

# Socket 类

.NET Framework 4

实现 Berkeley 套接字接口。

## 继承层次结构

System.Object  
System.Net.Sockets.Socket

命名空间：[System.Net.Sockets](#)  
程序集：System (在 System.dll 中)

## 语法

C#

```
public class Socket : IDisposable
```

Socket 类型公开以下成员。












## 构造函数

显示: ☒ 继承 ☒ 保护

	名称	说明
	<a href="#">Socket(SocketInformation)</a>	使用 <a href="#">DuplicateAndClose</a> 返回的指定的值初始化 Socket 类的新实例。
	<a href="#">Socket(AddressFamily, SocketType, ProtocolType)</a>	使用指定的地址族、套接字类型和协议初始化 Socket 类的新实例。

[页首](#)

## 属性

	名称	说明
	AddressFamily	获取 Socket 的地址族。
	Available	获取已经从网络接收且可供读取的数据量。
	Blocking	获取或设置一个值，该值指示 Socket 是否处于阻止模式。
	Connected	获取一个值，该值指示 Socket 在上次 Send 或者 Receive 操作时是否连接到远程主机。
	DontFragment	获取或设置 Boolean 值，该值指定 Socket 是否允许将 Internet 协议 (IP) 数据报分段。
	EnableBroadcast	获取或设置一个 Boolean 值，该值指定 Socket 是否可以发送或接收广播数据包。
	ExclusiveAddressUse	获取或设置 Boolean 值，该值指定 Socket 是否仅允许一个进程绑定到端口。
	Handle	获取 Socket 的操作系统句柄。
	IsBound	获取一个值，该值指示 Socket 是否绑定到特定本地端口。
	LingerState	获取或设置一个值，该值指定 Socket 在尝试发送所有挂起数据时是否延迟关闭套接字。
	LocalEndPoint	获取本地终结点。
	MulticastLoopback	获取或设置一个值，该值指定传出的多路广播数据包是否传递到发送应用程序。
	NoDelay	获取或设置 Boolean 值，该值指定流 Socket 是否正在使用 Nagle 算法。
 	OSSupportsIPv4	指示基础操作系统和网络适配器是否支持 Internet 协议第 4 版 (IPv4)。
 	OSSupportsIPv6	指示基础操作系统和网络适配器是否支持 Internet 协议第 6 版 (IPv6)。
	ProtocolType	获取 Socket 的协议类型。
	ReceiveBufferSize	获取或设置一个值，它指定 Socket 接收缓冲区的大小。
	ReceiveTimeout	获取或设置一个值，该值指定之后同步 Receive 调用将超时的时间长度。
	RemoteEndPoint	获取远程终结点。



















	<a href="#">SendBufferSize</a>	获取或设置一个值，该值指定 Socket 发送缓冲区的大小。
	<a href="#">SendTimeout</a>	获取或设置一个值，该值指定之后同步 <a href="#">Send</a> 调用将超时的时间长度。
	<a href="#">SocketType</a>	获取 Socket 的类型。
 	<a href="#">SupportsIPv4</a>	<b>已过时。</b> 获取一个值，该值指示在当前主机上 IPv4 支持是否可用并且已启用。
 	<a href="#">SupportsIPv6</a>	<b>已过时。</b> 获取一个值，该值指示 Framework 对某些已过时的 <a href="#">Dns</a> 成员是否支持 IPv6。
	<a href="#">Ttl</a>	获取或设置一个值，指定 Socket 发送的 Internet 协议 (IP) 数据包的生存时间 (TTL) 值。
	<a href="#">UseOnlyOverlappedIO</a>	指定套接字是否应仅使用重叠 I/O 模式。

[页首](#)




















# 方法















显示: ☒ 继承 ☒ 保护 

	名称	说明
	<a href="#">Accept</a>	为新建连接创建新的 Socket。
	<a href="#">AcceptAsync</a>	开始一个异步操作来接受一个传入的连接尝试。
	<a href="#">BeginAccept(AsyncCallback, Object)</a>	开始一个异步操作来接受一个传入的连接尝试。
	<a href="#">BeginAccept(Int32, AsyncCallback, Object)</a>	开始异步操作以接受传入的连接尝试并接收客户端应用程序发送的第一个数据块。
	<a href="#">BeginAccept(Socket, Int32, AsyncCallback, Object)</a>	开始异步操作以接受从指定套接字传入的连接尝试并接收客户端应用程序发送的第一个数据块。
	<a href="#">BeginConnect(EndPoint, AsyncCallback, Object)</a>	开始一个对远程主机连接的异步请求。
	<a href="#">BeginConnect(IPAddress, Int32, AsyncCallback, Object)</a>	开始一个对远程主机连接的异步请求。主机由 <a href="#">IPAddress</a> 和端口号指定。
	<a href="#">BeginConnect(IPAddress[], Int32, AsyncCallback, Object)</a>	开始一个对远程主机连接的异步请求。主机由 <a href="#">IPAddress</a> 数组和端口号指定。


















