C#

```
// This sample demonstrates the use of the PermissionSet class.

using System;
using System.Reflection;
using System.Security.Permissions;
using System.Security;
using System.IO;
using System.Collections;

class MyClass
{
    public static void PermissionSetDemo()
    {
        Console.WriteLine("Executing PermissionSetDemo");
        try
        {
            // Open a new PermissionSet.
            PermissionSet ps1 = new PermissionSet(PermissionState.None);
            Console.WriteLine("Adding permission to open a file from a file
dialog box.");
            // Add a permission to the permission set.
            ps1.AddPermission(
                new FileDialogPermission(FileDialogPermissionAccess.Open));
            Console.WriteLine("Demanding permission to open a file.");
            ps1.Demand();
            Console.WriteLine("Demand succeeded.");
            Console.WriteLine("Adding permission to save a file from a file
dialog box.");
            ps1.AddPermission(
                new FileDialogPermission(FileDialogPermissionAccess.Save));
            Console.WriteLine("Demanding permission to open and save a
file.");
            ps1.Demand();
            Console.WriteLine("Demand succeeded.");
            Console.WriteLine("Adding permission to read environment
variable USERNAME.");
            ps1.AddPermission(
                new EnvironmentPermission(EnvironmentPermissionAccess.Read,
"USERNAME"));
            ps1.Demand();
            Console.WriteLine("Demand succeeded.");
            Console.WriteLine("Adding permission to read environment
variable COMPUTERNAME.");
            ps1.AddPermission(
                new EnvironmentPermission(EnvironmentPermissionAccess.Read,
"COMPUTERNAME"));
            // Demand all the permissions in the set.
            Console.WriteLine("Demand all permissions.");
            ps1.Demand();
            Console.WriteLine("Demand succeeded.");
            // Display the number of permissions in the set.
            Console.WriteLine("Number of permissions = " + ps1.Count);
            // Display the value of the IsSynchronized property.
            Console.WriteLine("IsSynchronized property = " +

```

```

ps1.IsSynchronized);
    // Display the value of the IsReadOnly property.
    Console.WriteLine("IsReadOnly property = " + ps1.IsReadOnly);
    // Display the value of the SyncRoot property.
    Console.WriteLine("SyncRoot property = " + ps1.SyncRoot);
    // Display the result of a call to the
ContainsNonCodeAccessPermissions method.
    // Gets a value indicating whether the PermissionSet contains
permissions
        // that are not derived from CodeAccessPermission.
        // Returns true if the PermissionSet contains permissions that
are not
            // derived from CodeAccessPermission; otherwise, false.
            Console.WriteLine("ContainsNonCodeAccessPermissions method
returned " +
                ps1.ContainsNonCodeAccessPermissions());
            Console.WriteLine("Value of the permission set ToString = \n" +
ps1.ToString());
            PermissionSet ps2 = new PermissionSet(PermissionState.None);
            // Create a second permission set and compare it to the first
permission set.
            ps2.AddPermission(
                new EnvironmentPermission(EnvironmentPermissionAccess.Read,
"USERNAME"));
            ps2.AddPermission(
                new EnvironmentPermission(EnvironmentPermissionAccess.Write,
"COMPUTERNAME"));
            Ienumerator list = ps1.GetEnumerator();
            Console.WriteLine("Permissions in first permission set:");
            while (list.MoveNext())
                Console.WriteLine(list.Current.ToString());
            Console.WriteLine("Second permission IsSubsetOf first permission
= " + ps2.IsSubsetOf(ps1));
            // Display the intersection of two permission sets.
            PermissionSet ps3 = ps2.Intersect(ps1);
            Console.WriteLine("The intersection of the first permission set
and "
                + "the second permission set = " + ps3.ToString());
// Create a new permission set.
            PermissionSet ps4 = new PermissionSet(PermissionState.None);
            ps4.AddPermission(
                new FileIOPermission(FileIOPermissionAccess.Read,
"C:\\\\Temp\\\\Testfile.txt"));
            ps4.AddPermission(
                new FileIOPermission(FileIOPermissionAccess.Read |
FileIOPermissionAccess.Write |
FileIOPermissionAccess.Append,
"C:\\\\Temp\\\\Testfile.txt"));
            // Display the union of two permission sets.
            PermissionSet ps5 = ps3.Union(ps4);
            Console.WriteLine("The union of permission set 3 and permission
set 4 = "
                + ps5.ToString());
// Remove FileIOPermission from the permission set.
            ps5.RemovePermission(typeof(FileIOPermission));

```

```

        Console.WriteLine("The last permission set after removing
FileIOPermission = "
+ ps5.ToString());
// Change the permission set using SetPermission.
ps5.SetPermission(new
EnvironmentPermission(EnvironmentPermissionAccess.AllAccess, "USERNAME"));
Console.WriteLine("Permission set after SetPermission = " +
ps5.ToString());
// Display result of ToXml and FromXml operations.
PermissionSet ps6 = new PermissionSet(PermissionState.None);
ps6.FromXml(ps5.ToXml());
Console.WriteLine("Result of ToFromXml = " + ps6.ToString() +
"\n");
// Display results of PermissionSet.GetEnumerator.
IEnumerator psEnumerator = ps1.GetEnumerator();
while (psEnumerator.MoveNext())
{
    Console.WriteLine(psEnumerator.Current);
}
// Check for an unrestricted permission set.
PermissionSet ps7 = new
PermissionSet(PermissionState.Unrestricted);
Console.WriteLine("Permission set is unrestricted = " +
ps7.IsUnrestricted());
// Create and display a copy of a permission set.
ps7 = ps5.Copy();
Console.WriteLine("Result of copy = " + ps7.ToString());
}
catch (Exception e)
{
    Console.WriteLine(e.Message.ToString());
}
}

static void Main(string[] args)
{
    PermissionSetDemo();
}
}

```

Remarks

⊗ Caution

Code Access Security (CAS) has been deprecated across all versions of .NET Framework and .NET. Recent versions of .NET do not honor CAS annotations and produce errors if CAS-related APIs are used. Developers should seek alternative means of accomplishing security tasks.

You can use [PermissionSet](#) to perform operations on several different permissions as a group.

Constructors

[\[+\] Expand table](#)

PermissionSet(PermissionSet)	Initializes a new instance of the PermissionSet class with initial values taken from the <code>permSet</code> parameter.
PermissionSet(PermissionState)	Initializes a new instance of the PermissionSet class with the specified PermissionState .

Properties

[\[+\] Expand table](#)

Count	Gets the number of permission objects contained in the permission set.
IsReadOnly	Gets a value indicating whether the collection is read-only.
IsSynchronized	Gets a value indicating whether the collection is guaranteed to be thread safe.
SyncRoot	Gets the root object of the current collection.

Methods

[\[+\] Expand table](#)

AddPermission(IPermission)	Adds a specified permission to the PermissionSet .
AddPermissionImpl(IPermission)	Adds a specified permission to the PermissionSet .
Assert()	Declares that the calling code can access the resource protected by a permission demand through the code that calls this method, even if callers higher in the stack have not been granted permission to access the resource. Using Assert() can create security vulnerabilities.
ContainsNonCodeAccessPermissions()	Gets a value indicating whether the PermissionSet contains permissions that are not derived from CodeAccessPermission .

ConvertPermission	Obsolete.
Set(String, Byte[], String)	Converts an encoded PermissionSet from one XML encoding format to another XML encoding format.
Copy()	Creates a copy of the PermissionSet .
CopyTo(Array, Int32)	Copies the permission objects of the set to the indicated location in an Array .
Demand()	Forces a SecurityException at run time if all callers higher in the call stack have not been granted the permissions specified by the current instance.
Deny()	Obsolete. Causes any Demand() that passes through the calling code for a permission that has an intersection with a permission of a type contained in the current PermissionSet to fail.
Equals(Object)	Determines whether the specified PermissionSet or NamedPermissionSet object is equal to the current PermissionSet .
FromXml(SecurityElement)	Reconstructs a security object with a specified state from an XML encoding.
GetEnumerator()	Returns an enumerator for the permissions of the set.
GetEnumeratorImpl()	Returns an enumerator for the permissions of the set.
GetHashCode()	Gets a hash code for the PermissionSet object that is suitable for use in hashing algorithms and data structures such as a hash table.
GetPermission(Type)	Gets a permission object of the specified type, if it exists in the set.
GetPermissionImpl(Type)	Gets a permission object of the specified type, if it exists in the set.
GetType()	Gets the Type of the current instance. (Inherited from Object)
Intersect(PermissionSet)	Creates and returns a permission set that is the intersection of the current PermissionSet and the specified PermissionSet .
IsEmpty()	Gets a value indicating whether the PermissionSet is empty.
IsSubsetOf(PermissionSet)	Determines whether the current PermissionSet is a subset of the specified PermissionSet .
IsUnrestricted()	Determines whether the PermissionSet is Unrestricted .
MemberwiseClone()	Creates a shallow copy of the current Object . (Inherited from Object)

PermitOnly()	Causes any Demand() that passes through the calling code for any PermissionSet that is not a subset of the current PermissionSet to fail.
RemovePermission(Type)	Removes a permission of a certain type from the set.
RemovePermissionImpl(Type)	Removes a permission of a certain type from the set.
RevertAssert()	Causes any previous Assert() for the current frame to be removed and no longer be in effect.
SetPermission(IPermission)	Sets a permission to the PermissionSet , replacing any existing permission of the same type.
SetPermissionImpl(IPermission)	Sets a permission to the PermissionSet , replacing any existing permission of the same type.
ToString()	Returns a string representation of the PermissionSet .
ToXml()	Creates an XML encoding of the security object and its current state.
Union(PermissionSet)	Creates a PermissionSet that is the union of the current PermissionSet and the specified PermissionSet .

Explicit Interface Implementations

[] [Expand table](#)

IDeserializationCallback.OnDeserialization(Object)	Runs when the entire object graph has been deserialized.
--	--

Extension Methods

[] [Expand table](#)

Cast<TResult>(IEnumerable)	Casts the elements of an IEnumerable to the specified type.
OfType<TResult>(IEnumerable)	Filters the elements of an IEnumerable based on a specified type.
AsParallel(IEnumerable)	Enables parallelization of a query.
AsQueryable(IEnumerable)	Converts an IEnumerable to an IQueryable .

Applies to

Product	Versions (<i>Obsolete</i>)
.NET	Core 3.0, Core 3.1 (5, 6, 7, 8, 9)
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2 (5, 6, 7, 8, 9)
Windows Desktop	(5, 6, 7, 8, 9)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet Constructors

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Initializes a new instance of the [PermissionSet](#) class.

Overloads

[+] Expand table

PermissionSet(PermissionState)	Initializes a new instance of the PermissionSet class with the specified PermissionState .
PermissionSet(PermissionSet)	Initializes a new instance of the PermissionSet class with initial values taken from the <code>permSet</code> parameter.

PermissionSet(PermissionState)

Initializes a new instance of the [PermissionSet](#) class with the specified [PermissionState](#).

C#

```
public PermissionSet (System.Security.Permissions.PermissionState state);
```

Parameters

state [PermissionState](#)

One of the enumeration values that specifies the permission set's access to resources.

Exceptions

[ArgumentException](#)

The `state` parameter is not a valid [PermissionState](#).

Examples

The following code example shows the use of the [PermissionSet](#) constructor to create a permission set with a permission state of [None](#). This code example is part of a larger example provided for the [PermissionSet](#) class.

C#

```
// Open a new PermissionSet.  
PermissionSet ps1 = new PermissionSet(PermissionState.None);  
Console.WriteLine("Adding permission to open a file from a file dialog  
box.");  
// Add a permission to the permission set.  
ps1.AddPermission(  
    new FileDialogPermission(FileDialogPermissionAccess.Open));  
Console.WriteLine("Demanding permission to open a file.");  
ps1.Demand();  
Console.WriteLine("Demand succeeded.");
```

Remarks

The [Unrestricted](#) state allows all permissions that implement the [IUnrestrictedPermission](#) interface, while [None](#) allows no permissions.

Use [AddPermission](#) on an empty [PermissionSet](#) to define the set in greater detail.

Applies to

- ▼ .NET 9 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

PermissionSet(PermissionSet)

Initializes a new instance of the [PermissionSet](#) class with initial values taken from the `permSet` parameter.

C#

```
public PermissionSet (System.Security.PermissionSet? permSet);
```

Parameters

permSet [PermissionSet](#)

The set from which to take the value of the new [PermissionSet](#), or `null` to create an empty [PermissionSet](#).

Remarks

The new [PermissionSet](#) contains copies of the permissions contained in the specified [PermissionSet](#).

ⓘ Note

This is equivalent to [Copy](#) when the `permSet` parameter is not `null`.

Applies to

▼ .NET 9 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

more information, see [our contributor guide](#).

 [Provide product feedback](#)

PermissionSet.Count Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets the number of permission objects contained in the permission set.

C#

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

Property Value

[Int32](#)

The number of permission objects contained in the [PermissionSet](#).

Implements

[Count](#)

Examples

The following code example shows the use of the [Count](#) property to get the number of permission objects in a permission set. This code example is part of a larger example provided for the [PermissionSet](#) class.

C#

```
// Display the number of permissions in the set.  
Console.WriteLine("Number of permissions = " + ps1.Count);
```

Remarks

In the `None` or `Unrestricted` state this returns zero, because no actual permission object instances are used.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.IsReadOnly Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets a value indicating whether the collection is read-only.

C#

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

Property Value

[Boolean](#)

Always `false`.

Examples

The following code example shows the value returned by the [IsReadOnly](#) property. This code example is part of a larger example provided for the [PermissionSet](#) class.

C#

```
// Display the value of the IsReadOnly property.  
Console.WriteLine("IsReadOnly property = " + ps1.IsReadOnly);
```

Remarks

A [PermissionSet](#) cannot be read-only, so this property is always `false`.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9

Product	Versions
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.IsSynchronized Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets a value indicating whether the collection is guaranteed to be thread safe.

C#

```
public virtual bool IsSynchronized { get; }
```

Property Value

[Boolean](#)

Always `false`.

Implements

[IsSynchronized](#)

Examples

The following code example shows the value returned by the [IsSynchronized](#) property. This code example is part of a larger example provided for the [PermissionSet](#) class.

C#

```
// Display the value of the IsSynchronized property.  
Console.WriteLine("IsSynchronized property = " + ps1.IsSynchronized);
```

Remarks

[PermissionSet](#) does not automatically handle thread safety, so this property is always `false`.

This method is required to support [ICollection](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.SyncRoot Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets the root object of the current collection.

C#

```
public virtual object SyncRoot { get; }
```

Property Value

[Object](#)

The root object of the current collection.

Implements

[SyncRoot](#)

Examples

The following code example shows the use of [SyncRoot](#) property to get the root object of the current collection. This code example is part of a larger example provided for the [PermissionSet](#) class.

C#

```
// Display the value of the SyncRoot property.  
Console.WriteLine("SyncRoot property = " + ps1.SyncRoot);
```

Remarks

This method is required to support [ICollection](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.AddPermission(IPermission) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Adds a specified permission to the [PermissionSet](#).

C#

```
public System.Security.IPermission? AddPermission  
(System.Security.IPermission? perm);
```

Parameters

perm [IPermission](#)

The permission to add.

Returns

[IPermission](#)

The union of the permission added and any permission of the same type that already exists in the [PermissionSet](#).

Exceptions

[InvalidOperationException](#)

The method is called from a [ReadOnlyPermissionSet](#).

Examples

The following code example shows the use of the [AddPermission](#) method to add a permission to a permission set. This code example is part of a larger example provided for the [PermissionSet](#) class.

C#

```
// Add a permission to the permission set.  
ps1.AddPermission(  
    new FileDialogPermission(FileDialogPermissionAccess.Open));
```

Remarks

If a permission of the same type as the added permission already exists in the [PermissionSet](#), the new permission is the union of the existing permission object and the specified permission object. For example, if a permission that implements [IUnrestrictedPermission](#) is added to an [UnrestrictedPermissionSet](#), the resulting union is the original [Unrestricted PermissionSet](#).

Notes to Inheritors

When you inherit from [PermissionSet](#), you can change the behavior of the [AddPermission\(IPermission\)](#) method by overriding the [AddPermissionImpl\(IPermission\)](#) method.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.AddPermission Impl(IPermission) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Adds a specified permission to the [PermissionSet](#).

C#

```
protected virtual System.Security.IPermission? AddPermissionImpl  
(System.Security.IPermission? perm);
```

Parameters

perm [IPermission](#)

The permission to add.

Returns

[IPermission](#)

The union of the permission added and any permission of the same type that already exists in the [PermissionSet](#), or `null` if `perm` is `null`.

Exceptions

[InvalidOperationException](#)

The method is called from a [ReadOnlyPermissionSet](#).

Remarks

The [AddPermissionImpl](#) method is the implementation for the [AddPermission](#) method.

If a permission of the same type as the added permission already exists in the [PermissionSet](#), the new permission is the union of the existing permission object and the specified permission object. For example, if a permission that implements

`IUnrestrictedPermission` is added to an `UnrestrictedPermissionSet`, the resulting union is the original `Unrestricted PermissionSet`.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.Assert Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Declares that the calling code can access the resource protected by a permission demand through the code that calls this method, even if callers higher in the stack have not been granted permission to access the resource. Using [Assert\(\)](#) can create security vulnerabilities.

C#

```
public void Assert();
```

Implements

[Assert\(\)](#)

Exceptions

[SecurityException](#)

The [PermissionSet](#) instance asserted has not been granted to the asserting code.

-or-

There is already an active [Assert\(\)](#) for the current frame.

Remarks

This is the only way to assert multiple permissions at the same time within a frame because only one [Assert](#) can be active on a frame. [Assert](#) is only effective for granted permissions. Call the [CodeAccessPermission.RevertAssert](#) or [CodeAccessPermission.RevertAll](#) method to cancel an active [Assert](#).

⊗ Caution

Because calling the [Assert](#) method removes the requirement that all code in the call chain must be granted permission to access the specified resource, it can open up security vulnerabilities if used incorrectly or inappropriately. Therefore, it should be used with great caution.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

See also

- [Using the Assert Method](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.ContainsNonCodeAccessPermissions Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets a value indicating whether the [PermissionSet](#) contains permissions that are not derived from [CodeAccessPermission](#).

