# System.Security.PermissionSet Class

```
[ILAsm]
.class public serializable PermissionSet extends System.Object implements
System.Collections.ICollection, System.Collections.IEnumerable

[C#]
public class PermissionSet: ICollection, IEnumerable
```

## Assembly Info:

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

## Implements:

- **System.Collections.ICollection**
- **System.Collections.IEnumerable**

## Summary

Represents a collection that can contain different kinds of permissions and perform security operations.

## Inherits From: System.Object

**Library:** BCL

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

## Description

[*Note:* Use `System.Security.PermissionSet` to perform operations on different permission types as a group.]

The XML encoding of a `System.Security.PermissionSet` instance is defined below in EBNF format. The following conventions are used:

- All non-literals in the grammar below are shown in normal type.
- All literals are in bold font.

The following meta-language symbols are used:

- '*' represents a meta-language symbol suffixing an expression that can appear zero or more times.

- '?' represents a meta-language symbol suffixing an expression that can appear zero or one time.

- '+' represents a meta-language symbol suffixing an expression that can appear one or more times.

- '(',')' is used to group literals, non-literals or a mixture of literals and non-literals.

- '|' denotes an exclusive disjunction between two expressions.

- '::= ' denotes a production rule where a left hand non-literal is replaced by a right hand expression containing literals, non-literals or both.

The XML encoding of a `System.Security.PermissionSet` instance is as follows:

```
PermissionSet::=


(


<PermissionSet


class="System.Security.PermissionSet"


version="1" Unrestricted="true"/>


)


|


(


<PermissionSet


class="System.Security.PermissionSet"


version="1">

DnsPermissionXML ?

SocketPermissionXML ?

WebPermissionXML ?

EnvironmentPermissionXML ?

FileIOPermissionXML ?

ReflectionPermissionXML ?

SecurityPermissionXML ?

CustomPermissionXML *

</PermissionSet>

)
```

CustomPermissionXML represents any custom permission. The XML encoding for custom permissions makes use of the following symbols:

ClassName is the name of the class implementing the permission.

AssemblyName is the name of the assembly that contains the class implementing the permission.

Version is the version number indicating the version of the assembly implementing the permission.

StrongNamePublicKeyToken is the strong name public key token constituting the strong name of the assembly that implements the permission.

version is version information for the custom permission. Format and content are defined by the author of the custom permission.

PermissionAttributes is any attribute and attribute value on the `System.Security.IPermission` element used by the permission to represent a particular permission state, for example, unrestricted= "true". Format and content are defined by the author of the custom permission.

PermissionXML is any valid XML used by the permission to represent permission state. Format and content are defined by the author of the custom permission.

The XML encoding of a custom permission instance is as follows:

```
CustomPermissionXML::=

<IPermission class="

ClassName,

AssemblyName,

Version=Version,

Culture=neutral,

PublicKeyToken=StrongNamePublicKeyToken"

version="version"

(PermissionAttributes)*

>

(PermissionXML)?

</IPermission>
```

## PermissionSet(System.Security.PermissionSet) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(class
System.Security.PermissionSet permSet)

[C#]
public PermissionSet(PermissionSet permSet)
```

### Summary

Constructs a new instance of the `System.Security.PermissionSet` class with the values of the specified `System.Security.PermissionSet` instance.

### Parameters

| Parameter | Description |
|---|---|
| *permSet* | The `System.Security.PermissionSet` instance with which to initialize the values of the new instance, or `null` to initialize an empty permission set. |

### Description

If *permSet* is not `null`, the new instance is initialized with copies of the objects in *permSet*, not references to those objects. If *permSet* is `null`, the new instance contains no permissions.

[*Note:* To add a permission to an empty `System.Security.PermissionSet`, use `System.Security.PermissionSet.AddPermission`.]

### Exceptions

| Exception | Condition |
|---|---|
| **System.ArgumentException** | *permSet* is not `null` and is not an instance of `System.Security.PermissionSet`. |

## PermissionSet(System.Security.Permissions.PermissionState) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(valuetype
System.Security.Permissions.PermissionState state)

[C#]
public PermissionSet(PermissionState state)
```

### Summary

Constructs a new instance of the `System.Security.PermissionSet` class with the specified value.

### Parameters

| Parameter | Description |
|---|---|
| *state* | A `System.Security.Permissions.PermissionState` value. This value is either `System.Security.Permissions.PermissionState.None` or `System.Security.Permissions.PermissionState. Unrestricted`, to specify fully restricted or fully unrestricted access. |

### Description

[*Note:* The new instance contains no permissions. To add a permission to the new instance, use `System.Security.PermissionSet.AddPermission`.]