	<code>BeginConnect(String, Int32, AsyncCallback, Object)</code>	开始一个对远程主机连接的异步请求。主机由主机名和端口号指定。
	<code>BeginDisconnect</code>	开始异步请求从远程终结点断开连接。
	<code>BeginReceive(ICollection&lt;ArraySegment&lt;Byte&gt;&gt;, SocketFlags, AsyncCallback, Object)</code>	开始从连接的 Socket 中异步接收数据。
	<code>BeginReceive(ICollection&lt;ArraySegment&lt;Byte&gt;&gt;, SocketFlags, SocketError, AsyncCallback, Object)</code>	开始从连接的 Socket 中异步接收数据。
	<code>BeginReceive(Byte[], Int32, Int32, SocketFlags, AsyncCallback, Object)</code>	开始从连接的 Socket 中异步接收数据。
	<code>BeginReceive(Byte[], Int32, Int32, SocketFlags, SocketError, AsyncCallback, Object)</code>	开始从连接的 Socket 中异步接收数据。
	<code>BeginReceiveFrom</code>	开始从指定网络设备中异步接收数据。
	<code>BeginReceiveMessageFrom</code>	开始使用指定的 <code>SocketFlags</code> 将指定字节数的数据异步接收到数据缓冲区的指定位置，然后存储终结点和数据包信息。
	<code>BeginSend(ICollection&lt;ArraySegment&lt;Byte&gt;&gt;, SocketFlags, AsyncCallback, Object)</code>	将数据异步发送到连接的 Socket。
	<code>BeginSend(ICollection&lt;ArraySegment&lt;Byte&gt;&gt;, SocketFlags, SocketError, AsyncCallback, Object)</code>	将数据异步发送到连接的 Socket。
	<code>BeginSend(Byte[], Int32, Int32, SocketFlags, AsyncCallback, Object)</code>	将数据异步发送到连接的 Socket。
	<code>BeginSend(Byte[], Int32, Int32, SocketFlags, SocketError, AsyncCallback, Object)</code>	将数据异步发送到连接的 Socket。
	<code>BeginSendFile(String, AsyncCallback, Object)</code>	使用 <code>UseDefaultWorkerThread</code> 标志，将文件 <code>fileName</code> 发送到连接的 Socket 对象。
	<code>BeginSendFile(String, Byte[], Byte[], TransmitFileOptions, AsyncCallback, Object)</code>	将文件和数据缓冲区异步发送到连接的 Socket 对象。
	<code>BeginSendTo</code>	向特定远程主机异步发送数据。
	<code>Bind</code>	使 Socket 与一个本地终结点相关联。
	<code>CancelConnectAsync</code>	取消一个对远程主机连接的异步请求。
	<code>Close()</code>	关闭 Socket 连接并释放所有关联的资源。