C#

```
public bool ContainsNonCodeAccessPermissions();
```

Returns

[Boolean](#)

`true` if the [PermissionSet](#) contains permissions that are not derived from [CodeAccessPermission](#); otherwise, `false`.

Examples

The following code example shows the use of the [ContainsNonCodeAccessPermissions](#) method. This code example is part of a larger example provided for the [PermissionSet](#) class.

C#

```
// Display the result of a call to the ContainsNonCodeAccessPermissions
// method.
// Gets a value indicating whether the PermissionSet contains permissions
// that are not derived from CodeAccessPermission.
// Returns true if the PermissionSet contains permissions that are not
// derived from CodeAccessPermission; otherwise, false.
Console.WriteLine("ContainsNonCodeAccessPermissions method returned " +
    ps1.ContainsNonCodeAccessPermissions());
```

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.ConvertPermissionSet(String, Byte[], String) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

⊗ Caution

This API is now deprecated.

Converts an encoded [PermissionSet](#) from one XML encoding format to another XML encoding format.

C#

```
[System.Obsolete]
public static byte[] ConvertPermissionSet (string inFormat, byte[] inData,
string outFormat);
```

Parameters

inFormat [String](#)

A string representing one of the following encoding formats: ASCII, Unicode, or Binary. Possible values are "XMLASCII" or "XML", "XMLUNICODE", and "BINARY".

inData [Byte\[\]](#)

An XML-encoded permission set.

outFormat [String](#)

A string representing one of the following encoding formats: ASCII, Unicode, or Binary. Possible values are "XMLASCII" or "XML", "XMLUNICODE", and "BINARY".

Returns

[Byte\[\]](#)

An encrypted permission set with the specified output format.

Attributes [ObsoleteAttribute](#)

Exceptions

[NotImplementedException](#)

In all cases.

Remarks

Do not use this method.

Applies to

Product	Versions (<i>Obsolete</i>)
.NET	(Core 3.0, Core 3.1, 5, 6, 7, 8, 9)
.NET Framework	1.1, 2.0, 3.0, 3.5 (4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1)
.NET Platform Extensions	(2.1, 2.2, 3.0, 3.1, 5, 6, 7, 8, 9)
Windows Desktop	(3.0, 3.1, 5, 6, 7, 8, 9)

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 .NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.Copy Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Creates a copy of the [PermissionSet](#).

C#

```
public virtual System.Security.PermissionSet Copy ();
```

Returns

[PermissionSet](#)

A copy of the [PermissionSet](#).

Examples

The following code example shows the use of the [Copy](#) method to create a copy of a permission set. This code example is part of a larger example provided for the [PermissionSet](#) class.

C#

```
// Create and display a copy of a permission set.  
ps7 = ps5.Copy();  
Console.WriteLine("Result of copy = " + ps7.ToString());
```

Remarks

A copy of a [PermissionSet](#) represents the same access to resources as the original object. Changes made to the copy do not affect the original permission set.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.CopyTo(Array, Int32) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Copies the permission objects of the set to the indicated location in an [Array](#).

C#

```
public virtual void CopyTo (Array array, int index);
```

Parameters

array [Array](#)

The target array to which to copy.

index [Int32](#)

The starting position in the array to begin copying (zero based).

Implements

[CopyTo\(Array, Int32\)](#)

Exceptions

[ArgumentNullException](#)

The `array` parameter is `null`.

[ArgumentException](#)

The `array` parameter has more than one dimension.

[IndexOutOfRangeException](#)

The `index` parameter is out of the range of the `array` parameter.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.Demand Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Forces a [SecurityException](#) at run time if all callers higher in the call stack have not been granted the permissions specified by the current instance.

C#

```
public void Demand ();
```

Implements

[Demand\(\)](#)

Exceptions

[SecurityException](#)

A caller in the call chain does not have the permission demanded.

Examples

The following code example shows the use of the [Demand](#) method to demand all the permissions in a permission set. This code example is part of a larger example provided for the [PermissionSet](#) class.

C#

```
// Demand all the permissions in the set.  
Console.WriteLine("Demand all permissions.");  
ps1.Demand();
```

Remarks

Use `Demand` on a `PermissionSet` to ensure that all callers have all permissions in the set with one operation.

The permissions of the code that calls this method are not examined; the check begins from the immediate caller of that code and proceeds up the stack. The call stack is typically represented as growing down, so that methods higher in the call stack call methods lower in the call stack. `Demand` succeeds only if no `SecurityException` is thrown.

If the `PermissionSet` contains permissions that do not inherit from `CodeAccessPermission`, the `Demand` methods of those permissions are called as well.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.Deny Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

⊗ Caution

This API is now deprecated.

Causes any [Demand\(\)](#) that passes through the calling code for a permission that has an intersection with a permission of a type contained in the current [PermissionSet](#) to fail.

C#

```
[System.Obsolete]  
public void Deny ();
```

Implements

[Deny\(\)](#)

Attributes [ObsoleteAttribute](#)

Exceptions

[SecurityException](#)

A previous call to [Deny\(\)](#) has already restricted the permissions for the current stack frame.

Remarks

This method prevents callers higher in the call stack from accessing the protected resource through the code that calls this method, even if those callers have been granted permission to access it. The call stack is typically represented as growing down, so that methods higher in the call stack call methods lower in the call stack.

[Deny](#) can limit the liability of the programmer or help prevent accidental security vulnerabilities because it helps prevent the method that calls [Deny](#) from being used to access the resource protected by the denied permission. If a method calls [Deny](#) on a permission, and if a [Demand](#) for that permission is invoked by a caller lower in the call stack, that security check fails when it reaches the [Deny](#).

[Deny](#) is ignored for a permission that is not granted because a demand for that permission cannot succeed.

Applies to

Product	Versions (<i>Obsolete</i>)
.NET	(Core 3.0, Core 3.1, 5, 6, 7, 8, 9)
.NET Framework	1.1, 2.0, 3.0, 3.5 (4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1)
.NET Platform Extensions	(2.1, 2.2, 3.0, 3.1, 5, 6, 7, 8, 9)
Windows Desktop	(3.0, 3.1, 5, 6, 7, 8, 9)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.Equals(Object) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Determines whether the specified [PermissionSet](#) or [NamedPermissionSet](#) object is equal to the current [PermissionSet](#).

C#

```
public override bool Equals (object? o);
```

Parameters

o [Object](#)

The object to compare with the current [PermissionSet](#).

Returns

[Boolean](#)

`true` if the specified object is equal to the current [PermissionSet](#) object; otherwise, `false`.

Remarks

Equality is determined by the permissions contained in the permission set specified by `obj`. `obj` can be either a [PermissionSet](#) object or a [NamedPermissionSet](#) object. If `obj` is a [NamedPermissionSet](#), the name and description are ignored.

For more information, see [Object.Equals\(Object\)](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9

Product	Versions
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 .NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.FromXml(SecurityElement) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Reconstructs a security object with a specified state from an XML encoding.

C#

```
public virtual void FromXml (System.Security.SecurityElement et);
```

Parameters

et [SecurityElement](#)

The XML encoding to use to reconstruct the security object.

Implements

[FromXml\(SecurityElement\)](#)

Exceptions

[ArgumentNullException](#)

The **et** parameter is **null**.

[ArgumentException](#)

The **et** parameter is not a valid permission element.

-or-

The **et** parameter's version number is not supported.

Examples

The following code example shows the use of the `FromXml` method. This code example is part of a larger example provided for the [PermissionSet](#) class.

C#

```
// Display result of ToXml and FromXml operations.  
PermissionSet ps6 = new PermissionSet(PermissionState.None);  
ps6.FromXml(ps5.ToXml());  
Console.WriteLine("Result of ToFromXml = " + ps6.ToString() + "\n");
```

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.GetEnumerator Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Returns an enumerator for the permissions of the set.

C#

```
public System.Collections.IEnumerator GetEnumerator();
```

Returns

[IEnumerator](#)

An enumerator object for the permissions of the set.

Implements

[GetEnumerator\(\)](#)

Examples

The following code example shows the use of the [GetEnumerator](#) method to list all the permissions in a permission set. This code example is part of a larger example provided for the [PermissionSet](#) class.

C#

```
// Display results of PermissionSet.GetEnumerator.  
IEnumerator psEnumerator = ps1.GetEnumerator();  
while (psEnumerator.MoveNext())  
{  
    Console.WriteLine(psEnumerator.Current);  
}
```

Remarks

Use the enumerator as an index to access individual permission objects in the set.

Notes to Inheritors

When you inherit from [PermissionSet](#), you can change the behavior of the [GetEnumerator\(\)](#) method by overriding the [GetEnumeratorImpl\(\)](#) method.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.GetEnumeratorImpl Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Returns an enumerator for the permissions of the set.

C#

```
protected virtual System.Collections.IEnumerator GetEnumeratorImpl();
```

Returns

[IEnumerator](#)

An enumerator object for the permissions of the set.

Remarks

The [GetEnumeratorImpl](#) method is the implementation for the [GetEnumerator](#) method.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2



Collaborate with us on
GitHub

.NET

.NET feedback

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET is an open source project.
Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

PermissionSet.GetHashCode Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets a hash code for the [PermissionSet](#) object that is suitable for use in hashing algorithms and data structures such as a hash table.

C#

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

Returns

[Int32](#)

A hash code for the current [PermissionSet](#) object.

Remarks

The hash code for two instances of the same permission set might be different, so a hash code should not be used to compare two [PermissionSet](#) objects.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.GetPermission(Type) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets a permission object of the specified type, if it exists in the set.

C#

```
public System.Security.IPermission? GetPermission (Type? permClass);
```

Parameters

permClass [Type](#)

The type of the desired permission object.

Returns

[IPermission](#)

A copy of the permission object of the type specified by the `permClass` parameter contained in the [PermissionSet](#), or `null` if none exists.

Remarks

The method returns `null` for an [Unrestricted PermissionSet](#). Although an [Unrestricted PermissionSet](#) effectively contains all permissions, it does not have any actual instances to return.

Notes to Inheritors

When you inherit from [PermissionSet](#), you can change the behavior of the [GetPermission\(Type\)](#) method by overriding the [GetPermissionImpl\(Type\)](#) method.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.GetPermissionImpl(Type) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets a permission object of the specified type, if it exists in the set.

C#

```
protected virtual System.Security.IPermission? GetPermissionImpl (Type? permClass);
```

Parameters

permClass [Type](#)

The type of the permission object.

Returns

[IPermission](#)

A copy of the permission object, of the type specified by the `permClass` parameter, contained in the [PermissionSet](#), or `null` if none exists.

Remarks

The [GetPermissionImpl](#) method is the implementation for the [GetPermission](#) method.

The method returns `null` for an [Unrestricted PermissionSet](#). Although an [Unrestricted PermissionSet](#) effectively contains all permissions, it does not have any actual instances to return.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.Intersect(PermissionSet) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Creates and returns a permission set that is the intersection of the current [PermissionSet](#) and the specified [PermissionSet](#).

C#

```
public System.Security.PermissionSet? Intersect  
(System.Security.PermissionSet? other);
```

Parameters

other [PermissionSet](#)

A permission set to intersect with the current [PermissionSet](#).

Returns

[PermissionSet](#)

A new permission set that represents the intersection of the current [PermissionSet](#) and the specified target. This object is `null` if the intersection is empty.

Examples

The following code example shows the use of the [Intersect](#) method. This code example is part of a larger example provided for the [PermissionSet](#) class.

C#

```
// Display the intersection of two permission sets.  
PermissionSet ps3 = ps2.Intersect(ps1);  
Console.WriteLine("The intersection of the first permission set and "  
+ "the second permission set = " + ps3.ToString());
```

Remarks

The intersection of two permission sets is a permission set that describes the set of operations they both describe in common. Specifically, it represents the minimum permissions such that any demand that passes both permission sets also passes their intersection.

For each type of permission that is present in both sets, the two instances of those permissions are intersected using the permission's `Intersect` method; the resulting permission is included in the resulting `PermissionSet`. Permission types that exist in only one of the two sets are excluded from the resulting set.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.IsEmpty Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets a value indicating whether the [PermissionSet](#) is empty.

C#

```
public bool IsEmpty();
```

Returns

[Boolean](#)

`true` if the [PermissionSet](#) is empty; otherwise, `false`.

Remarks

A [PermissionSet](#) can be empty and yet contain instances of permissions if those permissions are in the fully-restricted state. Permissions are in a fully restricted state if their `IsSubsetOf` methods return `true` when `null` is passed as a parameter.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.IsSubsetOf(PermissionSet) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Determines whether the current [PermissionSet](#) is a subset of the specified [PermissionSet](#).

C#

```
public bool IsSubsetOf (System.Security.PermissionSet? target);
```

Parameters

target [PermissionSet](#)

The permission set to test for the subset relationship. This must be either a [PermissionSet](#) or a [NamedPermissionSet](#).

Returns

[Boolean](#)

`true` if the current [PermissionSet](#) is a subset of the `target` parameter; otherwise, `false`.

Examples

The following code example shows the use of the [IsSubsetOf](#) method. This code example is part of a larger example provided for the [PermissionSet](#) class.

C#

```
// Create a second permission set and compare it to the first permission
set.
ps2.AddPermission(
    new EnvironmentPermission(EnvironmentPermissionAccess.Read,
"USERNAME"));
ps2.AddPermission(
    new EnvironmentPermission(EnvironmentPermissionAccess.Write,
```

```

    "COMPUTERNAME"));
IEnumerator list = ps1.GetEnumerator();
Console.WriteLine("Permissions in first permission set:");
while (list.MoveNext())
    Console.WriteLine(list.Current.ToString());
Console.WriteLine("Second permission IsSubsetOf first permission = " +
ps2.IsSubsetOf(ps1));

```

Remarks

A [PermissionSet](#) is a subset of the target [PermissionSet](#) if all demands that succeed for the [PermissionSet](#) also succeed for the target. That is, the target contains at least the permissions contained in the subset.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 .NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.IsUnrestricted Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Determines whether the [PermissionSet](#) is [Unrestricted](#).

C#

```
public bool IsUnrestricted();
```

Returns

[Boolean](#)

`true` if the [PermissionSet](#) is [Unrestricted](#); otherwise, `false`.

Examples

The following code example shows the use of the [IsUnrestricted](#) method. This code example is part of a larger example provided for the [PermissionSet](#) class.

C#

```
// Check for an unrestricted permission set.  
PermissionSet ps7 = new PermissionSet(PermissionState.Unrestricted);  
Console.WriteLine("Permission set is unrestricted = " +  
ps7.IsUnrestricted());
```

Remarks

An [Unrestricted](#) [PermissionSet](#) effectively contains all permissions that implement the [IUnrestrictedPermission](#) interface.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.PermitOnly Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Causes any [Demand\(\)](#) that passes through the calling code for any [PermissionSet](#) that is not a subset of the current [PermissionSet](#) to fail.

C#

```
public void PermitOnly();
```

Implements

[PermitOnly\(\)](#)

Remarks

[PermitOnly](#) is similar to [Deny](#), in that both cause stack walks to fail when they would otherwise succeed. The difference is that [Deny](#) specifies permissions that will cause the stack walk to fail, but [PermitOnly](#) specifies the only permissions that do not cause the stack walk to fail. Call this method to ensure that your code can be used to access only the specified resources.

[PermitOnly](#) is ignored for a permission not granted because a demand for that permission cannot succeed. However, if code lower on the call stack later calls [Demand](#) for that permission, a [SecurityException](#) is thrown when the stack walk reaches the code that tried to call [PermitOnly](#). This is because the code that called [PermitOnly](#) has not been granted the permission, even though it called [PermitOnly](#) for that permission. The call stack is typically represented as growing down, so that methods higher in the call stack call methods lower in the call stack.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.RemovePermission(Type) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Removes a permission of a certain type from the set.

C#

```
public System.Security.IPermission? RemovePermission (Type? permClass);
```

Parameters

permClass [Type](#)

The type of permission to delete.

Returns

[IPermission](#)

The permission removed from the set.

Exceptions

[InvalidOperationException](#)

The method is called from a [ReadOnlyPermissionSet](#).

Examples

The following code example shows the use of the [RemovePermission](#) method to remove a [FileIOPermission](#) from a permission set. This code example is part of a larger example provided for the [PermissionSet](#) class.

C#

```
// Remove FileIOPermission from the permission set.  
ps5.RemovePermission(typeof(FileIOPermission));  
Console.WriteLine("The last permission set after removing FileIOPermission =  
"  
+ ps5.ToString());
```

Remarks

Important

You cannot remove permissions from an unrestricted permission set. The permission set remains unrestricted after you attempt to remove the permission, and an exception is not thrown.

The following C# code attempts to remove the [FileIOPermission](#) from the [FullTrust](#) permission set, but the permission is not removed.

```
PolicyLevel myPol = PolicyLevel.CreateAppDomainLevel();  
PermissionSet myPermSet = myPol.GetNamedPermissionSet("FullTrust");  
myPermSet.RemovePermission(typeof(FileIOPermission));
```

Notes to Inheritors

When you inherit from [PermissionSet](#), you can change the behavior of the [RemovePermission\(Type\)](#) method by overriding the [RemovePermissionImpl\(Type\)](#) method.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.RemovePermissionImpl(Type) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Removes a permission of a certain type from the set.

C#

```
protected virtual System.Security.IPermission? RemovePermissionImpl (Type? permClass);
```

Parameters

permClass [Type](#)

The type of the permission to remove.

Returns

[IPermission](#)

The permission removed from the set.

Exceptions

[InvalidOperationException](#)

The method is called from a [ReadOnlyPermissionSet](#).

Remarks

The [RemovePermissionImpl](#) method is the implementation for the [RemovePermission](#) method.

 **Important**

You cannot remove permissions from an unrestricted permission set. The permission set remains unrestricted after you attempt to remove the permission, and an exception is not thrown.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.RevertAssert Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Causes any previous [Assert\(\)](#) for the current frame to be removed and no longer be in effect.

C#

```
public static void RevertAssert ();
```

Exceptions

[InvalidOperationException](#)

There is no previous [Assert\(\)](#) for the current frame.

Remarks

If there is no [Assert](#) for the current frame, an [ExecutionEngineException](#) is thrown.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2



Collaborate with us on
GitHub



.NET feedback

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET is an open source project.
Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

PermissionSet.Set Permission(IPermission) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Sets a permission to the [PermissionSet](#), replacing any existing permission of the same type.