### Exceptions

| Exception | Condition |
|---|---|
| **System.ArgumentException** | *state* is not a valid `System.Security.Permissions.PermissionState` value. |

## PermissionSet.AddPermission(System.Security.IPermission) Method

```
[ILAsm]
.method public hidebysig virtual class System.Security.IPermission
AddPermission(class System.Security.IPermission perm)

[C#]
public virtual IPermission AddPermission(IPermission perm)
```

### Summary

Adds the specified `System.Security.IPermission` object to the current instance if that permission does not already exist in the current instance.

### Parameters

| Parameter | Description |
|---|---|
| *perm* | The `System.Security.IPermission` object to add. |

### Return Value

If *perm* is `null`, returns `null`. If a permission of the same type as *perm* already exists in the current instance, the union of the existing permission and *perm* is added to the current instance and is returned.

### Behaviors

The `System.Security.IPermission` is added if *perm* is not `null` and a permission of the same type as *perm* does not already exist in the current instance.

### Usage

Use this method to add permission objects to the current instance.

### Exceptions

| Exception | Condition |
|---|---|

| | |
|---|---|
| **System.ArgumentException** | *perm* is not a `System.Security.IPermission` object. |

## PermissionSet.Assert() Method

```
[ILAsm]
.method public hidebysig virtual void Assert()

[C#]
public virtual void Assert()
```

### Summary

Asserts that calling code can access the resources identified by the permissions contained in the current instance through the code that calls this method, even if callers have not been granted permission to access the resource.

### Description

[*Note:* This method is the only way to assert multiple permissions at the same time within a frame because only a single assert can be active on a frame at one time; subsequent asserts will result in an exception.]

### Behaviors

As described above.

### Usage

Use this method to insure that all callers can access a set of secured resources.

### Exceptions

| Exception | Condition |
|---|---|
| **System.Security.SecurityException** | The asserting code does not have sufficient permission to call this method.<br><br>-or-<br><br>This method was called with permissions already asserted for the current stack frame. |

### Permissions

| Permission | Description |
|---|---|
| **System.Security.Permissions.SecurityPermission** | Requires permission to perform the assertion security operation. See `System.Security.Permissions.SecurityPermissionFlag.Assertion`. |

## PermissionSet.Copy() Method

```
[ILAsm]
.method public hidebysig virtual class System.Security.PermissionSet
Copy()

[C#]
public virtual PermissionSet Copy()
```

### Summary

Returns a new `System.Security.PermissionSet` containing copies of the objects in the current instance.

### Return Value

A new `System.Security.PermissionSet` that is value equal to the current instance.

### Behaviors

This method creates copies of the permission objects in the current instance, and adds them to the new instance.

### Default

This method calls the `System.Security.PermissionSet` constructor that takes a `System.Security.PermissionSet` argument, and passes the current instance as that parameter.

### Usage

Use this method to create a new `System.Security.PermissionSet` instance containing permissions that are identical to the permissions contained in the current instance.

# PermissionSet.CopyTo(System.Array, System.Int32) Method

```
[ILAsm]
.method public hidebysig virtual void CopyTo(class System.Array array,
int32 index)

[C#]
public virtual void CopyTo(Array array, int index)
```

## Summary

Copies the permission objects in the current instance to the specified location in the specified `System.Array`.

## Parameters

| Parameter | Description |
|---|---|
| *array* | The destination `System.Array`. |
| *index* | A `System.Int32` that specifies the zero-based starting position in the array at which to begin copying. |

## Description

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

## Behaviors

As described above.

## Default

The default implementation uses the `System.Array.SetValue(System.Object, System.Int32)` method to add the value to the array.

## How and When to Override

Override this method to customize the manner in which elements are added to *array*.

**Exceptions**

| **Exception** | **Condition** |
|---|---|
| **System.ArgumentException** | *array* has more than one dimension. |
| **System.IndexOutOfRangeException** | *index* is outside the range of allowable values for *array*. |
| **System.ArgumentNullException** | *array* is `null`. |

## PermissionSet.Demand() Method

```
[ILAsm]
.method public hidebysig virtual void Demand()

[C#]
public virtual void Demand()
```

### Summary

Forces a `System.Security.SecurityException` if all callers do not have the permissions specified by the objects contained in the current instance.

