

# Remoting, events and custom encryption channel sinks

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.remoting/2005-10/msg00153.html>

---

- *From:* "Egmont Gabler" <[info@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:info@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Tue, 25 Oct 2005 18:21:05 +0200
- 

I have written an API which defines a standardized interface for various types of hardware devices that can be used by .NET applications. The interface supports both local and remote device services in order to access e.g. a network printer as easily as a local printer.

On command completion, the remote device services fire events asynchronously to the client application transporting customized result objects over the wire.

The architecture is as usual:

- a server app hosts the remotable device service(s)
- the client app connects to a remote device service which is hosted by the server app as server activated singleton instance
- the client app accesses functions on the remote device services by transparent proxies
- the completion events of a device service send the serialized result objects back to the client

Tcp channels are defined on both sides (client and server) using standard tcp channel sinks. The asynchronous callbacks is done by multicast delegates. Since .NET framework v1.1 needs full trust when deserializing server side runtime types, the typeFilterLevel="Full" attribute is used.

Here how the server config looks like:

```
<configuration>
</appSettings>
<system.runtime.remoting>
<customErrors mode="on"/>
<application>
<service>
<activated type="... />
</service>

<channels>
<channel ref="tcp" port="8002">
<serverProviders>
```

## Remoting, events and custom encryption channel sinks

```
<formatter ref = "binary" typeFilterLevel= "Full" />
</serverProviders>
</channel>
</channels>
</application>
</system.runtime.remoting>
</configuration>
```

The client app instantiates a tcp channel with a given port number because it wants to receive the async events from the server on a definite channel port number. (During receiving of the incoming events the client app behaves temporarily as a server and the remote device service behaves as the client.)

So the client config file looks like this (also nothing unusual here):

```
<configuration>
</appSettings>
<system.runtime.remoting>
<application>
<client url="..." activated type="..." />

<channels>
<channel ref="tcp" port="8001">
<cl
```