C#

```
public System.Security.IPermission? SetPermission  
(System.Security.IPermission? perm);
```

Parameters

perm [IPermission](#)

The permission to set.

Returns

[IPermission](#)

The set permission.

Exceptions

[InvalidOperationException](#)

The method is called from a [ReadOnlyPermissionSet](#).

Examples

The following code example shows the use of the [SetPermission](#) method. This code example is part of a larger example provided for the [PermissionSet](#) class.

C#

```
// Change the permission set using SetPermission.  
ps5.SetPermission(new  
EnvironmentPermission(EnvironmentPermissionAccess.AllAccess, "USERNAME"));  
Console.WriteLine("Permission set after SetPermission = " + ps5.ToString());
```

Remarks

This method removes any existing permission object of the same type from the [PermissionSet](#) and replaces it with the `perm` parameter. If a permission that implements [IUnrestrictedPermission](#) is set on a [PermissionSet](#) that is [Unrestricted](#), the resulting [PermissionSet](#) is no longer [Unrestricted](#).

Notes to Inheritors

When you inherit from [PermissionSet](#), you can change the behavior of the [SetPermission\(IPermission\)](#) method by overriding the [SetPermissionImpl\(IPermission\)](#) method.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.SetPermission Impl(IPermission) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Sets a permission to the [PermissionSet](#), replacing any existing permission of the same type.

C#

```
protected virtual System.Security.IPermission? SetPermissionImpl
(System.Security.IPermission? perm);
```

Parameters

perm [IPermission](#)

The permission to set.

Returns

[IPermission](#)

The set permission.

Exceptions

[InvalidOperationException](#)

The method is called from a [ReadOnlyPermissionSet](#).

Remarks

The [SetPermissionImpl](#) method is the implementation for the [SetPermission](#) method.

This method removes any existing permission object of the same type from the [PermissionSet](#) and replaces it with the **perm** parameter. If a permission that implements

`IUnrestrictedPermission` is set on a `PermissionSet` that is `Unrestricted`, the resulting `PermissionSet` is no longer `Unrestricted`.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.ToString Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Returns a string representation of the [PermissionSet](#).

C#

```
public override string ToString();
```

Returns

[String](#)

A representation of the [PermissionSet](#).

Examples

The following code example shows the use of the [ToString](#) method. This code example is part of a larger example provided for the [PermissionSet](#) class.

C#

```
Console.WriteLine("Value of the permission set ToString = \n" +
ps1.ToString());
```

Remarks

The string representation is useful in debugging to see the state of a [PermissionSet](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9

Product	Versions
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.ToXml Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Creates an XML encoding of the security object and its current state.

C#

```
public virtual System.Security.SecurityElement? ToXml();
```

Returns

[SecurityElement](#)

An XML encoding of the security object, including any state information.

Implements

[ToXml\(\)](#)

Examples

The following code example shows the use of the [ToXml](#) method. This code example is part of a larger example provided for the [PermissionSet](#) class.

C#

```
// Display result of ToXml and FromXml operations.
PermissionSet ps6 = new PermissionSet(PermissionState.None);
ps6.FromXml(ps5.ToXml());
Console.WriteLine("Result of ToFromXml = " + ps6.ToString() + "\n");
```

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.Union(PermissionSet) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Creates a [PermissionSet](#) that is the union of the current [PermissionSet](#) and the specified [PermissionSet](#).

C#

```
public System.Security.PermissionSet? Union (System.Security.PermissionSet?  
other);
```

Parameters

other [PermissionSet](#)

The permission set to form the union with the current [PermissionSet](#).

Returns

[PermissionSet](#)

A new permission set that represents the union of the current [PermissionSet](#) and the specified [PermissionSet](#).

Examples

The following code example shows the use of the [Union](#) method. This code example is part of a larger example provided for the [PermissionSet](#) class.

C#

```
// Display the union of two permission sets.  
PermissionSet ps5 = ps3.Union(ps4);  
Console.WriteLine("The union of permission set 3 and permission set 4 = "  
+ ps5.ToString());
```

Remarks

The result of a call to [Union](#) is a [PermissionSet](#) that represents all the operations represented by the current [PermissionSet](#) as well as all the operations represented by the specified [PermissionSet](#). If either set is [Unrestricted](#), the union is [Unrestricted](#) as well.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PermissionSet.IDeserializationCallback.OnDeserialization(Object) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Runs when the entire object graph has been deserialized.

C#

```
void IDeserializationCallback.OnDeserialization (object? sender);
```

Parameters

sender [Object](#)

The object that initiated the callback. The functionality for this parameter is not currently implemented.

Implements

[OnDeserialization\(Object\)](#)

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecureString Class

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.InteropServices.dll

Represents text that should be kept confidential, such as by deleting it from computer memory when no longer needed. This class cannot be inherited.

C#

```
public sealed class SecureString : IDisposable
```

Inheritance Object → SecureString

Implements [IDisposable](#)

Examples

The following example demonstrates how to use a [SecureString](#) to secure a user's password for use as a credential to start a new process.

C#

```
using System;
using System.ComponentModel;
using System.Diagnostics;
using System.Security;

public class Example
{
    public static void Main()
    {
        // Instantiate the secure string.
        SecureString securePwd = new SecureString();
        ConsoleKeyInfo key;

        Console.Write("Enter password: ");
        do {
            key = Console.ReadKey(true);

            // Ignore any key out of range.
        } while (key.KeyChar != 'P' && key.KeyChar != 'p');
    }
}
```

```

        if (((int) key.Key) >= 65 && ((int) key.Key <= 90)) {
            // Append the character to the password.
            securePwd.AppendChar(key.KeyChar);
            Console.Write("*");
        }
        // Exit if Enter key is pressed.
    } while (key.Key != ConsoleKey.Enter);
    Console.WriteLine();

    try {
        Process.Start("Notepad.exe", "MyUser", securePwd, "MYDOMAIN");
    }
    catch (Win32Exception e) {
        Console.WriteLine(e.Message);
    }
    finally {
        securePwd.Dispose();
    }
}
}

```

Remarks

For more information about this API, see [Supplemental API remarks for SecureString](#).

Constructors

[+] Expand table

SecureString()	Initializes a new instance of the SecureString class.
SecureString(Char*, Int32)	<p>Initializes a new instance of the SecureString class from a subarray of Char objects.</p> <p>This constructor is not CLS-compliant. The CLS-compliant alternative is SecureString().</p>

Properties

[+] Expand table

Length	Gets the number of characters in the current secure string.
------------------------	---

Methods

AppendChar(Char)	Appends a character to the end of the current secure string.
Clear()	Deletes the value of the current secure string.
Copy()	Creates a copy of the current secure string.
Dispose()	Releases all resources used by the current SecureString object.
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)
InsertAt(Int32, Char)	Inserts a character in this secure string at the specified index position.
IsReadOnly()	Indicates whether this secure string is marked read-only.
MakeReadOnly()	Makes the text value of this secure string read-only.
MemberwiseClone()	Creates a shallow copy of the current Object . (Inherited from Object)
RemoveAt(Int32)	Removes the character at the specified index position from this secure string.
SetAt(Int32, Char)	Replaces the existing character at the specified index position with another character.
ToString()	Returns a string that represents the current object. (Inherited from Object)

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

See also

- ComVisibleAttribute
- Marshal
- CriticalFinalizerObject
- IDisposable

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

SecureString Constructors

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.InteropServices.dll

Initializes a new instance of the [SecureString](#) class.

Overloads

[+] [Expand table](#)

SecureString()	Initializes a new instance of the SecureString class.
SecureString(Char*, Int32)	Initializes a new instance of the SecureString class from a subarray of Char objects. This constructor is not CLS-compliant. The CLS-compliant alternative is SecureString() .

SecureString()

Initializes a new instance of the [SecureString](#) class.

C#

```
public SecureString();
```

Exceptions

[CryptographicException](#)

An error occurred while protecting or unprotecting the value of this instance.

[NotSupportedException](#)

This operation is not supported on this platform.

Examples

The following example uses the default (or parameterless) constructor to instantiate a new [SecureString](#) object. It then calls the [AppendChar](#) method to add an array of characters to it.

```
C#  
  
using System;  
using System.Security;  
  
public class Example  
{  
    public static void Main()  
    {  
        // Define the string value to assign to a new secure string.  
        char[] chars = { 't', 'e', 's', 't' };  
        // Instantiate the secure string.  
        SecureString testString = new SecureString();  
        // Assign the character array to the secure string.  
        foreach (char ch in chars)  
            testString.AppendChar(ch);  
        // Display secure string length.  
        Console.WriteLine("The length of the string is {0} characters.",  
                          testString.Length);  
        testString.Dispose();  
    }  
}  
// The example displays the following output:  
//      The length of the string is 4 characters.
```

The following example creates a [SecureString](#) object from the value of a [String](#) object.

```
C#  
  
using System;  
using System.Security;  
  
public class Example  
{  
    public static void Main()  
    {  
        // Define the string value to be assigned to the secure string.  
        string initString = "TestString";  
        // Instantiate the secure string.  
        SecureString testString = new SecureString();  
        // Use the AppendChar method to add each char value to the secure  
        // string.  
        foreach (char ch in initString)  
            testString.AppendChar(ch);  
  
        // Display secure string length.
```

```

        Console.WriteLine("The length of the string is {0} characters.",
                           testString.Length);
        testString.Dispose();
    }
}
// The example displays the following output:
//      The length of the string is 10 characters.

```

Applies to

- ▼ .NET 9 and other versions

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

SecureString(Char*, Int32)

Important

This API is not CLS-compliant.

Initializes a new instance of the [SecureString](#) class from a subarray of [Char](#) objects.

This constructor is not CLS-compliant. The CLS-compliant alternative is [SecureString\(\)](#).

C#

```
[System.CLSCompliant(false)]
public SecureString (char* value, int length);
```

Parameters

value [Char*](#)

A pointer to an array of [Char](#) objects.

length Int32

The number of elements of `value` to include in the new instance.

Attributes [CLSCCompliantAttribute](#)

Exceptions

[ArgumentNullException](#)

`value` is `null`.

[ArgumentOutOfRangeException](#)

`length` is less than zero or greater than 65,536.

[CryptographicException](#)

An error occurred while protecting or unprotecting the value of this secure string.

[NotSupportedException](#)

This operation is not supported on this platform.

Examples

The following example instantiates a new [SecureString](#) object by passing its constructor a pointer to a character array.

C#

```
using System;
using System.Security;

public class Example
{
    unsafe public static void Main()
    {
        SecureString testString;
        // Define the string value to assign to a new secure string.
        char[] chars = { 't', 'e', 's', 't' };

        // Instantiate a new secure string.
        fixed(char* pChars = chars)
        {
            testString = new SecureString(pChars, chars.Length);
        }
        // Display secure string length.
        Console.WriteLine("The length of the string is {0} characters.",
                          testString.Length);
        testString.Dispose();
    }
}
```

```
    }
}

// The example displays the following output:
//      The length of the string is 4 characters.
```

Remarks

This constructor initializes the new [SecureString](#) object to the number of characters in `value` specified by `length`; the value of the instance is then encrypted.

In C#, this constructor is defined only in the context of unsafe code.

Applies to

▼ .NET 9 and other versions

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecureString.Length Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.InteropServices.dll

Gets the number of characters in the current secure string.

C#

```
public int Length { get; }
```

Property Value

[Int32](#)

The number of [Char](#) objects in this secure string.

Exceptions

[ObjectDisposedException](#)

This secure string has already been disposed.

Remarks

The [Length](#) property returns the number of [Char](#) objects in this instance, not the number of Unicode characters. A Unicode character might be represented by more than one [Char](#) object.

The maximum length of a [SecureString](#) instance is 65,536 characters.

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1

Product	Versions
.NET Standard	2.0, 2.1

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 .NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecureString.AppendChar(Char) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.InteropServices.dll

Appends a character to the end of the current secure string.

C#

```
public void AppendChar (char c);
```

Parameters

c [Char](#)

A character to append to this secure string.

Exceptions

[ObjectDisposedException](#)

This secure string has already been disposed.

[InvalidOperationException](#)

This secure string is read-only.

[ArgumentOutOfRangeException](#)

Performing this operation would make the length of this secure string greater than 65,536 characters.

[CryptographicException](#)

An error occurred while protecting or unprotecting the value of this secure string.

Examples

The following example demonstrates how the [AppendChar](#), [InsertAt](#), [RemoveAt](#), [SetAt](#), and [Clear](#) methods affect the value of a [SecureString](#) object.

C#

```
using System;
using System.Security;

class Example
{
    public static void Main()
    {
        string msg = "The current length of the SecureString object: {0}\n";
        Console.WriteLine("1) Instantiate the SecureString object.");
        SecureString ss = new SecureString();
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("2) Append 'a' to the value.");
        ss.AppendChar('a');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("3) Append 'X' to the value.");
        ss.AppendChar('X');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("4) Append 'c' to the value.");
        ss.AppendChar('c');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("5) Insert 'd' at the end of the value.");
        ss.InsertAt(ss.Length, 'd');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("6) Remove the last character ('d') from the
value.");
        ss.RemoveAt(3);
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("7) Set the second character of the value to
'b'.");
        ss.SetAt(1, 'b');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("8) Delete the value of the SecureString object:");
        ss.Clear();
        Console.WriteLine(msg, ss.Length);

        ss.Dispose();
    }
}

// The example displays the following output:
//      1) Instantiate the SecureString object.
//      The current length of the SecureString object: 0
//
//      2) Append 'a' to the value.
//      The current length of the SecureString object: 1
//
```

```

//      3) Append 'X' to the value.
//      The current length of the SecureString object: 2
//
//      4) Append 'c' to the value.
//      The current length of the SecureString object: 3
//
//      5) Insert 'd' at the end of the value.
//      The current length of the SecureString object: 4
//
//      6) Remove the last character ('d') from the value.
//      The current length of the SecureString object: 3
//
//      7) Set the second character of the value to 'b'.
//      The current length of the SecureString object: 3
//
//      8) Delete the value of the SecureString object:
//      The current length of the SecureString object: 0

```

The following example demonstrates how the [AppendChar](#) and [RemoveAt](#) methods can be used to collect the characters in a password.

C#

```

using System;
using System.Security;

class Example
{
    public static void Main()
    {
        ConsoleKeyInfo cki;
        String m = "\nEnter your password (up to 15 letters, numbers, and
underscores)\n" +
                   "Press BACKSPACE to delete the last character entered. " +
                   "\nPress Enter when done, or ESCAPE to quit:";
        SecureString password = new SecureString();
        int top, left;

        // The Console.TreatControlCAsInput property prevents CTRL+C from
        // ending this example.
        Console.TreatControlCAsInput = true;

        Console.Clear();
        Console.WriteLine(m);

        top = Console.CursorTop;
        left = Console.CursorLeft;

        // Read user input from the console. Store up to 15 letter, digit, or
underscore
        // characters in a SecureString object, or delete a character if the
user enters
        // a backspace. Display an asterisk (*) on the console to represent

```

```

each character
    // that is stored.

    do {
        cki = Console.ReadKey(true);
        if (cki.Key == ConsoleKey.Escape) break;

        if (cki.Key == ConsoleKey.Backspace) {
            if (password.Length > 0) {
                Console.SetCursorPosition(left + password.Length - 1, top);
                Console.Write(' ');
                Console.SetCursorPosition(left + password.Length - 1, top);
                password.RemoveAt(password.Length-1);
            }
        }
        else {
            if ((password.Length < 15) &&
                (Char.IsLetterOrDigit(cki.KeyChar) || cki.KeyChar == '_'))
{
                password.AppendChar(cki.KeyChar);
                Console.SetCursorPosition(left+password.Length-1, top);
                Console.Write('*');
            }
        }
    } while (cki.Key != ConsoleKey.Enter & password.Length < 15);

    // Make the password read-only to prevent modification.
    password.MakeReadOnly();
    // Dispose of the SecureString instance.
    password.Dispose();
}

// This example displays output like the following:
// Enter your password (up to 15 letters, numbers, and underscores)
// Press BACKSPACE to delete the last character entered.
// Press Enter when done, or ESCAPE to quit:
// *****

```

Remarks

If the implementation uses a protection mechanism, such as encryption, the value of this secure string, if any, is unprotected; `c` is appended; then the new value of the secure string is re-protected.

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9

Product	Versions
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

SecureString.Clear Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.InteropServices.dll

Deletes the value of the current secure string.

C#

```
public void Clear();
```

Exceptions

[ObjectDisposedException](#)

This secure string has already been disposed.

[InvalidOperationException](#)

This secure string is read-only.

Examples

The following example demonstrates how the [AppendChar](#), [InsertAt](#), [RemoveAt](#), [SetAt](#), and [Clear](#) methods affect the value of a [SecureString](#) object.

C#

```
using System;
using System.Security;

class Example
{
    public static void Main()
    {
        string msg = "The current length of the SecureString object: {0}\n";
        Console.WriteLine("1) Instantiate the SecureString object.");
        SecureString ss = new SecureString();
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("2) Append 'a' to the value.");
        ss.AppendChar('a');
```

```
Console.WriteLine(msg, ss.Length);

Console.WriteLine("3) Append 'X' to the value.");
ss.AppendChar('X');
Console.WriteLine(msg, ss.Length);

Console.WriteLine("4) Append 'c' to the value.");
ss.AppendChar('c');
Console.WriteLine(msg, ss.Length);

Console.WriteLine("5) Insert 'd' at the end of the value.");
ss.InsertAt(ss.Length, 'd');
Console.WriteLine(msg, ss.Length);

Console.WriteLine("6) Remove the last character ('d') from the
value.");
ss.RemoveAt(3);
Console.WriteLine(msg, ss.Length);

Console.WriteLine("7) Set the second character of the value to
'b'.");
ss.SetAt(1, 'b');
Console.WriteLine(msg, ss.Length);

Console.WriteLine("8) Delete the value of the SecureString object:");
ss.Clear();
Console.WriteLine(msg, ss.Length);

ss.Dispose();
}

}

// The example displays the following output:
// 1) Instantiate the SecureString object.
// The current length of the SecureString object: 0
//
// 2) Append 'a' to the value.
// The current length of the SecureString object: 1
//
// 3) Append 'X' to the value.
// The current length of the SecureString object: 2
//
// 4) Append 'c' to the value.
// The current length of the SecureString object: 3
//
// 5) Insert 'd' at the end of the value.
// The current length of the SecureString object: 4
//
// 6) Remove the last character ('d') from the value.
// The current length of the SecureString object: 3
//
// 7) Set the second character of the value to 'b'.
// The current length of the SecureString object: 3
//
// 8) Delete the value of the SecureString object:
// The current length of the SecureString object: 0
```

Remarks

The computer memory that contains the value of this secure string is zeroed, then the length of the value of this secure string is set to zero.