	<code>Close(Int32)</code>	关闭 Socket 连接并释放与指定超时关联的所有资源以允许发送排队数据。
	<code>Connect(EndPoint)</code>	建立与远程主机的连接。
	<code>Connect(IPAddress, Int32)</code>	建立与远程主机的连接。主机由 IP 地址和端口号指定。
	<code>Connect(IPAddress[], Int32)</code>	建立与远程主机的连接。主机由 IP 地址的数组和端口号指定。
	<code>Connect(String, Int32)</code>	建立与远程主机的连接。主机由主机名和端口号指定。
	<code>ConnectAsync(SocketAsyncEventArgs)</code>	开始一个对远程主机连接的异步请求。
	<code>ConnectAsync(SocketType, ProtocolType, SocketAsyncEventArgs)</code>	开始一个对远程主机连接的异步请求。
	<code>Disconnect</code>	关闭套接字连接并允许重用套接字。
	<code>DisconnectAsync</code>	开始异步请求从远程终结点断开连接。
	<code>Dispose()</code>	释放由 Socket 类的当前实例占用的所有资源。
	<code>Dispose(Boolean)</code>	释放由 Socket 使用的非托管资源，并可根据需要释放托管资源。
	<code>DuplicateAndClose</code>	重复目标进程的套接字引用，并关闭此进程的套接字。
	<code>EndAccept(IAsyncResult)</code>	异步接受传入的连接尝试，并创建新的 Socket 来处理远程主机通信。
	<code>EndAccept(Byte[], IAsyncResult)</code>	异步接受传入的连接尝试，并创建新的 Socket 对象来处理远程主机通信。此方法返回包含所传输的初始数据的缓冲区。
	<code>EndAccept(Byte[], Int32, IAsyncResult)</code>	异步接受传入的连接尝试，并创建新的 Socket 对象来处理远程主机通信。此方法返回一个缓冲区，其中包含初始数据和传输的字节数。
	<code>EndConnect</code>	结束挂起的异步连接请求。
	<code>EndDisconnect</code>	结束挂起的异步断开连接请求。
	<code>EndReceive(IAsyncResult)</code>	结束挂起的异步读取。
	<code>EndReceive(IAsyncResult, SocketError)</code>	结束挂起的异步读取。

	EndReceiveFrom	结束挂起的、从特定终结点进行异步读取。
	EndReceiveMessageFrom	结束挂起的、从特定终结点进行异步读取。 此方法还显示有关数据包而不是 <a href="#">EndReceiveFrom</a> 的更多信息。
	EndSend(IAsyncResult)	结束挂起的异步发送。
	EndSend(IAsyncResult, SocketError)	结束挂起的异步发送。
	EndSendFile	结束文件的挂起异步发送。
	EndSendTo	结束挂起的、向指定位置进行的异步发送。
	Equals(Object)	确定指定的 <a href="#">Object</a> 是否等于当前的 <a href="#">Object</a> 。（继承自 <a href="#">Object</a> 。）
	Finalize	Socket 类使用的可用资源。（重写 <a href="#">Object.Finalize()</a> 。）
	GetHashCode	用作特定类型的哈希函数。（继承自 <a href="#">Object</a> 。）
	GetSocketOption(SocketOptionLevel, SocketOptionName)	返回指定的 Socket 选项的值，表示为一个对象。
	GetSocketOption(SocketOptionLevel, SocketOptionName, Byte[])	返回指定的 Socket 选项设置，表示为字节数组。
	GetSocketOption(SocketOptionLevel, SocketOptionName, Int32)	返回数组中指定的 Socket 选项的值。
	GetType	获取当前实例的 <a href="#">Type</a> 。（继承自 <a href="#">Object</a> 。）
	IOControl(Int32, Byte[], Byte[])	使用数字控制代码，为 Socket 设置低级操作模式。
	IOControl(IOControlCode, Byte[], Byte[])	使用 <a href="#">IOControlCode</a> 枚举指定控制代码，为 Socket 设置低级操作模式。
	Listen	将 Socket 置于侦听状态。
	MemberwiseClone	创建当前 <a href="#">Object</a> 的浅表副本。（继承自 <a href="#">Object</a> 。）
	Poll	确定 Socket 的状态。
	Receive(IList<ArraySegment<Byte>>)	从绑定的 Socket 接收数据，将数据存入接收缓冲区列表中。
	Receive(Byte[])	从绑定的 Socket 套接字接收数据，将数据存入接收缓冲区。