### Behaviors

The permission check for `System.Security.PermissionSet.Demand` begins with the immediate caller of the code that calls this method and continues until all callers have been checked or a caller has been found that is not granted the demanded permission, in which case a `System.Security.SecurityException` exception is thrown.

If the current instance is empty, a call to `System.Security.PermissionSet.Demand` succeeds.

### Usage

Use this method to ensure in a single operation that all callers have all permissions contained in a permission set.

### Exceptions

| Exception | Condition |
|---|---|
| **System.Security.SecurityException** | A caller does not have the permission specified by the current instance. |

# PermissionSet.Deny() Method

```
[ILAsm]
.method public hidebysig virtual void Deny()

[C#]
public virtual void Deny()
```

## Summary

Denies access to the resources secured by the objects contained in the current instance through the code that calls this method.

## Description

This is the only way to deny multiple permissions at the same time within a frame because only a single deny can be active on a frame at one time; subsequent denies will result in an exception.

## Behaviors

This method is required to prevent callers from accessing all resources protected by the objects in the current instance even if the callers had been granted permission to access them.

A call to `System.Security.PermissionSet.Deny` is effective until the calling code returns.

## Usage

Use this method to force all security checks for the objects contained in the current instance to fail.

## Exceptions

| Exception | Condition |
|---|---|
| **System.Security.SecurityException** | A previous call to `Deny` has already restricted the permissions for the current stack frame. |

## PermissionSet.FromXml(System.Security.SecurityElement) Method

```
[ILAsm]
.method public hidebysig virtual void FromXml(class
System.Security.SecurityElement et)

[C#]
public virtual void FromXml(SecurityElement et)
```

### Summary

Reconstructs the state of a `System.Security.PermissionSet` object using the specified XML encoding.

### Parameters

| Parameter | Description |
|---|---|
| *et* | A `System.Security.SecurityElement` instance containing the XML encoding to use to reconstruct the state of a `System.Security.PermissionSet` object. |

### Description

[*Note:* For the XML encoding for this class, see the `System.Security.PermissionSet` class page.]

### Behaviors

When this call completes, the objects in the current instance are required to be identical to the objects in the `System.Security.PermissionSet` encoded in *et*.

### How and When to Override

Override this method to reconstruct subclasses of `System.Security.PermissionSet`.

### Usage

Applications do not typically call this method; it is called by the system.

**Exceptions**

| Exception | Condition |
|---|---|
| **System.ArgumentNullException** | *et* is `null`. |
| **System.ArgumentException** | *et* does not contain an XML encoding for a `System.Security.PermissionSet` instance.<br><br>-or-<br><br>An error occurred while reconstructing *et*. |

# PermissionSet.GetEnumerator() Method

[ILAsm]
```
.method public hidebysig virtual class System.Collections.IEnumerator
GetEnumerator()
```

[C#]
```
public virtual IEnumerator GetEnumerator()
```

## Summary

Returns an enumerator used to iterate over the permissions in the current instance.

## Return Value

A `System.Collections.IEnumerator` object for the permissions of the set.

## Description

[*Note:* This method is implemented to support the `System.Collections.ICollection` interface, which supports the `System.Collections.IEnumerable` interface.]

## Behaviors

As described above.

## How and When to Override

Override this method to customize the enumerator returned by this method.

## PermissionSet.IsSubsetOf(System.Security.PermissionSet) Method

```
[ILAsm]
.method public hidebysig virtual bool IsSubsetOf(class
System.Security.PermissionSet target)

[C#]
public virtual bool IsSubsetOf(PermissionSet target)
```

### Summary

Determines whether the current instance is a subset of the specified object.

### Parameters

| Parameter | Description |
|---|---|
| *target* | A `System.Security.PermissionSet` instance that is to be tested for the subset relationship. |

### Return Value

`true` if the current instance is a subset of *target*; otherwise, `false`.

### Description

[*Note:* The current instance is a subset *target* if all demands that succeed for the current instance also succeed for *target*. That is, the current instance is a subset of *target* if *target* contains at least the permissions contained in the current instance.

If this method returns `true`, the current instance does not describe a level of access to a set of resources that is not already described by *target*.

]

### Behaviors

As described above.

### Usage

Use this method to determine if the all permissions contained in the current instance are also contained in *target*.

## PermissionSet.PermitOnly() Method

```
[ILAsm]
.method public hidebysig virtual void PermitOnly()

[C#]
public virtual void PermitOnly()
```

### Summary

Specifies that only the resources described by the current instance can be accessed by calling code, even if the code has been granted permission to access other resources.