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

See also

- [Length](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecureString.Copy Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.InteropServices.dll

Creates a copy of the current secure string.

C#

```
public System.Security.SecureString Copy();
```

Returns

[SecureString](#)

A duplicate of this secure string.

Exceptions

[ObjectDisposedException](#)

This secure string has already been disposed.

[CryptographicException](#)

An error occurred while protecting or unprotecting the value of this secure string.

Remarks

If an instance of a [SecureString](#) is marked read-only, the copy of that instance will not be read-only.

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1

Product	Versions
.NET Standard	2.0, 2.1

See also

- [MakeReadOnly\(\)](#)
- [IsReadOnly\(\)](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

SecureString.Dispose Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.InteropServices.dll

Releases all resources used by the current [SecureString](#) object.

C#

```
public void Dispose();
```

Implements

[Dispose\(\)](#)

Remarks

The [Dispose](#) method writes binary zeroes to the allocated memory that contains the value of this [SecureString](#) object, then frees the allocated memory.

For more information, see [Garbage Collection](#).

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on
GitHub



.NET feedback

.NET is an open source project.
Select a link to provide feedback:

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecureString.InsertAt(Int32, Char) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.InteropServices.dll

Inserts a character in this secure string at the specified index position.

C#

```
public void InsertAt (int index, char c);
```

Parameters

index [Int32](#)

The index position where parameter **c** is inserted.

c [Char](#)

The character to insert.

Exceptions

[ObjectDisposedException](#)

This secure string has already been disposed.

[InvalidOperationException](#)

This secure string is read-only.

[ArgumentOutOfRangeException](#)

index is less than zero, or greater than the length of this secure string.

-or-

Performing this operation would make the length of this secure string greater than 65,536 characters.

CryptographicException

An error occurred while protecting or unprotecting the value of this secure string.

Examples

The following example demonstrates how the [AppendChar](#), [InsertAt](#), [RemoveAt](#), [SetAt](#), and [Clear](#) methods affect the value of a [SecureString](#) object.

C#

```
using System;
using System.Security;

class Example
{
    public static void Main()
    {
        string msg = "The current length of the SecureString object: {0}\n";
        Console.WriteLine("1) Instantiate the SecureString object.");
        SecureString ss = new SecureString();
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("2) Append 'a' to the value.");
        ss.AppendChar('a');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("3) Append 'X' to the value.");
        ss.AppendChar('X');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("4) Append 'c' to the value.");
        ss.AppendChar('c');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("5) Insert 'd' at the end of the value.");
        ss.InsertAt(ss.Length, 'd');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("6) Remove the last character ('d') from the
value.");
        ss.RemoveAt(3);
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("7) Set the second character of the value to
'b'.");
        ss.SetAt(1, 'b');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("8) Delete the value of the SecureString object:");
        ss.Clear();
        Console.WriteLine(msg, ss.Length);
```

```

        ss.Dispose();
    }
}

// The example displays the following output:
//      1) Instantiate the SecureString object.
//          The current length of the SecureString object: 0
//
//      2) Append 'a' to the value.
//          The current length of the SecureString object: 1
//
//      3) Append 'X' to the value.
//          The current length of the SecureString object: 2
//
//      4) Append 'c' to the value.
//          The current length of the SecureString object: 3
//
//      5) Insert 'd' at the end of the value.
//          The current length of the SecureString object: 4
//
//      6) Remove the last character ('d') from the value.
//          The current length of the SecureString object: 3
//
//      7) Set the second character of the value to 'b'.
//          The current length of the SecureString object: 3
//
//      8) Delete the value of the SecureString object:
//          The current length of the SecureString object: 0

```

Remarks

The index is zero-based; the first character in this secure string is at index position zero.

If the implementation uses a protection mechanism, such as encryption, the value of the secure string, if any, is unprotected; `c` is inserted at the specified index position; then the new value is re-protected. The [InsertAt](#) method yields the same results as the [AppendChar](#) method, which inserts a character at the end of a secure string, if the `index` parameter of [InsertAt](#) is set to the length of this instance.

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecureString.IsReadOnly Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.InteropServices.dll

Indicates whether this secure string is marked read-only.

C#

```
public bool IsReadOnly();
```

Returns

[Boolean](#)

`true` if this secure string is marked read-only; otherwise, `false`.

Exceptions

[ObjectDisposedException](#)

This secure string has already been disposed.

Remarks

After an instance of [SecureString](#) is marked read-only by the [MakeReadOnly](#) method, any attempt to modify the value of the instance throws an [InvalidOperationException](#). Use the [IsReadOnly](#) method to test whether a [SecureString](#) is read-only before attempting to modify it.

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1

Product	Versions
.NET Standard	2.0, 2.1

See also

- [MakeReadOnly\(\)](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecureString.MakeReadOnly Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.InteropServices.dll

Makes the text value of this secure string read-only.

C#

```
public void MakeReadOnly();
```

Exceptions

[ObjectDisposedException](#)

This secure string has already been disposed.

Examples

The following example demonstrates how the [AppendChar](#) and [RemoveAt](#) methods can be used to collect the characters in a password. After the password is collected, it is made read-only.

C#

```
using System;
using System.Security;

class Example
{
    public static void Main()
    {
        ConsoleKeyInfo cki;
        String m = "\nEnter your password (up to 15 letters, numbers, and
underscores)\n" +
                   "Press BACKSPACE to delete the last character entered. " +
                   "\nEnter when done, or ESCAPE to quit:";
        SecureString password = new SecureString();
        int top, left;

        // The Console.TreatControlCAsInput property prevents CTRL+C from
        // ending this example.
```

```

Console.TreatControlCAsInput = true;

Console.Clear();
Console.WriteLine(m);

top = Console.CursorTop;
left = Console.CursorLeft;

// Read user input from the console. Store up to 15 letter, digit, or
underscore
// characters in a SecureString object, or delete a character if the
user enters
// a backspace. Display an asterisk (*) on the console to represent
each character
// that is stored.

do {
    cki = Console.ReadKey(true);
    if (cki.Key == ConsoleKey.Escape) break;

    if (cki.Key == ConsoleKey.Backspace) {
        if (password.Length > 0) {
            Console.SetCursorPosition(left + password.Length - 1, top);
            Console.Write(' ');
            Console.SetCursorPosition(left + password.Length - 1, top);
            password.RemoveAt(password.Length-1);
        }
    }
    else {
        if ((password.Length < 15) &&
            (Char.IsLetterOrDigit(cki.KeyChar) || cki.KeyChar == '_'))
    {
        password.AppendChar(cki.KeyChar);
        Console.SetCursorPosition(left+password.Length-1, top);
        Console.Write('*');
    }
    }
} while (cki.Key != ConsoleKey.Enter & password.Length < 15);

// Make the password read-only to prevent modification.
password.MakeReadOnly();
// Dispose of the SecureString instance.
password.Dispose();
}

}

// This example displays output like the following:
// Enter your password (up to 15 letters, numbers, and underscores)
// Press BACKSPACE to delete the last character entered.
// Press Enter when done, or ESCAPE to quit:
// *****

```

Remarks

Initialize the text value of an instance of the [SecureString](#) class with the [SecureString](#) constructors, and modify the value with the [Clear](#), [RemoveAt](#), [SetAt](#), [InsertAt](#), and [AppendChar](#) methods.

After you have made your final modifications, use the [MakeReadOnly](#) method to make the value of the instance immutable (read-only). After the value is marked as read-only, any further attempt to modify it throws an [InvalidOperationException](#).

The effect of invoking [MakeReadOnly](#) is permanent because the [SecureString](#) class provides no means to make the secure string modifiable again. Use the [IsReadOnly](#) method to test whether an instance of [SecureString](#) is read-only.

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

See also

- [IsReadOnly\(\)](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecureString.RemoveAt(Int32) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.InteropServices.dll

Removes the character at the specified index position from this secure string.

C#

```
public void RemoveAt (int index);
```

Parameters

index Int32

The index position of a character in this secure string.

Exceptions

[ObjectDisposedException](#)

This secure string has already been disposed.

[InvalidOperationException](#)

This secure string is read-only.

[ArgumentOutOfRangeException](#)

index is less than zero, or greater than or equal to the length of this secure string.

[CryptographicException](#)

An error occurred while protecting or unprotecting the value of this secure string.

Examples

The following example demonstrates how the [AppendChar](#), [InsertAt](#), [RemoveAt](#), [SetAt](#), and [Clear](#) methods affect the value of a [SecureString](#) object.

C#

```
using System;
using System.Security;

class Example
{
    public static void Main()
    {
        string msg = "The current length of the SecureString object: {0}\n";
        Console.WriteLine("1) Instantiate the SecureString object.");
        SecureString ss = new SecureString();
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("2) Append 'a' to the value.");
        ss.AppendChar('a');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("3) Append 'X' to the value.");
        ss.AppendChar('X');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("4) Append 'c' to the value.");
        ss.AppendChar('c');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("5) Insert 'd' at the end of the value.");
        ss.InsertAt(ss.Length, 'd');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("6) Remove the last character ('d') from the
value.");
        ss.RemoveAt(3);
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("7) Set the second character of the value to
'b'.");
        ss.SetAt(1, 'b');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("8) Delete the value of the SecureString object:");
        ss.Clear();
        Console.WriteLine(msg, ss.Length);

        ss.Dispose();
    }
}

// The example displays the following output:
//      1) Instantiate the SecureString object.
//          The current length of the SecureString object: 0
//
//      2) Append 'a' to the value.
//          The current length of the SecureString object: 1
//
//      3) Append 'X' to the value.
//          The current length of the SecureString object: 2
```

```
//  
//      4) Append 'c' to the value.  
//      The current length of the SecureString object: 3  
//  
//      5) Insert 'd' at the end of the value.  
//      The current length of the SecureString object: 4  
//  
//      6) Remove the last character ('d') from the value.  
//      The current length of the SecureString object: 3  
//  
//      7) Set the second character of the value to 'b'.  
//      The current length of the SecureString object: 3  
//  
//      8) Delete the value of the SecureString object:  
//      The current length of the SecureString object: 0
```

The following example demonstrates how the [AppendChar](#) and [RemoveAt](#) methods can be used to collect the characters in a password.

C#

```
using System;  
using System.Security;  
  
class Example  
{  
    public static void Main()  
    {  
        ConsoleKeyInfo cki;  
        String m = "\nEnter your password (up to 15 letters, numbers, and  
underscores)\n" +  
                  "Press BACKSPACE to delete the last character entered. " +  
                  "\nPress Enter when done, or ESCAPE to quit:";  
        SecureString password = new SecureString();  
        int top, left;  
  
        // The Console.TreatControlCAsInput property prevents CTRL+C from  
        // ending this example.  
        Console.TreatControlCAsInput = true;  
  
        Console.Clear();  
        Console.WriteLine(m);  
  
        top = Console.CursorTop;  
        left = Console.CursorLeft;  
  
        // Read user input from the console. Store up to 15 letter, digit, or  
        // underscore  
        // characters in a SecureString object, or delete a character if the  
        user enters  
        // a backspace. Display an asterisk (*) on the console to represent  
        each character  
        // that is stored.
```

```

do {
    cki = Console.ReadKey(true);
    if (cki.Key == ConsoleKey.Escape) break;

    if (cki.Key == ConsoleKey.Backspace) {
        if (password.Length > 0) {
            Console.SetCursorPosition(left + password.Length - 1, top);
            Console.Write(' ');
            Console.SetCursorPosition(left + password.Length - 1, top);
            password.RemoveAt(password.Length-1);
        }
    }
    else {
        if ((password.Length < 15) &&
            (Char.IsLetterOrDigit(cki.KeyChar) || cki.KeyChar == '_'))
    {
        password.AppendChar(cki.KeyChar);
        Console.SetCursorPosition(left+password.Length-1, top);
        Console.Write('*');
    }
    }
} while (cki.Key != ConsoleKey.Enter & password.Length < 15);

// Make the password read-only to prevent modification.
password.MakeReadOnly();
// Dispose of the SecureString instance.
password.Dispose();
}

}

// This example displays output like the following:
// Enter your password (up to 15 letters, numbers, and underscores)
// Press BACKSPACE to delete the last character entered.
// Press Enter when done, or ESCAPE to quit:
// *****

```

Remarks

The index is zero-based; the first character in this instance is at index position zero.

If the implementation uses a protection mechanism, such as encryption, the value of this secure string, if any, is unprotected; the character at the specified index position is removed; then the new value is re-protected.

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9

Product	Versions
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecureString.SetAt(Int32, Char) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.InteropServices.dll

Replaces the existing character at the specified index position with another character.

C#

```
public void SetAt (int index, char c);
```

Parameters

index Int32

The index position of an existing character in this secure string.

c Char

A character that replaces the existing character.

Exceptions

[ObjectDisposedException](#)

This secure string has already been disposed.

[InvalidOperationException](#)

This secure string is read-only.

[ArgumentOutOfRangeException](#)

index is less than zero, or greater than or equal to the length of this secure string.

[CryptographicException](#)

An error occurred while protecting or unprotecting the value of this secure string.

Examples

The following example demonstrates how the [AppendChar](#), [InsertAt](#), [RemoveAt](#), [SetAt](#), and [Clear](#) methods affect the value of a [SecureString](#) object.

C#

```
using System;
using System.Security;

class Example
{
    public static void Main()
    {
        string msg = "The current length of the SecureString object: {0}\n";
        Console.WriteLine("1) Instantiate the SecureString object.");
        SecureString ss = new SecureString();
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("2) Append 'a' to the value.");
        ss.AppendChar('a');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("3) Append 'X' to the value.");
        ss.AppendChar('X');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("4) Append 'c' to the value.");
        ss.AppendChar('c');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("5) Insert 'd' at the end of the value.");
        ss.InsertAt(ss.Length, 'd');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("6) Remove the last character ('d') from the
value.");
        ss.RemoveAt(3);
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("7) Set the second character of the value to
'b'.");
        ss.SetAt(1, 'b');
        Console.WriteLine(msg, ss.Length);

        Console.WriteLine("8) Delete the value of the SecureString object:");
        ss.Clear();
        Console.WriteLine(msg, ss.Length);

        ss.Dispose();
    }
}

// The example displays the following output:
//      1) Instantiate the SecureString object.
//      The current length of the SecureString object: 0
//
```

```

//      2) Append 'a' to the value.
//      The current length of the SecureString object: 1
//
//      3) Append 'X' to the value.
//      The current length of the SecureString object: 2
//
//      4) Append 'c' to the value.
//      The current length of the SecureString object: 3
//
//      5) Insert 'd' at the end of the value.
//      The current length of the SecureString object: 4
//
//      6) Remove the last character ('d') from the value.
//      The current length of the SecureString object: 3
//
//      7) Set the second character of the value to 'b'.
//      The current length of the SecureString object: 3
//
//      8) Delete the value of the SecureString object:
//      The current length of the SecureString object: 0

```

Remarks

The index is zero-based; the first character in this instance is at index position zero.

If the implementation uses a protection mechanism, such as encryption, the value of the secure string, if any, is unprotected; `c` is assigned to the specified index position; then the new value is re-protected.

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on
GitHub

The source for this content can
be found on GitHub, where you
can also create and review



.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

issues and pull requests. For more information, see [our contributor guide](#).

 [Provide product feedback](#)

SecureStringMarshal Class

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.InteropServices.dll

Important

This API is not CLS-compliant.

Provides a collection of methods for allocating unmanaged memory and copying unmanaged memory blocks.

C#

```
public static class SecureStringMarshal
```

Inheritance [Object](#) → [SecureStringMarshal](#)

Methods

[+] [Expand table](#)

SecureStringToCoTaskMemAnsi(SecureString)	Copies the contents of a managed SecureString object to a block of memory allocated from the unmanaged COM task allocator.
SecureStringToCoTaskMemUnicode(SecureString)	Copies the contents of a managed SecureString object to a block of memory allocated from the unmanaged COM task allocator.
SecureStringToGlobalAllocAnsi(SecureString)	Copies the contents of a managed SecureString into unmanaged memory, converting into ANSI format as it copies.
SecureStringToGlobalAllocUnicode(SecureString)	Copies the contents of a managed SecureString object into unmanaged memory.

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecureStringMarshal.SecureStringToCoTaskMemAnsi(SecureString) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.InteropServices.dll

Copies the contents of a managed [SecureString](#) object to a block of memory allocated from the unmanaged COM task allocator.

C#

```
public static IntPtr SecureStringToCoTaskMemAnsi  
(System.Security.SecureString s);
```

Parameters

s [SecureString](#)

The managed object to copy.

Returns

[IntPtr](#)

The address, in unmanaged memory, where the **s** parameter was copied to, or 0 if a null object was supplied.

Exceptions

[ArgumentNullException](#)

The **s** parameter is [null](#).

[OutOfMemoryException](#)

There is insufficient memory available.

Remarks

The `SecureStringToCoTaskMemAnsi` method is useful for custom marshaling or when mixing managed and unmanaged code. Because this method allocates the unmanaged memory required for a string, always free the memory by calling `ZeroFreeCoTaskMemAnsi`. The characters of the string are copied as ANSI characters.

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecureStringMarshal.SecureStringToCoTaskMemUnicode(SecureString) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.InteropServices.dll

Copies the contents of a managed [SecureString](#) object to a block of memory allocated from the unmanaged COM task allocator.

C#

```
public static IntPtr SecureStringToCoTaskMemUnicode  
(System.Security.SecureString s);
```

Parameters

s [SecureString](#)

The managed object to copy.

Returns

[IntPtr](#)

The address, in unmanaged memory, where the **s** parameter was copied to, or 0 if a null object was supplied.

Exceptions

[ArgumentNullException](#)

The **s** parameter is [null](#).

[OutOfMemoryException](#)

There is insufficient memory available.