	<code>Receive(IList&lt;ArraySegment&lt;Byte&gt;&gt;, SocketFlags)</code>	使用指定的 <a href="#">SocketFlags</a> ，从绑定的 Socket 接收数据，将数据存入接收缓冲区列表中。
	<code>Receive(Byte[], SocketFlags)</code>	使用指定的 <a href="#">SocketFlags</a> ，从绑定的 Socket 接收数据，将数据存入接收缓冲区。
	<code>Receive(IList&lt;ArraySegment&lt;Byte&gt;&gt;, SocketFlags, SocketError)</code>	使用指定的 <a href="#">SocketFlags</a> ，从绑定的 Socket 接收数据，将数据存入接收缓冲区列表中。
	<code>Receive(Byte[], Int32, SocketFlags)</code>	使用指定的 <a href="#">SocketFlags</a> ，从绑定的 Socket 接收指定字节数的数据，并将数据存入接收缓冲区。
	<code>Receive(Byte[], Int32, Int32, SocketFlags)</code>	使用指定的 <a href="#">SocketFlags</a> ，从绑定的 Socket 接收指定的字节数，存入接收缓冲区的指定偏移量位置。
	<code>Receive(Byte[], Int32, Int32, SocketFlags, SocketError)</code>	使用指定的 <a href="#">SocketFlags</a> ，从绑定的 Socket 接收数据，将数据存入接收缓冲区。
	<code>ReceiveAsync</code>	开始一个异步请求以便从连接的 Socket 对象中接收数据。
	<code>ReceiveFrom(Byte[], EndPoint)</code>	将数据报接收到数据缓冲区并存储终结点。
	<code>ReceiveFrom(Byte[], SocketFlags, EndPoint)</code>	使用指定的 <a href="#">SocketFlags</a> 将数据报接收到数据缓冲区并存储终结点。
	<code>ReceiveFrom(Byte[], Int32, SocketFlags, EndPoint)</code>	使用指定的 <a href="#">SocketFlags</a> 将指定的字节数接收到数据缓冲区并存储终结点。
	<code>ReceiveFrom(Byte[], Int32, Int32, SocketFlags, EndPoint)</code>	使用指定的 <a href="#">SocketFlags</a> 将指定字节数的数据接收到数据缓冲区的指定位置并存储终结点。
	<code>ReceiveFromAsync</code>	开始从指定网络设备中异步接收数据。
	<code>ReceiveMessageFrom</code>	使用指定的 <a href="#">SocketFlags</a> 将指定字节数的数据接收到数据缓冲区的指定位置，然后存储终结点和数据包信息。
	<code>ReceiveMessageFromAsync</code>	开始使用指定的 <a href="#">SocketAsyncEventArgs.SocketFlags</a> 将指定字节数的数据异步接收到数据缓冲区的指定位置，并存储终结点和数据包信息。
	<code>Select</code>	确定一个或多个套接字的状态。



	<code>Send(IList&lt;ArraySegment&lt;Byte&gt;&gt;)</code>	将列表中的一组缓冲区发送到连接的 Socket。
	<code>Send(Byte[])</code>	将数据发送到连接的 Socket。
	<code>Send(IList&lt;ArraySegment&lt;Byte&gt;&gt;, SocketFlags)</code>	使用指定的 <a href="#">SocketFlags</a> ，将列表中的一组缓冲区发送到连接的 Socket。
	<code>Send(Byte[], SocketFlags)</code>	使用指定的 <a href="#">SocketFlags</a> 将数据发送到连接的 Socket。
	<code>Send(IList&lt;ArraySegment&lt;Byte&gt;&gt;, SocketFlags, SocketError)</code>	使用指定的 <a href="#">SocketFlags</a> ，将列表中的一组缓冲区发送到连接的 Socket。
	<code>Send(Byte[], Int32, SocketFlags)</code>	使用指定的 <a href="#">SocketFlags</a> ，将指定字节数的数据发送到已连接的 Socket。
	<code>Send(Byte[], Int32, Int32, SocketFlags)</code>	使用指定的 <a href="#">SocketFlags</a> ，将指定字节数的数据发送到已连接的 Socket（从指定的偏移量开始）。
	<code>Send(Byte[], Int32, Int32, SocketFlags, SocketError)</code>	从指定的偏移量开始使用指定的 <a href="#">SocketFlags</a> 将指定字节数的数据发送到连接的 Socket。
	<code>SendAsync</code>	将数据异步发送到连接的 Socket 对象。
	<code>SendFile(String)</code>	使用 <a href="#">UseDefaultWorkerThread</a> 传输标志，将文件 <code>fileName</code> 发送到连接的 Socket 对象。
	<code>SendFile(String, Byte[], Byte[], TransmitFileOptions)</code>	使用指定的 <a href="#">TransmitFileOptions</a> 值，将文件 <code>fileName</code> 和数据缓冲区发送到连接的 Socket 对象。
	<code>SendPacketsAsync</code>	将文件集合或者内存中的数据缓冲区以异步方法发送给连接的 Socket 对象。
	<code>SendTo(Byte[], EndPoint)</code>	将数据发送到指定的终结点。
	<code>SendTo(Byte[], SocketFlags, EndPoint)</code>	使用指定的 <a href="#">SocketFlags</a> ，将数据发送到特定的终结点。
	<code>SendTo(Byte[], Int32, SocketFlags, EndPoint)</code>	使用指定的 <a href="#">SocketFlags</a> ，将指定字节数的数据发送到指定的终结点。
	<code>SendTo(Byte[], Int32, Int32, SocketFlags, EndPoint)</code>	使用指定的 <a href="#">SocketFlags</a> ，将指定字节数的数据发送到指定终结点（从缓冲区中的指定位置开始）。
	<code>SendToAsync</code>	向特定远程主机异步发送数据。
	<code>SetIPProtectionLevel</code>	设置套接字的 IP 保护级别。