### Description

[*Note:* `System.Security.PermissionSet.PermitOnly` is similar to `System.Security.PermissionSet.Deny` in that both methods cause access to fail where it might otherwise succeed. The difference is that `System.Security.PermissionSet.Deny` specifies permissions for which to refuse access, while `System.Security.PermissionSet.PermitOnly` specifies the only permissions that will succeed.

This is the only way to permit multiple permissions at the same time within a stack frame because only a single permit at a time can be active on a frame; subsequent permits will result in an exception.

]

### Behaviors

Callers are required to be prevented from accessing resources not secured by the contents of the current instance, even if a caller has been granted permission to access such resources.

A `System.Security.PermissionSet.PermitOnly` is in effect until the calling code returns to its caller.

### Usage

Use this method to limit access to a specified set of resources.

### Exceptions

| **Exception** | **Condition** |
|---|---|
| **System.Security.SecurityException** | A previous call to `PermitOnly` has already set the |

| | |
|---|---|
| | permissions for the current stack frame. |

# PermissionSet.ToString() Method

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

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

## Summary

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

## Return Value

A `System.String` containing the XML representation of the state of the current instance.

## Description

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

]

## Example

The following example displays the XML that encodes the state of a `System.Security.PermissionSet`.

[C#]

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

public class PermissionSetToStringExample {
  public static void Main() {

    PermissionSet ps = new PermissionSet(PermissionState.Unrestricted);
    Console.WriteLine(ps.ToString());
  }
}
```

The output is

```
<PermissionSet class="System.Security.PermissionSet" version="1" Unrestricted="true"/>
```

# PermissionSet.ToXml() Method

[ILAsm]
```
.method public hidebysig virtual class System.Security.SecurityElement
ToXml()
```

[C#]
```
public virtual SecurityElement ToXml()
```

## Summary

Returns the XML encoding of the current instance.

## Return Value

A `System.Security.SecurityElement` containing an XML encoding of the state of the current instance.

## Behaviors

As described above.

## How and When to Override

Override this method to return an object containing the XML encoding for types derived from `System.Security.PermissionSet`.

## Usage

This method is called by the system.

# PermissionSet.Union(System.Security.PermissionSet) Method

```
[ILAsm]
.method public hidebysig virtual class System.Security.PermissionSet
Union(class System.Security.PermissionSet other)

[C#]
public virtual PermissionSet Union(PermissionSet other)
```

## Summary

Returns a `System.Security.PermissionSet` object that is the union of the current instance and the specified object.

## Parameters

| Parameter | Description |
|---|---|
| *other* | A `System.Security.PermissionSet` instance to be combined with the current instance. |

## Return Value

A new `System.Security.PermissionSet` instance that represents the union of the current instance and *other*. If the current instance or *other* is unrestricted, returns a `System.Security.PermissionSet` instance that is unrestricted.

## Description

The result of a call to `System.Security.PermissionSet.Union` is a new `System.Security.PermissionSet` instance that represents all the operations represented by the current instance as well as all the operations represented by *other*. If either set is unrestricted, the union is unrestricted, as well.

## Behaviors

As described above.

## Usage

Use this method to create a `System.Security.PermissionSet` instance that contains all of the permissions of the current instance and *other*.

## PermissionSet.Count Property

[ILAsm]
```
.property int32 ICollection.Count { public hidebysig virtual abstract
specialname int32 get_ICollection.Count() }
```

[C#]
```
int ICollection.Count { get; }
```

### Summary

Implemented to support the `System.Collections.ICollection` interface. [Note: For more information, see `System.Collections.ICollection.Count`.]

## PermissionSet.IsSynchronized Property

[ILAsm]
```
.property bool ICollection.IsSynchronized { public hidebysig virtual
abstract specialname bool get_ICollection.IsSynchronized() }
```

[C#]
```
bool ICollection.IsSynchronized { get; }
```

### Summary

Implemented to support the `System.Collections.ICollection` interface. [Note: For more information, see `System.Collections.ICollection.IsSynchronized`.]

## PermissionSet.SyncRoot Property

```
[ILAsm]
.property object ICollection.SyncRoot { public hidebysig virtual abstract
specialname object get_ICollection.SyncRoot() }

[C#]
object ICollection.SyncRoot { get; }
```

### Summary

Implemented to support the `System.Collections.ICollection` interface. [Note: For more information, see `System.Collections.ICollection.SyncRoot`.]