Remarks

The [SecureStringToCoTaskMemUnicode](#) method is useful for custom marshaling or when mixing managed and unmanaged code. Because this method allocates the unmanaged memory required for a string, always free the memory by calling the [ZeroFreeCoTaskMemUnicode](#) method. The characters of the string are copied as Unicode characters.

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecureStringMarshal.SecureString ToGlobalAllocAnsi(SecureString) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.InteropServices.dll

Copies the contents of a managed [SecureString](#) into unmanaged memory, converting into ANSI format as it copies.

C#

```
public static IntPtr SecureStringToGlobalAllocAnsi  
(System.Security.SecureString s);
```

Parameters

s [SecureString](#)

The managed object to copy.

Returns

[IntPtr](#)

The address, in unmanaged memory, to where the **s** parameter was copied, or 0 if a null object was supplied.

Exceptions

[ArgumentNullException](#)

The **s** parameter is `null`.

[OutOfMemoryException](#)

There is insufficient memory available.

Remarks

The [SecureStringToGlobalAllocAansi](#) method is useful for custom marshaling or when mixing managed and unmanaged code. Because this method allocates the unmanaged memory required for a string, always free the memory by calling the [ZeroFreeGlobalAllocAansi](#) method.

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecureStringMarshal.SecureString ToGlobalAllocUnicode(SecureString) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.InteropServices.dll

Copies the contents of a managed [SecureString](#) object into unmanaged memory.

C#

```
public static IntPtr SecureStringToGlobalAllocUnicode  
(System.Security.SecureString s);
```

Parameters

s [SecureString](#)

The managed object to copy.

Returns

[IntPtr](#)

The address, in unmanaged memory, where **s** was copied, or 0 if **s** is a [SecureString](#) object whose length is 0.

Exceptions

[ArgumentNullException](#)

The **s** parameter is `null`.

[OutOfMemoryException](#)

There is insufficient memory available.

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityCriticalAttribute Class

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Specifies that code or an assembly performs security-critical operations.

C#

```
[System.AttributeUsage(System.AttributeTargets.Assembly |  
    System.AttributeTargets.Class | System.AttributeTargets.Constructor |  
    System.AttributeTargets.Delegate | System.AttributeTargets.Enum |  
    System.AttributeTargets.Field | System.AttributeTargets.Interface |  
    System.AttributeTargets.Method | System.AttributeTargets.Struct,  
    AllowMultiple=false, Inherited=false)]  
public sealed class SecurityCriticalAttribute : Attribute
```

Inheritance Object → [Attribute](#) → SecurityCriticalAttribute

Attributes [AttributeUsageAttribute](#)

Remarks

ⓘ Important

Partially trusted code is no longer supported. This attribute has no effect in .NET Core.

Security-critical operations are actions that affect code access security, such as elevation of privilege through suppression of code access security checks by using the [Assert](#) method, calling unsafe managed code, and so forth. Either the [SecurityCriticalAttribute](#) attribute or the [SecuritySafeCriticalAttribute](#) attribute must be applied to code for the code to perform security-critical operations.

ⓘ Note

The [SecurityCriticalAttribute](#) is equivalent to a link demand for full trust. A type or member marked with the [SecurityCriticalAttribute](#) can be called only by fully trusted code; it does not have to demand specific permissions. It cannot be called by partially trusted code.

Applying the [SecurityCriticalAttribute](#) at the assembly level identifies the assembly as a security-critical assembly. The entire assembly can be identified as critical by setting the scope parameter [SecurityCriticalScope.Everything](#).

Constructors

[+] [Expand table](#)

SecurityCriticalAttribute()	Initializes a new instance of the SecurityCriticalAttribute class.
SecurityCriticalAttribute(SecurityCriticalScope)	Initializes a new instance of the SecurityCriticalAttribute class with the specified scope.

Properties

[+] [Expand table](#)

Scope	Obsolete.
	Gets the scope for the attribute.
TypeId	When implemented in a derived class, gets a unique identifier for this Attribute . (Inherited from Attribute)

Methods

[+] [Expand table](#)

Equals(Object)	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute)
GetHashCode()	Returns the hash code for this instance. (Inherited from Attribute)
GetType()	Gets the Type of the current instance.

(Inherited from [Object](#))

IsDefaultAttribute()	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute)
Match(Object)	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute)
MemberwiseClone()	Creates a shallow copy of the current Object . (Inherited from Object)
ToString()	Returns a string that represents the current object. (Inherited from Object)

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0

See also

- [SecurityTransparentAttribute](#)
- [SecuritySafeCriticalAttribute](#)
- [Security-Transparent Code, Level 1](#)
- [Security-Transparent Code, Level 2](#)

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

more information, see [our contributor guide](#).

SecurityCriticalAttribute Constructors

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Initializes a new instance of the [SecurityCriticalAttribute](#) class.

Overloads

[+] Expand table

SecurityCriticalAttribute()	Initializes a new instance of the SecurityCriticalAttribute class.
SecurityCriticalAttribute(SecurityCriticalScope)	Initializes a new instance of the SecurityCriticalAttribute class with the specified scope.

SecurityCriticalAttribute()

Initializes a new instance of the [SecurityCriticalAttribute](#) class.

C#

```
public SecurityCriticalAttribute();
```

Applies to

▼ .NET 9 and other versions

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1

Product	Versions
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0

SecurityCriticalAttribute(SecurityCriticalScope)

Initializes a new instance of the [SecurityCriticalAttribute](#) class with the specified scope.

C#

```
public SecurityCriticalAttribute (System.Security.SecurityCriticalScope  
scope);
```

Parameters

scope [SecurityCriticalScope](#)

One of the enumeration values that specifies the scope of the attribute.

Remarks

This constructor is provided for compatibility with the .NET Framework version 2.0 transparency model. It does not apply to the .NET Framework 4. For more information, see [Security-Transparent Code, Level 2](#).

Applies to

▼ .NET 9 and other versions

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityCriticalAttribute.Scope Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

⊗ Caution

SecurityCriticalScope is only used for .NET 2.0 transparency compatibility.

Gets the scope for the attribute.

C#

```
[System.Obsolete("SecurityCriticalScope is only used for .NET 2.0
transparency compatibility.")]
public System.Security.SecurityCriticalScope Scope { get; }
```

Property Value

[SecurityCriticalScope](#)

One of the enumeration values that specifies the scope of the attribute. The default is [Explicit](#), which indicates that the attribute applies only to the immediate target.

Attributes [ObsoleteAttribute](#)

Applies to

Product	Versions (<i>Obsolete</i>)
.NET	(Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9)
.NET Framework	2.0, 3.0, 3.5 (4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1)
.NET Standard	(2.0, 2.1)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityCriticalScope Enum

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

⊗ Caution

SecurityCriticalScope is only used for .NET 2.0 transparency compatibility.

Specifies the scope of a [SecurityCriticalAttribute](#).

C#

```
[System.Obsolete("SecurityCriticalScope is only used for .NET 2.0
transparency compatibility.")]
public enum SecurityCriticalScope
```

Inheritance Object → [ValueType](#) → [Enum](#) → SecurityCriticalScope

Attributes [ObsoleteAttribute](#)

Fields

[] [Expand table](#)

Everything	1	The attribute applies to all code that follows it.
Explicit	0	The attribute applies only to the immediate target.

Remarks

Security-critical code is any code that protects sensitive resources and exposes brokered access to the resource to untrusted or partially trusted code by performing the necessary elevation of privilege operations.

Applies to

Product	Versions (<i>Obsolete</i>)
.NET	(Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9)
.NET Framework	2.0, 3.0, 3.5 (4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1)
.NET Standard	(2.0, 2.1)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project.
Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

SecurityElement Class

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Represents the XML object model for encoding security objects. This class cannot be inherited.

C#

```
public sealed class SecurityElement
```

Inheritance Object → SecurityElement

Examples

The following example shows the use of members of the [SecurityElement](#) class.

C#

```
using System;
using System.Security;
using System.Collections;

class SecurityElementMembers
{
    [STAThread]
    static void Main(string[] args)
    {
        SecurityElement xmlRootElement =
            new SecurityElement("RootTag", "XML security tree");

        AddAttribute(xmlRootElement, "creationdate", DateTime.Now.ToString());
        AddChildElement(xmlRootElement, "destroytime",
            DateTime.Now.AddSeconds(1.0).ToString());

        SecurityElement windowsRoleElement =
            new SecurityElement("WindowsMembership.WindowsRole");

        windowsRoleElement.AddAttribute("version", "1.00");

        // Add a child element and a creationdate attribute.
```

```

        AddChildElement(windowsRoleElement, "BabyElement",
                         "This is a child element");
        AddAttribute(windowsRoleElement, "creationdate",
                      DateTime.Now.ToString());

        xmlRootElement.AddChild(windowsRoleElement);

        CompareAttributes(xmlRootElement, "creationdate");
        ConvertToHashTable(xmlRootElement);

        DisplaySummary(xmlRootElement);

        // Determine if the security element is too old to keep.
        xmlRootElement = DestroyTree(xmlRootElement);
        if (xmlRootElement != null)
        {
            string elementInXml = xmlRootElement.ToString();
            Console.WriteLine(elementInXml);
        }

        Console.WriteLine("This sample completed successfully; " +
                          "press Enter to exit.");
        Console.ReadLine();
    }

    // Add an attribute to the specified security element.
    private static SecurityElement AddAttribute(
        SecurityElement xmlElement,
        string attributeName,
        string attributeValue)
    {
        if (xmlElement != null)
        {
            // Verify that the attribute name and value are valid XML
            formats.
            if (SecurityElement.IsValidAttributeName(attributeName) &&
                SecurityElement.IsValidAttributeValue(attributeValue))
            {
                // Add the attribute to the security element.
                xmlElement.AddAttribute(attributeName, attributeValue);
            }
        }
        return xmlElement;
    }

    // Add a child element to the specified security element.
    private static SecurityElement AddChildElement(
        SecurityElement parentElement,
        string tagName,
        string tagText)
    {
        if (parentElement != null)
        {
            // Ensure that the tag text is in valid XML format.
            if (!SecurityElement.IsValidText(tagText))

```

```

    {
        // Replace invalid text with valid XML text
        // to enforce proper XML formatting.
        tagText = SecurityElement.Escape(tagText);
    }

    // Determine whether the tag is in valid XML format.
    if (SecurityElement.IsValidTag(tagName))
    {
        SecurityElement childElement;
        childElement = parentElement.SearchForChildByTag(tagName);

        if (childElement != null)
        {
            String elementText;
            elementText = parentElement.SearchForTextOfTag(tagName);

            if (!elementText.Equals(tagText))
            {
                // Add child element to the parent security element.
                parentElement.AddChild(
                    new SecurityElement(tagName, tagText));
            }
        }
        else
        {
            // Add child element to the parent security element.
            parentElement.AddChild(
                new SecurityElement(tagName, tagText));
        }
    }
    return parentElement;
}

// Create and display a summary sentence
// about the specified security element.
private static void DisplaySummary(SecurityElement xmlElement)
{
    // Retrieve tag name for the security element.
    string xmlTreeName = xmlElement.Tag.ToString();

    // Retrieve tag text for the security element.
    string xmlTreeDescription = xmlElement.Text;

    // Retrieve value of the creationdate attribute.
    string xmlCreationDate = xmlElement.Attribute("creationdate");

    // Retrieve the number of children under the security element.
    string childrenCount = xmlElement.Children.Count.ToString();

    string outputMessage = "The security XML tree named " + xmlTreeName;
    outputMessage += "(" + xmlTreeDescription + ")";
    outputMessage += " was created on " + xmlCreationDate + " and ";
    outputMessage += "contains " + childrenCount + " child elements.";
}

```

```

        Console.WriteLine(outputMessage);
    }

    // Compare the first two occurrences of an attribute
    // in the specified security element.
    private static void CompareAttributes(
        SecurityElement xmlElement, string attributeName)
{
    // Create a hash table containing the security element's attributes.
    Hashtable attributeKeys = xmlElement.Attributes;
    string attributeValue = attributeKeys[attributeName].ToString();

    foreach(SecurityElement xmlChild in xmlElement.Children)
    {
        if (attributeValue.Equals(xmlChild.Attribute(attributeName)))
        {
            // The security elements were created at the exact same
            time.
        }
    }
}

// Convert the contents of the specified security element
// to hash codes stored in a hash table.
private static void ConvertToHashTable(SecurityElement xmlElement)
{
    // Create a hash table to hold hash codes of the security elements.
    Hashtable xmlAsHash = new Hashtable();
    int rootIndex = xmlElement.GetHashCode();
    xmlAsHash.Add(rootIndex, "root");

    int parentNum = 0;

    foreach(SecurityElement xmlParent in xmlElement.Children)
    {
        parentNum++;
        xmlAsHash.Add(xmlParent.GetHashCode(), "parent" + parentNum);
        if ((xmlParent.Children != null) &&
            (xmlParent.Children.Count > 0))
        {
            int childNum = 0;
            foreach(SecurityElement xmlChild in xmlParent.Children)
            {
                childNum++;
                xmlAsHash.Add(xmlChild.GetHashCode(), "child" +
                childNum);
            }
        }
    }
}

// Delete the specified security element if the current time is past
// the time stored in the destroytime tag.
private static SecurityElement DestroyTree(SecurityElement xmlElement)

```

```

{
    SecurityElement localXmlElement = xmlElement;
    SecurityElement destroyElement =
        localXmlElement.SearchForChildByTag("destroytime");

    // Verify that a destroytime tag exists.
    if (localXmlElement.SearchForChildByTag("destroytime") != null)
    {
        // Retrieve the destroytime text to get the time
        // the tree can be destroyed.
        string storedDestroyTime =
            localXmlElement.SearchForTextOfTag("destroytime");

        DateTime destroyTime = DateTime.Parse(storedDestroyTime);
        if (DateTime.Now > destroyTime)
        {
            localXmlElement = null;
            Console.WriteLine("The XML security tree has been
deleted.");
        }
    }

    // Verify that xmlElement is of type SecurityElement.
    if (xmlElement.GetType().Equals(
        typeof(System.Security.SecurityElement)))
    {
        // Determine whether the localXmlElement object
        // differs from xmlElement.
        if (xmlElement.Equals(localXmlElement))
        {
            // Verify that the tags, attributes and children of the
            // two security elements are identical.
            if (xmlElement.Equal(localXmlElement))
            {
                // Return the original security element.
                return xmlElement;
            }
        }
    }
}

// Return the modified security element.
return localXmlElement;
}
}

// This sample produces the following output:
//
// The security XML tree named RootTag(XML security tree)
// was created on 2/23/2004 1:23:00 PM and contains 2 child elements.
//<RootTag creationdate="2/23/2004 1:23:00 PM">XML security tree
//  <destroytime>2/23/2004 1:23:01 PM</destroytime>
//  <WindowsMembership.WindowsRole version="1.00"
//                                creationdate="2/23/2004 1:23:00 PM">
//    <BabyElement>This is a child element.</BabyElement>

```

```
//  
//This sample completed successfully; press Exit to continue.
```

Remarks

This class is intended to be a lightweight implementation of a simple XML object model for use within the security system, and not for use as a general XML object model. This documentation assumes a basic knowledge of XML.

The simple XML object model for an element consists of the following parts:

- The tag is the element name.
- The attributes are zero or more name/value attribute pairs on the element.
- The children are zero or more elements nested within `<tag>` and `</tag>`.

It is strongly suggested that attribute based XML representation is used to express security elements and their values. This means properties of an element are expressed as attributes and property values are expressed as attribute values. Avoid nesting text within tags. For any `<tag>text</tag>` representation a representation of type `<tag value="text"/>` is usually available. Using this attribute-based XML representation increases readability and allows easy WMI portability of the resulting XML serialization.

An attribute name must be one character or longer, and cannot be `null`. If element-based value representation is used, elements with a text string that is `null` are represented in the `<tag/>` form; otherwise, text is delimited by the `<tag>` and `</tag>` tokens. Both forms can be combined with attributes, which are shown if present.

The tags, attributes, and text of elements, if present, are always case-sensitive. The XML form contains quotations and escapes where necessary. String values that include characters invalid for use in XML result in an [ArgumentException](#). These rules apply to all properties and methods.

ⓘ Note

For performance reasons, character validity is only checked when the element is encoded into XML text form, and not on every set of a property or method. Static methods allow explicit checking where needed.

Constructors

[\[\] Expand table](#)

SecurityElement(String)	Initializes a new instance of the SecurityElement class with the specified tag.
SecurityElement(String, String)	Initializes a new instance of the SecurityElement class with the specified tag and text.

Properties

[\[\] Expand table](#)

Attributes	Gets or sets the attributes of an XML element as name/value pairs.
Children	Gets or sets the array of child elements of the XML element.
Tag	Gets or sets the tag name of an XML element.
Text	Gets or sets the text within an XML element.

Methods

[\[\] Expand table](#)

AddAttribute(String, String)	Adds a name/value attribute to an XML element.
AddChild(Security Element)	Adds a child element to the XML element.
Attribute(String)	Finds an attribute by name in an XML element.
Copy()	Creates and returns an identical copy of the current SecurityElement object.
Equal(SecurityElement)	Compares two XML element objects for equality.
Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object)
Escape(String)	Replaces invalid XML characters in a string with their valid XML equivalent.
FromString(String)	Creates a security element from an XML-encoded string.
GetHashCode()	Serves as the default hash function. (Inherited from Object)

GetType()	Gets the Type of the current instance. (Inherited from Object)
IsValidAttribute Name(String)	Determines whether a string is a valid attribute name.
IsValidAttribute Value(String)	Determines whether a string is a valid attribute value.
IsValidTag(String)	Determines whether a string is a valid tag.
IsValidText(String)	Determines whether a string is valid as text within an XML element.
MemberwiseClone()	Creates a shallow copy of the current Object . (Inherited from Object)
SearchForChild ByTag(String)	Finds a child by its tag name.
SearchForText OfTag(String)	Finds a child by its tag name and returns the contained text.
ToString()	Produces a string representation of an XML element and its constituent attributes, child elements, and text.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

[Open a documentation issue](#)

[Provide product feedback](#)

SecurityElement Constructors

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Initializes a new instance of the [SecurityElement](#) class.

Overloads

[+] Expand table

SecurityElement(String)	Initializes a new instance of the SecurityElement class with the specified tag.
SecurityElement(String, String)	Initializes a new instance of the SecurityElement class with the specified tag and text.

