Connection Filter

English (United States) v1.6

Keywords: Connection Filter, Session Filter, Allow Connect

Connection Filter in SuperSocket is the interface which is used for filtering client connections. By connection filter, you can allow or disallow the client connections from the specified source.

Connection Filter interface is defined like below:

```
/// <summary>
/// The basic interface of connection filter
/// </summary>
public interface IConnectionFilter
{
    /// <summary>
    /// Initializes the connection filter
    /// </summary>
    /// <param name="name">The name.</param>
    /// <param name="appServer">The app server.</param>
    /// <returns></returns>
    bool Initialize(string name, IAppServer appServer);

    /// <summary>
    /// Gets the name of the filter.
    /// </summary>
    string Name { get; }

    /// <summary>
    /// Whether allows the connect according the remote endpoint
    /// </summary>
    /// <param name="remoteAddress">The remote address.</param>
    /// <returns></returns>
    bool AllowConnect(IPEndPoint remoteAddress);
}
```

bool Initialize(string name, IAppServer appServer);

This method is used to initialize the connection filter, name is the name of Filter.

string Name {get;}

Return Filter name

bool AllowConnect (IPEndPoint remoteAddress);

This method requires the client to achieve the endpoint to determine whether to allow connection to the server.

The following code implemented a connection filter which only allow connection from the specific ip range:

```
public class IPConnectionFilter : IConnectionFilter
{
    private Tuple<long, long>[] m_IpRanges;

    public bool Initialize(string name, IAppServer appServer)
    {
        Name = name;

        var ipRange = appServer.Config.Options.GetValue("ipRange");

        string[] ipRangeArray;

        if (string.IsNullOrEmpty(ipRange)
            || (ipRangeArray = ipRange.Split(new char[] { ',', ';' }, StringSplitOptions.RemoveEmptyEntries)).Length <= 0)
        {
            throw new ArgumentException("The ipRange doesn't exist in configuration!");
        }

        m_IpRanges = new Tuple<long, long>[ipRangeArray.Length];

        for (int i = 0; i < ipRangeArray.Length; i++)
        {
            var range = ipRangeArray[i];
            m_IpRanges[i] = GenerateIpRange(range);
        }

        return true;
    }

    private Tuple<long, long> GenerateIpRange(string range)
    {
        var ipArray = range.Split(new char[] { '-' }, StringSplitOptions.RemoveEmptyEntries);

        if(ipArray.Length != 2)
            throw new ArgumentException("Invalid ipRange exist in configuration!");

        return new Tuple<long, long>(ConvertIpToLong(ipArray[0]), ConvertIpToLong(ipArray[1]));
    }

    private long ConvertIpToLong(string ip)
    {
        var points = ip.Split(new char[] { '.' }, StringSplitOptions.RemoveEmptyEntries);

        if(points.Length != 4)
            throw new ArgumentException("Invalid ipRange exist in configuration!");

        long value = 0;
        long unit = 1;

        for (int i = points.Length - 1; i >= 0; i--)
        {
            value += unit * points[i].ToInt32();
            unit *= 256;
        }

        return value;
    }

    public string Name { get; private set; }

    public bool AllowConnect(IPEndPoint remoteAddress)
    {
        var ip = remoteAddress.Address.ToString();
        var ipValue = ConvertIpToLong(ip);

        for (var i = 0; i < m_IpRanges.Length; i++)
        {
            var range = m_IpRanges[i];

            if (ipValue > range.Item2)
                return false;

            if (ipValue < range.Item1)
                return false;
        }

        return true;
    }
}
```

Then you need to update the configuration file to use this connection filter:

1) add configuration node "connectionFilters";

```
<connectionFilters>
  <add name="IpRangeFilter"
       type="SuperSocket.QuickStart.ConnectionFilter.IPConnectionFilter, SuperSocket.QuickStart.ConnectionFilter" />
</connectionFilters>
```

2) add configuration attributes for server instance;

```
<server name="EchoServer"
        serverTypeName="EchoService" ip="Any" port="2012"
        connectionFilter="IpRangeFilter"
        ipRange="127.0.1.0-127.0.1.255">
</server>
```

3) the finally configuration should look like;

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <configSections>
        <section name="superSocket" type="SuperSocket.SocketEngine.Configuration.SocketServiceConfig, SuperSocket.SocketEngine"/>
    </configSections>
    <appSettings>
        <add key="ServiceName" value="EchoService"/>
    </appSettings>
    <superSocket>
        <servers>
            <server name="EchoServer"
                serverTypeName="EchoService"
                ip="Any" port="2012"
                connectionFilter="IpRangeFilter"
                ipRange="127.0.1.0-127.0.1.255">
            </server>
        </servers>
       <serverTypes>
           <add name="EchoService"
                type="SuperSocket.QuickStart.EchoService.EchoServer, SuperSocket.QuickStart.EchoService" />
       </serverTypes>
       <connectionFilters>
           <add name="IpRangeFilter"
                type="SuperSocket.QuickStart.ConnectionFilter.IPConnectionFilter, SuperSocket.QuickStart.ConnectionFilter" />
       </connectionFilters>
    </superSocket>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0" />
    </startup>
</configuration>
```


- Prev: Command Filter (/v1-6/en-US/Command-Filter)
- Next: Multiple Listeners (/v1-6/en-US/Multiple-Listeners)