	<a href="#">SetSocketOption(SocketOptionLevel, SocketOptionName, Boolean)</a>	将指定的 Socket 选项设置为指定的 <a href="#">Boolean</a> 值。
	<a href="#">SetSocketOption(SocketOptionLevel, SocketOptionName, Byte[])</a>	将指定的 Socket 选项设置为指定的值，表示为字节数组。
	<a href="#">SetSocketOption(SocketOptionLevel, SocketOptionName, Int32)</a>	将指定的 Socket 选项设置为指定的整数值。
	<a href="#">SetSocketOption(SocketOptionLevel, SocketOptionName, Object)</a>	将指定的 Socket 选项设置为指定值，表示为对象。
	<a href="#">Shutdown</a>	禁用某 Socket 上的发送和接收。
	<a href="#">ToString</a>	返回表示当前对象的字符串。（继承自 <a href="#">Object</a> 。）

[页首](#)

## 备注

Socket 类为网络通信提供了一套丰富的方法和属性。Socket 类允许您使用 [ProtocolType](#) 枚举中所列出的任何一种协议执行异步和同步数据传输。

Socket 类对异步方法遵循 .NET Framework 命名模式。例如，同步的 [Receive](#) 方法对应于异步的 [BeginReceive](#) 和 [EndReceive](#) 方法。

如果应用程序在执行期间只需要一个线程，请使用下面的方法，这些方法适用于同步操作模式。

- 如果当前使用的是面向连接的协议（如 TCP），则服务器可以使用 [Listen](#) 方法侦听连接。[Accept](#) 方法处理任何传入的连接请求，并返回可用于与远程主机进行数据通信的 Socket。可以使用此返回的 Socket 来调用 [Send](#) 或 [Receive](#) 方法。如果要指定本地 IP 地址和端口号，请在调用 [Listen](#) 方法之前先调用 [Bind](#) 方法。如果您希望基础服务提供程序为您分配可用端口，请使用端口号 0。如果希望连接到侦听主机，请调用 [Connect](#) 方法。若要进行数据通信，请调用 [Send](#) 或 [Receive](#) 方法。
- 如果当前使用的是无连接协议（如 UDP），则根本不需要侦听连接。调用 [ReceiveFrom](#) 方法可接受任何传入的数据报。使用 [SendTo](#) 方法可将数据报发送到远程主机。

若要在执行过程中使用单独的线程处理通信，请使用下面的方法，这些方法适用于异步操作模式。

- 如果当前使用的是面向连接的协议（如 TCP），则可使用 Socket、[BeginConnect](#) 和 [EndConnect](#) 方法来连接侦听主机。通过使用 [BeginSend](#) 和 [EndSend](#) 方法，或者使用 [BeginReceive](#) 和 [EndReceive](#) 方法，可以进行异步数据通信。可以使用 [BeginAccept](#) 和 [EndAccept](#) 处理传入的连接请求。
- 如果您使用的是 UDP 等无连接协议，则可以使用 [BeginSendTo](#) 和 [EndSendTo](#) 来发送数据报，而使用 [BeginReceiveFrom](#) 和 [EndReceiveFrom](#) 来接收数据报。

如果对一个套接字执行多个异步操作，它们不一定按启动时的顺序完成。

当数据发送和数据接收完成之后，可使用 [Shutdown](#) 方法来禁用 Socket。在调用 [Shutdown](#) 之后，可调用

[Close](#) 方法来释放与 Socket 关联的所有资源。

通过 Socket 类，您可以使用 [SetSocketOption](#) 方法来配置 Socket。可以使用 [GetSocketOption](#) 方法来检索这些设置。

#### 注意

如果要编写相对简单的应用程序，而且不需要最高的性能，则可以考虑使用 [TcpClient](#)、[TcpListener](#) 和 [UdpClient](#)。这些类为 Socket 通信提供了更简单、对用户更友好的接口。