SecurityElement(String)

Initializes a new instance of the [SecurityElement](#) class with the specified tag.

C#

```
public SecurityElement (string tag);
```

Parameters

tag [String](#)

The tag name of an XML element.

Exceptions

[ArgumentNullException](#)

The `tag` parameter is `null`.

[ArgumentException](#)

The `tag` parameter is invalid in XML.

Examples

The following code shows the use of the [SecurityElement](#) constructor to create a new [SecurityElement](#) object. This code example is part of a larger example provided for the [SecurityElement](#) class.

C#

```
SecurityElement windowsRoleElement =
    new SecurityElement("WindowsMembership.WindowsRole");
```

Remarks

The `tag` parameter must consist of a valid XML tag name. Use [Escape](#) to remove invalid characters from the string.

Applies to

- ▼ .NET 9 and other versions

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

SecurityElement(String, String)

Initializes a new instance of the [SecurityElement](#) class with the specified tag and text.

C#

```
public SecurityElement (string tag, string? text);
```

Parameters

tag [String](#)

The tag name of the XML element.

text `String`

The text content within the element.

Exceptions

[ArgumentNullException](#)

The `tag` parameter is `null`.

[ArgumentException](#)

The `tag` parameter or `text` parameter is invalid in XML.

Remarks

If the `text` parameter is `null` this constructor produces an element identical to the parameterless constructor.

Applies to

▼ .NET 9 and other versions

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 .NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityElement.Attributes Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets or sets the attributes of an XML element as name/value pairs.

C#

```
public System.Collections.Hashtable? Attributes { get; set; }
```

Property Value

[Hashtable](#)

The [Hashtable](#) object for the attribute values of the XML element.

Exceptions

[InvalidOperationException](#)

The name or value of the [Hashtable](#) object is invalid.

[ArgumentException](#)

The name is not a valid XML attribute name.

Examples

The following code shows the use of the [Attributes](#) property to get an attribute of an XML element. This code example is part of a larger example provided for the [SecurityElement](#) class.

C#

```
Hashtable attributeKeys = xmlElement.Attributes;
string attributeName = attributeKeys[attributeName].ToString();
```

Remarks

Each attribute is stored in the [Hashtable](#) as a name/value pair.

Names and values in attributes should contain only valid XML attribute characters. Use [Escape](#) to remove invalid characters from the string.

There is no support for quoted strings, so strings for name/value pairs should not contain quotes or other characters requiring quoting.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 [Collaborate with us on GitHub](#)

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityElement.Children Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets or sets the array of child elements of the XML element.

C#

```
public System.Collections.ArrayList? Children { get; set; }
```

Property Value

[ArrayList](#)

The ordered child elements of the XML element as security elements.

Exceptions

[ArgumentException](#)

A child of the XML parent node is `null`.

Examples

The following code shows the use of the [Children](#) property to get the array of child elements of the XML element. This code example is part of a larger example provided for the [SecurityElement](#) class.

C#

```
string childrenCount = xmlElement.Children.Count.ToString();
```

Remarks

If a [SecurityElement](#) contains both [Text](#) and [Children](#), [Text](#) will appear first.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project.
Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

SecurityElement.Tag Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets or sets the tag name of an XML element.

C#

```
public string Tag { get; set; }
```

Property Value

[String](#)

The tag name of an XML element.

Exceptions

[ArgumentNullException](#)

The tag is `null`.

[ArgumentException](#)

The tag is not valid in XML.

Examples

The following code shows the use of the `Tag` property to get the tag name of an XML element. This code example is part of a larger example provided for the [SecurityElement](#) class.

C#

```
string xmlTreeName = xmlElement.Tag.ToString();
```

Remarks

In XML, the tag appears in the script as appears below:

```
<tag attributes>text</tag>
```

If this element has child elements, the children will replace `text`.

Assign only valid XML tag strings to this property. Use [Escape](#) to remove invalid characters from the string.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityElement.Text Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets or sets the text within an XML element.

C#

```
public string? Text { get; set; }
```

Property Value

[String](#)

The value of the text within an XML element.

Exceptions

[ArgumentException](#)

The text is not valid in XML.

Examples

The following code shows the use of the [Text](#) property to get the text of an XML element. This code example is part of a larger example provided for the [SecurityElement](#) class.

C#

```
string xmlTreeDescription = xmlElement.Text;
```

Remarks

The text should not contain XML special characters. Use [Escape](#) to remove invalid characters from the string.

If a `SecurityElement` contains both `Text` and `Children`, `Text` will appear first.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityElement.AddAttribute(String, String) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Adds a name/value attribute to an XML element.

C#

```
public void AddAttribute (string name, string value);
```

Parameters

name [String](#)

The name of the attribute.

value [String](#)

The value of the attribute.

Exceptions

[ArgumentNullException](#)

The `name` parameter or `value` parameter is `null`.

[ArgumentException](#)

The `name` parameter or `value` parameter is invalid in XML.

-or-

An attribute with the name specified by the `name` parameter already exists.

Examples

The following code shows the use of the [AddAttribute](#) method to add a name/value attribute to an XML element. This code example is part of a larger example provided for

the [SecurityElement](#) class.

C#

```
windowsRoleElement.AddAttribute("version", "1.00");
```

Remarks

Names and values in attributes should only contain valid XML attribute characters. XML attribute names must be unique. Use [Escape](#) to remove invalid characters from the string.

There is no support for quoted strings, so strings for name/value pairs should not contain quotes or other characters requiring quoting.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityElement.AddChild(SecurityElement) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Adds a child element to the XML element.

C#

```
public void AddChild (System.Security.SecurityElement child);
```

Parameters

child [SecurityElement](#)

The child element to add.

Exceptions

[ArgumentNullException](#)

The `child` parameter is `null`.

Examples

The following code shows the use of the [AddChild](#) method to add a child element to the XML element. This code example is part of a larger example provided for the [SecurityElement](#) class.

C#

```
xmlRootElement.AddChild(windowsRoleElement);
```

Remarks

The child element is added following any previously existing child elements.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project.
Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

SecurityElement.Attribute(String) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Finds an attribute by name in an XML element.

C#

```
public string? Attribute (string name);
```

Parameters

name [String](#)

The name of the attribute for which to search.

Returns

[String](#)

The value associated with the named attribute, or `null` if no attribute with `name` exists.

Exceptions

[ArgumentNullException](#)

The `name` parameter is `null`.

Examples

The following code shows the use of the [Attribute](#) method to find an attribute by name. This code example is part of a larger example provided for the [SecurityElement](#) class.

C#

```
string xmlCreationDate = xmlElement.Attribute("creationdate");
```

Remarks

With XML as follows, `Attribute("B")` would return "456".

```
<thetag A="123" B="456" C="789">text</thetag>
```

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityElement.Copy Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Creates and returns an identical copy of the current [SecurityElement](#) object.

C#

```
public System.Security.SecurityElement Copy();
```

Returns

[SecurityElement](#)

A copy of the current [SecurityElement](#) object.

Remarks

The copy includes both the [Children](#) and [Attributes](#) properties.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on
GitHub

The source for this content can
be found on GitHub, where you
can also create and review



.NET feedback

.NET is an open source project.
Select a link to provide feedback:

issues and pull requests. For more information, see [our contributor guide](#).

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityElement.Equal(SecurityElement) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Compares two XML element objects for equality.

C#

```
public bool Equal (System.Security.SecurityElement? other);
```

Parameters

other [SecurityElement](#)

An XML element object to which to compare the current XML element object.

Returns

[Boolean](#)

`true` if the tag, attribute names and values, child elements, and text fields in the current XML element are identical to their counterparts in the `other` parameter; otherwise, `false`.

Examples

The following code shows the use of the `Equal` method to compare two XML elements. This code example is part of a larger example provided for the [SecurityElement](#) class.

C#

```
if (xmlElement.Equal(localXmlElement))
```

Remarks

If there are child elements, comparison extends recursively to them.

There is no support for comparing different XML representations of the same characters.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityElement.Escape(String) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Replaces invalid XML characters in a string with their valid XML equivalent.

C#

```
public static string? Escape (string? str);
```

Parameters

str [String](#)

The string within which to escape invalid characters.

Returns

[String](#)

The input string with invalid characters replaced.

Examples

The following code shows the use of the [Escape](#) method to replace invalid XML characters in a string with their valid XML equivalent. This code example is part of a larger example provided for the [SecurityElement](#) class.

C#

```
tagText = SecurityElement.Escape(tagText);
```

Remarks

Use this method to replace invalid characters in a string before using the string in a [SecurityElement](#). If invalid characters are used in a [SecurityElement](#) without being

escaped, an [ArgumentException](#) is thrown.

The following table shows the invalid XML characters and their escaped equivalents.

[Expand table](#)

Invalid XML character	Replaced with
<	<
>	>
"	"
'	'
&	&

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityElement.FromString(String) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Creates a security element from an XML-encoded string.

C#

```
public static System.Security.SecurityElement? FromString (string xml);
```

Parameters

`xml` [String](#)

The XML-encoded string from which to create the security element.

Returns

[SecurityElement](#)

A [SecurityElement](#) created from the XML.

Exceptions

[XmlSyntaxException](#)

`xml` contains one or more single quotation mark characters.

[ArgumentNullException](#)

`xml` is `null`.

Remarks

Do not use single quotation marks in the XML string; instead, use escaped double quotation marks. For example, instead of "`<value name='Company'>Microsoft</value>`" use "`<value name=\"Company\">Microsoft</value>`".

Using single quotation marks can result in either an exception being thrown or, in some cases, the single quotation marks being treated as text in the string.

Evidence based security model is not supported on .NET Core and this method will return `null`.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityElement.IsValidAttributeName(String) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Determines whether a string is a valid attribute name.

C#

```
public static bool IsValidAttributeName (string? name);
```

Parameters

name [String](#)

The attribute name to test for validity.

Returns

[Boolean](#)

`true` if the `name` parameter is a valid XML attribute name; otherwise, `false`.

Examples

The following code shows the use of the [IsValidAttributeName](#) method to determine whether a string is a valid attribute name. This code example is part of a larger example provided for the [SecurityElement](#) class.

C#

```
if (SecurityElement.IsValidAttributeName(attributeName) &&
    SecurityElement.IsValidAttributeValue(attributeValue))
```

Remarks

This method can be used to test an attribute before adding it to a security element.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityElement.IsValidAttribute Value(String) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Determines whether a string is a valid attribute value.

C#

```
public static bool IsValidAttributeValue (string? value);
```

Parameters

value [String](#)

The attribute value to test for validity.

Returns

[Boolean](#)

`true` if the `value` parameter is a valid XML attribute value; otherwise, `false`.

Examples

The following code shows the use of the [IsValidAttributeValue](#) method to determine whether a string is a valid attribute value. This code example is part of a larger example provided for the [SecurityElement](#) class.

C#

```
if (SecurityElement.IsValidAttributeName(attributeName) &&
    SecurityElement.IsValidAttributeValue(attributeValue))
```

Remarks

This method can be used to test an attribute before adding it to a security element.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityElement.IsValidTag(String) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Determines whether a string is a valid tag.

C#

```
public static bool IsValidTag (string? tag);
```

Parameters

tag [String](#)

The tag to test for validity.

Returns

[Boolean](#)

`true` if the `tag` parameter is a valid XML tag; otherwise, `false`.

Examples

The following code shows the use of the `IsValidTag` method to determine whether a string is a valid attribute tag. This code example is part of a larger example provided for the [SecurityElement](#) class.

C#

```
if (SecurityElement.IsValidTag(tagName))
```

Remarks

This can be used to test a [Tag](#) before setting it.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityElement.IsValidText(String) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Determines whether a string is valid as text within an XML element.

C#

```
public static bool IsValidText (string? text);
```

Parameters

text [String](#)

The text to test for validity.

Returns

[Boolean](#)

`true` if the `text` parameter is a valid XML text element; otherwise, `false`.

Examples

The following code shows the use of the `IsValidText` method to determine whether a string is valid as XML element text. This code example is part of a larger example provided for the [SecurityElement](#) class.

C#

```
if (!SecurityElement.IsValidText(tagText))
```

Remarks

This method can be used to test [Text](#) before setting it.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityElement.SearchForChildByTag(String) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Finds a child by its tag name.

C#

```
public System.Security.SecurityElement? SearchForChildByTag (string tag);
```

Parameters

tag [String](#)

The tag for which to search in child elements.

Returns

[SecurityElement](#)

The first child XML element with the specified tag value, or `null` if no child element with `tag` exists.

Exceptions

[ArgumentNullException](#)

The `tag` parameter is `null`.

Examples

The following code shows the use of the [SearchForChildByTag](#) method to find a child by its tag name. This code example is part of a larger example provided for the [SecurityElement](#) class.

C#

```
if (localXmlElement.SearchForChildByTag("destroytime") != null)
```

Remarks

With XML as follows, `SearchForChildByTag("second")` would return the child element

`<second>`.

```
<thetag A="123" B="456" C="789"> <first>text1</first>
    <second>text2</second></thetag>
```

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityElement.SearchForText OfTag(String) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Finds a child by its tag name and returns the contained text.

C#

```
public string? SearchForTextOfTag (string tag);
```

Parameters

tag [String](#)

The tag for which to search in child elements.

Returns

[String](#)

The text contents of the first child element with the specified tag value.

Exceptions

[ArgumentNullException](#)

tag is `null`.

Examples

The following code shows the use of the [SearchForTextOfTag](#) method to find a child by its tag name and return the contained text. This code example is part of a larger example provided for the [SecurityElement](#) class.

C#

```
string storedDestroyTime =  
    localXmlElement.SearchForTextOfTag("destroytime");
```

Remarks

This method is equivalent to the following:

C#

```
string SearchForTextOfTag(string tag)  
{  
    SecurityElement element = this.SearchForChildByTag(tag);  
    return element.Text;  
}
```

With XML as follows, `SearchForTextOfTag("second")` would return "text2".

```
<thetag A="123" B="456" C="789"> <first>text1</first>  
    <second>text2</second></thetag>
```

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityElement.ToString Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Produces a string representation of an XML element and its constituent attributes, child elements, and text.

C#

```
public override string ToString ();
```

Returns

[String](#)

The XML element and its contents.

Remarks

This method is useful in debugging to see the XML representation of the element.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1



Collaborate with us on
GitHub



.NET feedback

.NET is an open source project.
Select a link to provide feedback:

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityException Class

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

The exception that is thrown when a security error is detected.

C#

```
public class SecurityException : SystemException
```

Inheritance Object → Exception → SystemException → SecurityException

Remarks

A [SecurityException](#) exception is thrown when a caller does not have the permissions required to access a resource. The following example instantiates a [PermissionSet](#) object that includes a [UIPermission](#) object to allow access to UI objects and the Clipboard and a [RegistryPermission](#) object to prevent registry access. The call to the [PermissionSet.PermitOnly](#) method means that these permissions will apply regardless of the permissions assigned to the caller. As a result, the attempt to create a registry key throws a [SecurityException](#).

C#

```
using Microsoft.Win32;
using System;
using System.Security;
using System.Security.Permissions;

public class Example
{
    public static void Main()
    {
        PermissionSet perms = new PermissionSet(null);
        perms.AddPermission(new UIPermission(PermissionState.Unrestricted));
        perms.AddPermission(new RegistryPermission(PermissionState.None));
        perms.PermitOnly();

        try {
```

```

        RegistryKey key =
Registry.CurrentUser.CreateSubKey("MyCompany\\Applications");
        Console.WriteLine("Registry key: {0}", key.Name);
    }
    catch (SecurityException e) {
        Console.WriteLine("Security Exception:\n\n{0}", e.Message);
    }
}
// The example displays the following output:
//   Security Exception:
//
//   Request for the permission of type
'System.Security.Permissions.RegistryPermission,
//   mscorlib, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089' failed.

```

[SecurityException](#) uses the HRESULT COR_E_SECURITY, which has the value 0x8013150A.

For a list of the initial property values for an instance of the [SecurityException](#) class, see a specific [SecurityException](#) constructor.

Constructors

[+] [Expand table](#)

SecurityException()	Initializes a new instance of the SecurityException class with default properties.
SecurityException(SerializationInfo, StreamingContext)	Obsolete. Initializes a new instance of the SecurityException class with serialized data.
SecurityException(String)	Initializes a new instance of the SecurityException class with a specified error message.
SecurityException(String, Exception)	Initializes a new instance of the SecurityException class with a specified error message and a reference to the inner exception that is the cause of this exception.
SecurityException(String, Type)	Initializes a new instance of the SecurityException class with a specified error message and the permission type that caused the exception to be thrown.
SecurityException(String, Type, String)	Initializes a new instance of the SecurityException class with a specified error message, the permission type that caused the exception to be thrown, and the permission state.

Properties

[Expand table](#)

Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception)
Demanded	Gets or sets the demanded security permission, permission set, or permission set collection that failed.
DenySetInstance	Gets or sets the denied security permission, permission set, or permission set collection that caused a demand to fail.
FailedAssemblyInfo	Gets or sets information about the failed assembly.
GrantedSet	Gets or sets the granted permission set of the assembly that caused the SecurityException .
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception)
Message	Gets a message that describes the current exception. (Inherited from Exception)
Method	Gets or sets the information about the method associated with the exception.
PermissionState	Gets or sets the state of the permission that threw the exception.
PermissionType	Gets or sets the type of the permission that failed.
PermitOnlySetInstance	Gets or sets the permission, permission set, or permission set collection that is part of the permit-only stack frame that caused a security check to fail.
RefusedSet	Gets or sets the refused permission set of the assembly that caused the SecurityException .
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception)
StackTrace	Gets a string representation of the immediate frames on the call stack. (Inherited from Exception)

TargetSite	Gets the method that throws the current exception. (Inherited from Exception)
Url	Gets or sets the URL of the assembly that caused the exception.

Methods

[] [Expand table](#)

Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object)
GetBaseException()	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception)
GetHashCode()	Serves as the default hash function. (Inherited from Object)
GetObjectData(SerializationInfo, StreamingContext)	Obsolete. Sets the SerializationInfo with information about the SecurityException .
GetType()	Gets the runtime type of the current instance. (Inherited from Exception)
MemberwiseClone()	Creates a shallow copy of the current Object . (Inherited from Object)
ToString()	Returns a representation of the current SecurityException .

Events

[] [Expand table](#)

SerializeObject	Obsolete.
State	Occurs when an exception is serialized to create an exception state object that contains serialized data about the exception. (Inherited from Exception)

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0

See also

- [Exception](#)
- [Handling and Throwing Exceptions](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

SecurityException Constructors

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Initializes a new instance of the [SecurityException](#) class.

Overloads

[+] Expand table

SecurityException()	Initializes a new instance of the SecurityException class with default properties.
SecurityException(String)	Initializes a new instance of the SecurityException class with a specified error message.
SecurityException(SerializationInfo, StreamingContext)	Obsolete. Initializes a new instance of the SecurityException class with serialized data.
SecurityException(String, Exception)	Initializes a new instance of the SecurityException class with a specified error message and a reference to the inner exception that is the cause of this exception.
SecurityException(String, Type)	Initializes a new instance of the SecurityException class with a specified error message and the permission type that caused the exception to be thrown.
SecurityException(String, Type, String)	Initializes a new instance of the SecurityException class with a specified error message, the permission type that caused the exception to be thrown, and the permission state.

Examples

For an example of the use of a [SecurityException](#) constructor, see the example provided for the [SecurityException\(String, Object, Object, MethodInfo, Object, IPermission\)](#) constructor.

SecurityException()

Initializes a new instance of the [SecurityException](#) class with default properties.

C#

```
public SecurityException ();
```

Examples

For an example of the use of a [SecurityException](#) constructor, see the example provided for the [SecurityException\(String, Object, Object, MethodInfo, Object, IPermission\)](#) constructor.

Remarks

The following table shows the initial property values for an instance of the [SecurityException](#) class.

[] [Expand table](#)

Property	Value
InnerException	A null reference (<code>Nothing</code> in Visual Basic).
Message	The localized error message string.

Applies to

▼ .NET 9 and other versions

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0

SecurityException(String)

Initializes a new instance of the [SecurityException](#) class with a specified error message.

C#

```
public SecurityException (string? message);
```

Parameters

message [String](#)

The error message that explains the reason for the exception.

Examples

For an example of the use of a [SecurityException](#) constructor, see the example provided for the [SecurityException\(String, Object, Object, MethodInfo, Object, IPermission\)](#) constructor.

Remarks

The following table shows the initial property values for an instance of the [SecurityException](#) class.

[+] [Expand table](#)

Property	Value
InnerException	A null reference (<code>Nothing</code> in Visual Basic).
Message	The localized error message string.

Applies to

▼ .NET 9 and other versions

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9

Product	Versions
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0

SecurityException(SerializationInfo, StreamingContext)

⊗ Caution

This API supports obsolete formatter-based serialization. It should not be called or extended by application code.

Initializes a new instance of the [SecurityException](#) class with serialized data.

C#

```
[System.Obsolete("This API supports obsolete formatter-based
serialization. It should not be called or extended by application code.",
DiagnosticId="SYSLIB0051", UrlFormat="https://aka.ms/dotnet-
warnings/{0}")]
protected SecurityException
(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context);
```

Parameters

info [SerializationInfo](#)

The object that holds the serialized object data.

context [StreamingContext](#)

The contextual information about the source or destination.

Attributes [ObsoleteAttribute](#)

Exceptions

ArgumentNullException

`info` is `null`.

Examples

For an example of the use of a [SecurityException](#) constructor, see the example provided for the [SecurityException\(String, Object, Object, MethodInfo, Object, IPermission\)](#) constructor.

Remarks

This constructor is called during deserialization to reconstitute the exception object transmitted over a stream.

This constructor sets the following property values based on the information in the `info` parameter:

- [Action](#)
- [FirstPermissionThatFailed](#)
- [Demanded](#)
- [RefusedSet](#)
- [DenySetInstance](#)
- [PermitOnlySetInstance](#)
- [FailedAssemblyInfo](#)
- [Method](#)
- [Zone](#)
- [Url](#)

Applies to

▼ .NET 9 and other versions

Product	Versions (<i>Obsolete</i>)
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7 (8, 9)

Product	Versions (<i>Obsolete</i>)
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

SecurityException(String, Exception)

Initializes a new instance of the [SecurityException](#) class with a specified error message and a reference to the inner exception that is the cause of this exception.

C#

```
public SecurityException (string? message, Exception? inner);
```

Parameters

message [String](#)

The error message that explains the reason for the exception.

inner [Exception](#)

The exception that is the cause of the current exception. If the `inner` parameter is not `null`, the current exception is raised in a `catch` block that handles the inner exception.

Examples

For an example of the use of a [SecurityException](#) constructor, see the example provided for the [SecurityException\(String, Object, Object, MethodInfo, Object, IPermission\)](#) constructor.

Remarks

An exception that is thrown as a direct result of a previous exception should include a reference to the previous exception in the [InnerException](#) property. The [InnerException](#) property returns the same value that is passed into the constructor, or `null` if the [InnerException](#) property does not supply the inner exception value to the constructor.

The following table shows the initial property values for an instance of the [SecurityException](#) class.

[+] Expand table

Property	Value
InnerException	The inner exception reference.
Message	The localized error message string.

See also

- [Exception](#)
- [Handling and Throwing Exceptions](#)

Applies to

▼ .NET 9 and other versions

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0

SecurityException(String, Type)

Initializes a new instance of the [SecurityException](#) class with a specified error message and the permission type that caused the exception to be thrown.

C#

```
public SecurityException (string? message, Type? type);
```

Parameters

message [String](#)

The error message that explains the reason for the exception.

type [Type](#)

The type of the permission that caused the exception to be thrown.

Examples

For an example of the use of a [SecurityException](#) constructor, see the example provided for the [SecurityException\(String, Object, Object, MethodInfo, Object, IPermission\)](#) constructor.

Remarks

The following table shows the property values set by this constructor.

[Expand table](#)

Property	Value
Message	The localized error message string specified by <code>message</code> .
PermissionType	The Type of the permission that failed, specified by <code>type</code> .

Applies to

▼ .NET 9 and other versions

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

SecurityException(String, Type, String)

Initializes a new instance of the [SecurityException](#) class with a specified error message, the permission type that caused the exception to be thrown, and the permission state.

C#

```
public SecurityException (string? message, Type? type, string? state);
```

Parameters

message [String](#)

The error message that explains the reason for the exception.

type [Type](#)

The type of the permission that caused the exception to be thrown.

state [String](#)

The state of the permission that caused the exception to be thrown.

Examples

For an example of the use of a [SecurityException](#) constructor, see the example provided for the [SecurityException\(String, Object, Object, MethodInfo, Object, IPermission\)](#) constructor.

Remarks

The following table shows the property values set by this constructor.

[] [Expand table](#)

Property	Value
Message	The localized error message string specified by message .
PermissionType	The Type of the permission that failed, specified by type .
Demanded	The demanded security permission, permission set, or permission set collection that failed.

Applies to

▼ .NET 9 and other versions

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityException.Demanded Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets or sets the demanded security permission, permission set, or permission set collection that failed.

C#

```
public object? Demanded { get; set; }
```

Property Value

[Object](#)

A permission, permission set, or permission set collection object.

Examples

The following code example shows the use of the [Demanded](#) property to display the demanded security permission, permission set, or permission set collection that failed. This code example is part of a larger example provided for the [SecurityException](#) class.

C#

```
Display("The demanded permission is: " +
    sE.Demanded.ToString());
```

Remarks

In the case of a returned permission set or permission set collection, the returned object contains all the demanded permissions, one or more of which caused the failure.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityException.DenySetInstance Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets or sets the denied security permission, permission set, or permission set collection that caused a demand to fail.

C#

```
public object? DenySetInstance { get; set; }
```

Property Value

[Object](#)

A permission, permission set, or permission set collection object.

Remarks

This property contains the denied permission, permission set, or permission set collection that caused the security check to fail. It is `null` for exceptions that are not caused by a Deny. The property is of type [Object](#) because it can contain a permission, a permission set, or a permission set collection. To test the run-time type of this property, you can use the [Object.GetType](#) method or a specific language operator, such as the [is operator](#) in C# or the [TypeOf operator](#) in Visual Basic.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1

Product	Versions
.NET Standard	2.0, 2.1

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 .NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityException.FailedAssemblyInfo Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets or sets information about the failed assembly.

C#

```
public System.Reflection.AssemblyName? FailedAssemblyInfo { get; set; }
```

Property Value

[AssemblyName](#)

An [AssemblyName](#) that identifies the failed assembly.

Examples

The following code example shows the use of the [FailedAssemblyInfo](#) property to display the information about the failed assembly.

C#

```
Display("The failed assembly is: " +
    sE.FailedAssemblyInfo.EscapedCodeBase);
```

Remarks

This property contains an [AssemblyName](#) object that identifies the assembly that caused the security check to fail.

 Note

This property is not populated when a security exception occurs in a [Deny](#) or [PermitOnly](#) stack frame, because the assembly issuing the [Deny](#) or [PermitOnly](#) security action is not the assembly that failed the stack walk. In these cases, the security exception is created with a constructor that does not require an assembly name, granted set information, or refused set information.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityException.GrantedSet Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets or sets the granted permission set of the assembly that caused the [SecurityException](#).

C#

```
public string? GrantedSet { get; set; }
```

Property Value

[String](#)

The XML representation of the granted set of the assembly.

Remarks

This property might not contain relevant data in security exceptions other than exceptions involving checks for the [AllowPartiallyTrustedCallersAttribute](#).

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on
GitHub

.NET

.NET feedback

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET is an open source project.
Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

SecurityException.Method Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets or sets the information about the method associated with the exception.

C#

```
public System.Reflection.MethodInfo? Method { get; set; }
```

Property Value

[MethodInfo](#)

A [MethodInfo](#) object describing the method.

Remarks

Important

Using an instance of this object with untrusted data is a security risk. Use this object only with trusted data. For more information, see [Validate All Inputs ↗](#).

This property contains one of the following:

- Information about the method that the failed assembly was executing when the security check that triggered the exception occurred.
- Information about the method that placed either a [PermitOnly](#) or [Deny](#) security action on the call stack, in the case of a failure due to a [PermitOnly](#) or [Deny](#).
- `null`, in the case of a failure that is impossible to attribute to a specific method.

The [MethodInfo](#) object provides the method name, class name, and assembly name information that uniquely identifies the method.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project.
Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

SecurityException.PermissionState Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets or sets the state of the permission that threw the exception.

C#

```
public string? PermissionState { get; set; }
```

Property Value

[String](#)

The state of the permission at the time the exception was thrown.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

more information, see [our contributor guide](#).

SecurityException.PermissionType Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets or sets the type of the permission that failed.

C#

```
public Type? PermissionType { get; set; }
```

Property Value

Type

The type of the permission that failed.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on
GitHub

The source for this content can
be found on GitHub, where you
can also create and review
issues and pull requests. For

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

more information, see [our contributor guide](#).

SecurityException.PermitOnlySet Instance Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets or sets the permission, permission set, or permission set collection that is part of the permit-only stack frame that caused a security check to fail.

C#

```
public object? PermitOnlySetInstance { get; set; }
```

Property Value

[Object](#)

A permission, permission set, or permission set collection object.

Remarks

The [PermitOnlySetInstance](#) property represents the permitted permission, permission set, or permission set collection contained in the stack frame that caused the security exception. For instance, when a security exception occurs because of a [PermissionSet.Demand](#) failure, the permitted permission appears in this property and the demanded [PermissionSet](#) is contained in the [Demanded](#) property.

This property is of type [Object](#) because permissions, permission sets, or permission set collections can all be demanded and [Object](#) is their common base class. To test the run-time type of this property, you can use the [GetType](#) method or a specific language operator, such as the [is operator](#) in C# or the [TypeOf operator](#) in Visual Basic.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityException.RefusedSet Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets or sets the refused permission set of the assembly that caused the [SecurityException](#).

C#

```
public string? RefusedSet { get; set; }
```

Property Value

[String](#)

The XML representation of the refused permission set of the assembly.

Remarks

This property might not contain relevant data in security exceptions other than exceptions involving checks for the [AllowPartiallyTrustedCallersAttribute](#).

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on
GitHub

.NET

.NET feedback

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET is an open source project.
Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

SecurityException.Url Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets or sets the URL of the assembly that caused the exception.

C#

```
public string? Url { get; set; }
```

Property Value

[String](#)

A URL that identifies the location of the assembly.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityException.GetObjectData(SerializationInfo, StreamingContext) Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

⊗ Caution

This API supports obsolete formatter-based serialization. It should not be called or extended by application code.

Sets the [SerializationInfo](#) with information about the [SecurityException](#).

C#

```
[System.Obsolete("This API supports obsolete formatter-based serialization.  
It should not be called or extended by application code.",  
DiagnosticId="SYSLIB0051", UrlFormat="https://aka.ms/dotnet-warnings/{0}")]  
public override void GetObjectData  
(System.Runtime.Serialization.SerializationInfo info,  
System.Runtime.Serialization.StreamingContext context);
```

Parameters

info [SerializationInfo](#)

The [SerializationInfo](#) that holds the serialized object data about the exception being thrown.

context [StreamingContext](#)

The [StreamingContext](#) that contains contextual information about the source or destination.

Attributes [ObsoleteAttribute](#)

Exceptions

ArgumentNullException

The `info` parameter is `null`.

Examples

The following code shows the use of the `GetObjectData` method to display the permission state contained in the `SerializationInfo` object.

C#

```
Display("Demonstrating the use of the GetObjectData method.");
SerializationInfo si = new SerializationInfo(
    typeof(EntryPoint), new FormatterConverter());
sE.GetObjectData(si,
    new StreamingContext(StreamingContextStates.All));
Display("The FirstPermissionThatFailed from the " +
    "call to GetObjectData is: ");
Display(si.GetString("FirstPermissionThatFailed"));
```

Remarks

`GetObjectData` sets a `SerializationInfo` with all the exception object data targeted for serialization. During deserialization, the exception is reconstituted from the `SerializationInfo` transmitted over the stream.

Applies to

Product	Versions (<i>Obsolete</i>)
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7 (8, 9)
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

See also

- [SerializationInfo](#)
- [StreamingContext](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityException.ToString Method

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Returns a representation of the current [SecurityException](#).

C#

```
public override string ToString ();
```

Returns

[String](#)

A string representation of the current [SecurityException](#).

Remarks

The [ToString](#) method returns the full description of the [SecurityException](#).

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0

 Collaborate with us on
GitHub



.NET feedback

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET is an open source project.
Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

SecurityRulesAttribute Class

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Indicates the set of security rules the common language runtime should enforce for an assembly.

C#

```
[System.AttributeUsage(System.AttributeTargets.Assembly,  
AllowMultiple=false)]  
public sealed class SecurityRulesAttribute : Attribute
```

Inheritance Object → [Attribute](#) → SecurityRulesAttribute

Attributes [AttributeUsageAttribute](#)

Remarks

ⓘ Important

Partially trusted code is no longer supported. This attribute has no effect in .NET Core.

This class indicates which set of security rules the common language runtime should enforce for an assembly. For example, an assembly that is marked with

`[SecurityRules(SecurityRuleSet.Level1)]` uses the .NET Framework version 2.0 transparency rules, where public security-critical types and members are treated as security-safe-critical outside the assembly. This requires security-critical types and members to perform a link demand for full trust to enforce security-critical behavior when they are accessed by external callers. Typically, level 1 rules should be used only for compatibility, such as for version 2.0 assemblies. For more information about level 1 behavior, see [Security-Transparent Code, Level 1](#). For information about level 2 behavior, see [Security-Transparent Code, Level 2](#).

Constructors

[+] [Expand table](#)

SecurityRulesAttribute(SecurityRuleSet)	Initializes a new instance of the SecurityRulesAttribute class using the specified rule set value.
---	--

Properties

[+] [Expand table](#)

RuleSet	Gets the rule set to be applied.
SkipVerificationInFullTrust	Determines whether fully trusted transparent code should skip Microsoft intermediate language (MSIL) verification.
TypeId	When implemented in a derived class, gets a unique identifier for this Attribute . (Inherited from Attribute)

Methods

[+] [Expand table](#)

Equals(Object)	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute)
GetHashCode()	Returns the hash code for this instance. (Inherited from Attribute)
GetType()	Gets the Type of the current instance. (Inherited from Object)
IsDefaultAttribute()	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute)
Match(Object)	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute)
MemberwiseClone()	Creates a shallow copy of the current Object . (Inherited from Object)

ToString()	Returns a string that represents the current object. (Inherited from Object)
----------------------------	--

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

See also

- [Security Changes in the .NET Framework Version 4.0](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityRulesAttribute(SecurityRuleSet) Constructor

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Initializes a new instance of the [SecurityRulesAttribute](#) class using the specified rule set value.

C#

```
public SecurityRulesAttribute (System.Security.SecurityRuleSet ruleSet);
```

Parameters

ruleSet [SecurityRuleSet](#)

One of the enumeration values that specifies the transparency rules set.

Remarks

When you specify the `ruleSet` parameter, use [Level1](#) for .NET Framework version 2.0 rules or [Level2](#) for .NET Framework 4 rules. For more information about [Level1](#) behavior, see [Security-Transparent Code, Level 1](#). For information about [Level2](#) behavior, see [Security-Transparent Code, Level 2](#).

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

See also

- [Security Changes in the .NET Framework Version 4.0](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

SecurityRulesAttribute.RuleSet Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Gets the rule set to be applied.

C#

```
public System.Security.SecurityRuleSet RuleSet { get; }
```

Property Value

[SecurityRuleSet](#)

One of the enumeration values that specifies the transparency rules to be applied.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityRulesAttribute.SkipVerificationInFullTrust Property

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Determines whether fully trusted transparent code should skip Microsoft intermediate language (MSIL) verification.

C#

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

Property Value

[Boolean](#)

`true` if MSIL verification should be skipped; otherwise, `false`. The default is `false`.

Remarks

This property should be used only for optimization, because security guarantees made for transparent code cannot be enforced if the code is unverifiable.

If you use this property to skip MSIL verification for an assembly, use the `/transparent` option in the [Perverify tool](#) to statically verify that the assembly's transparent code meets type safety requirements.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityRuleSet Enum

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Identifies the set of security rules the common language runtime should enforce for an assembly.