## 示例

下面的代码示例演示如何使用 Socket 类向 HTTP 服务器发送数据和接收响应。接收到整个页前，此示例将一直处于阻止状态。

**C#**

```
using System;
using System.Text;
using System.IO;
using System.Net;
using System.Net.Sockets;

public class GetSocket
{
    private static Socket ConnectSocket(string server, int port)
    {
        Socket s = null;
        IPHostEntry hostEntry = null;

        // Get host related information.
        hostEntry = Dns.GetHostEntry(server);

        // Loop through the AddressList to obtain the supported AddressFamily. This is to avoid
        // an exception that occurs when the host IP Address is not compatible with the address
        // (typical in the IPv6 case).
        foreach(IPAddress address in hostEntry.AddressList)
        {
            IPEndPoint ipe = new IPEndPoint(address, port);
            Socket tempSocket =
                new Socket(ipe.AddressFamily, SocketType.Stream, ProtocolType.Tcp);

            tempSocket.Connect(ipe);

            if(tempSocket.Connected)
            {
                s = tempSocket;
                break;
            }
        }
        else
```

```

        {
            continue;
        }
    }
    return s;
}

// This method requests the home page content for the specified server.
private static string SocketSendReceive(string server, int port)
{
    string request = "GET / HTTP/1.1\r\nHost: " + server +
        "\r\nConnection: Close\r\n\r\n";
    Byte[] bytesSent = Encoding.ASCII.GetBytes(request);
    Byte[] bytesReceived = new Byte[256];

    // Create a socket connection with the specified server and port.
    Socket s = ConnectSocket(server, port);

    if (s == null)
        return ("Connection failed");

    // Send request to the server.
    s.Send(bytesSent, bytesSent.Length, 0);

    // Receive the server home page content.
    int bytes = 0;
    string page = "Default HTML page on " + server + ":\r\n";

    // The following will block until the page is transmitted.
    do {
        bytes = s.Receive(bytesReceived, bytesReceived.Length, 0);
        page = page + Encoding.ASCII.GetString(bytesReceived, 0, bytes);
    }
    while (bytes > 0);

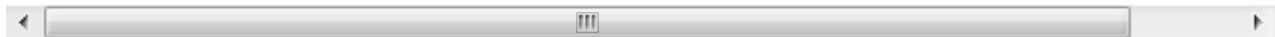
    return page;
}

public static void Main(string[] args)
{
    string host;
    int port = 80;

    if (args.Length == 0)
        // If no server name is passed as argument to this program,
        // use the current host name as the default.
        host = Dns.GetHostName();
    else
        host = args[0];

    string result = SocketSendReceive(host, port);
    Console.WriteLine(result);
}
}

```



## 版本信息

.NET Framework

受以下版本支持：4、3.5、3.0、2.0、1.1、1.0

.NET Framework Client Profile

受以下版本支持：4、3.5 SP1

## .NET Framework 安全性

- [SocketPermission](#)  
建立外向连接或接受传入请求。

## 平台

Windows 7, Windows Vista SP1 或更高版本, Windows XP SP3, Windows XP SP2 x64 Edition, Windows Server 2008 ( 不支持服务器核心 ), Windows Server 2008 R2 ( 支持 SP1 或更高版本的服务器核心 ), Windows Server 2003 SP2

.NET Framework 并不是对每个平台的所有版本都提供支持。有关支持的版本的列表，请参见[.NET Framework 系统要求](#)。

## 线程安全

此类的实例是线程安全的。

## 请参见

参考

[System.Net.Sockets](#) 命名空间

[System.Net](#)

[System.Net.Cache](#)

[System.Net.Security](#)

[System.Net.Sockets.SocketAsyncEventArgs](#)

其他资源

[Network Programming](#)

[Best Practices for System.Net Classes](#)

[Cache Management for Network Applications](#)

[Internet Protocol Version 6](#)

[Network Programming Samples](#)

[Networking Samples for .NET on MSDN Code Gallery \( MSDN 代码库上的 .NET 网络连接示例 \)](#)

[Network Tracing](#)  
[Security in Network Programming](#)  
[Socket Performance Enhancements in Version 3.5](#)  
[IPv6 Sockets Sample \( IPv6 套接字示例 \)](#)  
[Socket Performance Technology Sample \( 套接字性能技术示例 \)](#)

---

## 社区附加资源

---

© 2015 Microsoft