C#

```
public enum SecurityRuleSet
```

Inheritance Object → [ValueType](#) → [Enum](#) → SecurityRuleSet

Fields

[] [Expand table](#)

Level1	1	Indicates that the runtime will enforce level 1 (.NET Framework version 2.0) transparency rules.
Level2	2	Indicates that the runtime will enforce level 2 transparency rules.
None	0	Unsupported. Using this value results in a FileLoadException being thrown.

Remarks

This enumeration indicates which set of security rules the common language runtime should enforce for an assembly. For example, an assembly that is marked with `[SecurityRules(SecurityRuleSet.Level1)]` uses the .NET Framework version 2.0 transparency rules, where public security-critical types and members are treated as security-safe-critical outside the assembly. This requires security-critical types and members to perform a link demand for full trust to enforce security-critical behavior when they are accessed by external callers. Typically, level 1 rules should be used only for compatibility, such as for .NET Framework 2.0 assemblies. By default, .NET Framework 2.0 assemblies become level 2 assemblies when they are recompiled for the

.NET Framework 4. To compile these assemblies as level 1, you must mark them explicitly as level 1. For more information about level 1 behavior, see [Security-Transparent Code, Level 1](#). For information about level 2 behavior, see [Security-Transparent Code, Level 2](#).

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

See also

- [Security Changes in the .NET Framework Version 4.0](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

SecuritySafeCriticalAttribute Class

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Identifies types or members as security-critical and safely accessible by transparent code.

C#

```
[System.AttributeUsage(System.AttributeTargets.Class |  
    System.AttributeTargets.Constructor | System.AttributeTargets.Delegate |  
    System.AttributeTargets.Enum | System.AttributeTargets.Field |  
    System.AttributeTargets.Interface | System.AttributeTargets.Method |  
    System.AttributeTargets.Struct, AllowMultiple=false, Inherited=false)]  
public sealed class SecuritySafeCriticalAttribute : Attribute
```

Inheritance Object → [Attribute](#) → SecuritySafeCriticalAttribute

Attributes [AttributeUsageAttribute](#)

Remarks

Important

Partially trusted code is no longer supported. This attribute has no effect in .NET Core.

Types or members that are marked with the [SecuritySafeCriticalAttribute](#) attribute can be accessed by partially trusted types and members. These partially trusted types and members can be within any assembly that is marked with the [SecurityTransparentAttribute](#) or [AllowPartiallyTrustedCallersAttribute](#) (APCA) attribute, or they are partially trusted for other reasons, such as being loaded into partial trust. Code that is marked with the [SecuritySafeCriticalAttribute](#) must be subject to a rigorous security audit to ensure that it can be used safely in a secure execution environment.

Security-safe-critical code must validate the permissions of callers to determine whether they have authority to access protected resources used by the code.

Constructors

[+] Expand table

SecuritySafeCriticalAttribute()	Initializes a new instance of the SecuritySafeCriticalAttribute class.
---	--

Properties

[+] Expand table

TypeId	When implemented in a derived class, gets a unique identifier for this Attribute . (Inherited from Attribute)
------------------------	---

Methods

[+] Expand table

Equals(Object)	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute)
GetHashCode()	Returns the hash code for this instance. (Inherited from Attribute)
GetType()	Gets the Type of the current instance. (Inherited from Object)
IsDefaultAttribute()	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute)
Match(Object)	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute)
MemberwiseClone()	Creates a shallow copy of the current Object . (Inherited from Object)
ToString()	Returns a string that represents the current object. (Inherited from Object)

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0

See also

- [SecurityCriticalAttribute](#)
- [SecurityTransparentAttribute](#)
- [Security-Transparent Code, Level 1](#)
- [Security-Transparent Code, Level 2](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

SecuritySafeCriticalAttribute Constructor

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Initializes a new instance of the [SecuritySafeCriticalAttribute](#) class.

C#

```
public SecuritySafeCriticalAttribute();
```

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityTransparentAttribute Class

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Specifies that an assembly cannot cause an elevation of privilege.

C#

```
[System.AttributeUsage(System.AttributeTargets.Assembly,  
AllowMultiple=false, Inherited=false)]  
public sealed class SecurityTransparentAttribute : Attribute
```

Inheritance Object → [Attribute](#) → SecurityTransparentAttribute

Attributes [AttributeUsageAttribute](#)

Remarks

Important

Partially trusted code is no longer supported. This attribute has no effect in .NET Core.

Transparent assemblies can be accessed from partially trusted code and cannot expose access to any protected resources or functionality. Code in the assembly is not allowed to suppress code access security checks and cannot cause an elevation of privilege.

Note

Transparency is enforced by the just-in-time compiler, not by the security infrastructure code. Applying this attribute to an assembly allows the assembly to access only transparent and security-safe-critical types and members regardless of its permission set, including full trust. Transparent code that accesses a security-critical type or member results in a [MethodAccessException](#) being thrown.

Constructors

[+] Expand table

SecurityTransparentAttribute()	Initializes a new instance of the SecurityTransparentAttribute class.
--	---

Properties

[+] Expand table

TypeId	When implemented in a derived class, gets a unique identifier for this Attribute . (Inherited from Attribute)
------------------------	---

Methods

[+] Expand table

Equals(Object)	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute)
GetHashCode()	Returns the hash code for this instance. (Inherited from Attribute)
GetType()	Gets the Type of the current instance. (Inherited from Object)
IsDefaultAttribute()	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute)
Match(Object)	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute)
MemberwiseClone()	Creates a shallow copy of the current Object . (Inherited from Object)
ToString()	Returns a string that represents the current object. (Inherited from Object)

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0

See also

- [SecurityCriticalAttribute](#)
- [SecurityTransparentAttribute](#)
- [Security-Transparent Code, Level 1](#)
- [Security-Transparent Code, Level 2](#)

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityTransparentAttribute Constructor

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Initializes a new instance of the [SecurityTransparentAttribute](#) class.

C#

```
public SecurityTransparentAttribute ();
```

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityTreatAsSafeAttribute Class

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

⊗ Caution

SecurityTreatAsSafe is only used for .NET 2.0 transparency compatibility. Use the SecuritySafeCriticalAttribute instead.

Identifies which of the nonpublic [SecurityCriticalAttribute](#) members are accessible by transparent code within the assembly.

C#

```
[System.AttributeUsage(System.AttributeTargets.Assembly |  
    System.AttributeTargets.Class | System.AttributeTargets.Constructor |  
    System.AttributeTargets.Delegate | System.AttributeTargets.Enum |  
    System.AttributeTargets.Field | System.AttributeTargets.Interface |  
    System.AttributeTargets.Method | System.AttributeTargets.Struct,  
    AllowMultiple=false, Inherited=false)]  
[System.Obsolete("SecurityTreatAsSafe is only used for .NET 2.0 transparency  
compatibility. Use the SecuritySafeCriticalAttribute instead.")]  
public sealed class SecurityTreatAsSafeAttribute : Attribute
```

Inheritance Object → [Attribute](#) → SecurityTreatAsSafeAttribute

Attributes [AttributeUsageAttribute](#), [ObsoleteAttribute](#)

Remarks

ⓘ Important

Partially trusted code is no longer supported. This attribute has no effect in .NET Core.

Types or members marked with the [SecurityCriticalAttribute](#) attribute and the [SecurityTreatAsSafeAttribute](#) attribute can be accessed by types and members within the assembly that are marked with the [SecurityTransparentAttribute](#) attribute.

Constructors

[] [Expand table](#)

SecurityTreatAsSafeAttribute()	Initializes a new instance of the SecurityTreatAsSafeAttribute class.
--	---

Properties

[] [Expand table](#)

TypeId	When implemented in a derived class, gets a unique identifier for this Attribute . (Inherited from Attribute)
------------------------	---

Methods

[] [Expand table](#)

Equals(Object)	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute)
GetHashCode()	Returns the hash code for this instance. (Inherited from Attribute)
GetType()	Gets the Type of the current instance. (Inherited from Object)
IsDefaultAttribute()	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute)
Match(Object)	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute)
MemberwiseClone()	Creates a shallow copy of the current Object . (Inherited from Object)
ToString()	Returns a string that represents the current object.

Applies to

Product	Versions (<i>Obsolete</i>)
.NET	(Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9)
.NET Framework	2.0, 3.0, 3.5 (4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1)
.NET Standard	(2.0, 2.1)

See also

- [SecurityTransparentAttribute](#)
- [SecurityCriticalAttribute](#)

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 .NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecurityTreatAsSafeAttribute Constructor

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Initializes a new instance of the [SecurityTreatAsSafeAttribute](#) class.

C#

```
public SecurityTreatAsSafeAttribute ();
```

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SuppressUnmanagedCodeSecurityAttribute Class

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Allows managed code to call into unmanaged code without a stack walk. This class cannot be inherited.

C#

```
[System.AttributeUsage(System.AttributeTargets.Class |  
    System.AttributeTargets.Delegate | System.AttributeTargets.Interface |  
    System.AttributeTargets.Method, AllowMultiple=true, Inherited=false)]  
public sealed class SuppressUnmanagedCodeSecurityAttribute : Attribute
```

Inheritance Object → [Attribute](#) → SuppressUnmanagedCodeSecurityAttribute

Attributes [AttributeUsageAttribute](#)

Remarks

ⓘ Important

Partially trusted code is no longer supported. This attribute has no effect in .NET Core.

✖ Caution

Use this attribute with extreme care. Incorrect use can create security weaknesses.

This attribute can be applied to methods that want to call into native code without incurring the performance loss of a run-time security check when doing so. The stack walk performed when calling unmanaged code is omitted at run time, resulting in

substantial performance savings. Using this attribute in a class applies it to all contained methods.

Generally, whenever managed code calls into unmanaged code (by `PInvoke` or COM interop into native code), there is a demand for the `UnmanagedType` permission to ensure all callers have the necessary permission to allow this. By applying this explicit attribute, developers can suppress the demand at run time. The developer must take responsibility for assuring that the transition into unmanaged code is sufficiently protected by other means. The demand for the `UnmanagedType` permission will still occur at link time. For example, if function A calls function B and function B is marked with `SuppressUnmanagedCodeSecurityAttribute`, function A will be checked for unmanaged code permission during just-in-time compilation, but not subsequently during run time.

This attribute is only effective when applied to `PInvoke` methods (or classes that contain `PInvoke` methods) or the definition of an interface through which interop calls will be made. It will be ignored in all other contexts.

This attribute is useful for implementing a class that provides access to system resources through unmanaged code. Code that does not have permission to access unmanaged code can call a class with this attribute to access unmanaged code. This is only safe if the writer of the class with this attribute has programmed the class to be secure. If not, this attribute is dangerous and can allow the code that uses it to be misused.

This is not a declarative security attribute, but a regular attribute (it derives from [Attribute](#), not [SecurityAttribute](#)).

Constructors

[+] Expand table

<code>SuppressUnmanagedCodeSecurityAttribute()</code>	Initializes a new instance of the SuppressUnmanagedCodeSecurityAttribute class.
---	---

Properties

[+] Expand table

<code>TypeId</code>	When implemented in a derived class, gets a unique identifier for this Attribute . (Inherited from Attribute)
---------------------	---

Methods

 Expand table

Equals(Object)	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute)
GetHashCode()	Returns the hash code for this instance. (Inherited from Attribute)
GetType()	Gets the Type of the current instance. (Inherited from Object)
IsDefaultAttribute()	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute)
Match(Object)	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute)
MemberwiseClone()	Creates a shallow copy of the current Object . (Inherited from Object)
ToString()	Returns a string that represents the current object. (Inherited from Object)

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

See also

- [Extending Metadata Using Attributes](#)



Collaborate with us on

GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SuppressUnmanagedCodeSecurityAttribute Constructor

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Initializes a new instance of the [SuppressUnmanagedCodeSecurityAttribute](#) class.

C#

```
public SuppressUnmanagedCodeSecurityAttribute();
```

Remarks

The parameterless constructor initializes any fields to their default values.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

UnverifiableCodeAttribute Class

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Marks modules containing unverifiable code. This class cannot be inherited.

C#

```
[System.AttributeUsage(System.AttributeTargets.Module, AllowMultiple=true,
Inherited=false)]
public sealed class UnverifiableCodeAttribute : Attribute
```

Inheritance Object → [Attribute](#) → UnverifiableCodeAttribute

Attributes [AttributeUsageAttribute](#)

Remarks

This custom attribute is only for use by compilers and not intended to be used directly by application developers.

Modules containing unverifiable code should be marked with this attribute. This attribute carries no internal state. Its presence in the module metadata indicates that the module contains unverifiable code.

Constructors

[] [Expand table](#)

UnverifiableCodeAttribute()	Initializes a new instance of the UnverifiableCodeAttribute class.
---	--

Properties

[] [Expand table](#)

TypeId When implemented in a derived class, gets a unique identifier for this [Attribute](#).
(Inherited from [Attribute](#))

Methods

[+] [Expand table](#)

Equals(Object)	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute)
GetHashCode()	Returns the hash code for this instance. (Inherited from Attribute)
GetType()	Gets the Type of the current instance. (Inherited from Object)
IsDefaultAttribute()	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute)
Match(Object)	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute)
MemberwiseClone()	Creates a shallow copy of the current Object . (Inherited from Object)
ToString()	Returns a string that represents the current object. (Inherited from Object)

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

See also

- [Extending Metadata Using Attributes](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

UnverifiableCodeAttribute Constructor

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Initializes a new instance of the [UnverifiableCodeAttribute](#) class.

C#

```
public UnverifiableCodeAttribute ();
```

Remarks

The parameterless constructor initializes any fields to their default values.

Applies to

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

VerificationException Class

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

The exception that is thrown when the security policy requires code to be type safe and the verification process is unable to verify that the code is type safe.

C#

```
public class VerificationException : SystemException
```

Inheritance Object → [Exception](#) → [SystemException](#) → VerificationException

Constructors

[+] Expand table

VerificationException()	Initializes a new instance of the VerificationException class with default properties.
VerificationException(SerializationInfo, StreamingContext)	Obsolete. Initializes a new instance of the VerificationException class with serialized data.
VerificationException(String)	Initializes a new instance of the VerificationException class with an explanatory message.
VerificationException(String, Exception)	Initializes a new instance of the VerificationException class with a specified error message and a reference to the inner exception that is the cause of this exception.

Properties

[+] Expand table

Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception)
Message	Gets a message that describes the current exception. (Inherited from Exception)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception)
StackTrace	Gets a string representation of the immediate frames on the call stack. (Inherited from Exception)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception)

Methods

[] [Expand table](#)

Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object)
GetBaseException()	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception)
GetHashCode()	Serves as the default hash function. (Inherited from Object)
GetObjectData(SerializationInfo, StreamingContext)	Obsolete. When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception)
GetType()	Gets the runtime type of the current instance. (Inherited from Exception)

MemberwiseClone()	Creates a shallow copy of the current Object . (Inherited from Object)
ToString()	Creates and returns a string representation of the current exception. (Inherited from Exception)

Events

[\[+\] Expand table](#)

SerializeObject	Obsolete.
State	Occurs when an exception is serialized to create an exception state object that contains serialized data about the exception. (Inherited from Exception)

Applies to

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0

See also

- [Exception](#)
- [Handling and Throwing Exceptions](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For



.NET feedback

.NET is an open source project. Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

more information, see [our contributor guide](#).

VerificationException Constructors

Reference

Definition

Namespace: [System.Security](#)

Assembly: System.Runtime.dll

Initializes a new instance of the [VerificationException](#) class.

Overloads

[+] Expand table

VerificationException()	Initializes a new instance of the VerificationException class with default properties.
VerificationException(String)	Initializes a new instance of the VerificationException class with an explanatory message.
VerificationException(SerializationInfo, StreamingContext)	Obsolete. Initializes a new instance of the VerificationException class with serialized data.
VerificationException(String, Exception)	Initializes a new instance of the VerificationException class with a specified error message and a reference to the inner exception that is the cause of this exception.

VerificationException()

Initializes a new instance of the [VerificationException](#) class with default properties.

C#

```
public VerificationException ();
```

Remarks

[VerificationException](#) uses the HRESULT COR_E_VERIFICATION.

Applies to

- ▼ .NET 9 and other versions

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0

VerificationException(String)

Initializes a new instance of the [VerificationException](#) class with an explanatory message.

C#

```
public VerificationException (string? message);
```

Parameters

message [String](#)

A message indicating the reason the exception occurred.

Applies to

- ▼ .NET 9 and other versions

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0

VerificationException(SerializationInfo, StreamingContext)

⊗ Caution

This API supports obsolete formatter-based serialization. It should not be called or extended by application code.

Initializes a new instance of the [VerificationException](#) class with serialized data.

C#

```
[System.Obsolete("This API supports obsolete formatter-based  
serialization. It should not be called or extended by application code.",  
DiagnosticId="SYSLIB0051", UrlFormat="https://aka.ms/dotnet-  
warnings/{0}")]  
protected VerificationException  
(System.Runtime.Serialization.SerializationInfo info,  
System.Runtime.Serialization.StreamingContext context);
```

Parameters

info [SerializationInfo](#)

The object that holds the serialized object data.

context [StreamingContext](#)

The contextual information about the source or destination.

Attributes [ObsoleteAttribute](#)

Remarks

This constructor is called during deserialization to reconstitute the exception object transmitted over a stream.

Applies to

- ▼ .NET 9 and other versions

Product	Versions (<i>Obsolete</i>)
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7 (8, 9)
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

VerificationException(String, Exception)

Initializes a new instance of the [VerificationException](#) class with a specified error message and a reference to the inner exception that is the cause of this exception.

C#

```
public VerificationException (string? message, Exception?
innerException);
```

Parameters

message [String](#)

The error message that explains the reason for the exception.

innerException [Exception](#)

The exception that is the cause of the current exception. If the `innerException` parameter is not `null`, the current exception is raised in a `catch` block that handles the inner exception.

Remarks

An exception that is thrown as a direct result of a previous exception should include a reference to the previous exception in the [InnerException](#) property. The [InnerException](#) property returns the same value that is passed into the constructor, or `null` if the [InnerException](#) property does not supply the inner exception value to the constructor.

The following table shows the initial property values for an instance of [VerificationException](#).

[] Expand table

Property	Value
InnerException	The inner exception reference.
Message	The error message string.

See also

- [Exception](#)
- [Serialization](#)

Applies to

▼ .NET 9 and other versions

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2.0, 2.1
UWP	10.0

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)