

XF.Server Component

**Implement High Performance .NET
Application Servers**

Introduction

[XF.Server component](#) helps to create high performance TCP/IP application servers using .NET platform. The component is implemented using the most effective network programming model – I/O completion ports (IOCP) and thread pool.

Probably you have already checked the performance of native .NET Socket class. It can be used to create a simple client-server solution, but not more. Synchronous model is not scalable, asynchronous model spawns threads on each new client connection which is not acceptable for a high performance server, because context switching becomes the most expensive operation. Both of them use lots of CPU and memory resources. The following articles address these issues:

<http://msdn.microsoft.com/msdnmag/issues/07/09/Networking/Default.aspx>
<http://msdn.microsoft.com/msdnmag/issues/05/10/ConcurrentAffairs/>

Even though Microsoft claims that those problems were fixed in .NET 3.5, you can compare the performance results of XF.Server and .NET Socket class to see a huge difference. Moreover you can [download benchmark project sources](#), make code review and run your own test experiment.

XF.Server component minimizes transitions from managed to unmanaged code because its core server functionality is implemented using unmanaged code. Managed .NET layer works as a proxy that routes messages (method calls, events). XF.Server component will stay unbeatable until the servers are implemented using native .NET classes, because they involve too many managed unmanaged code transitions.

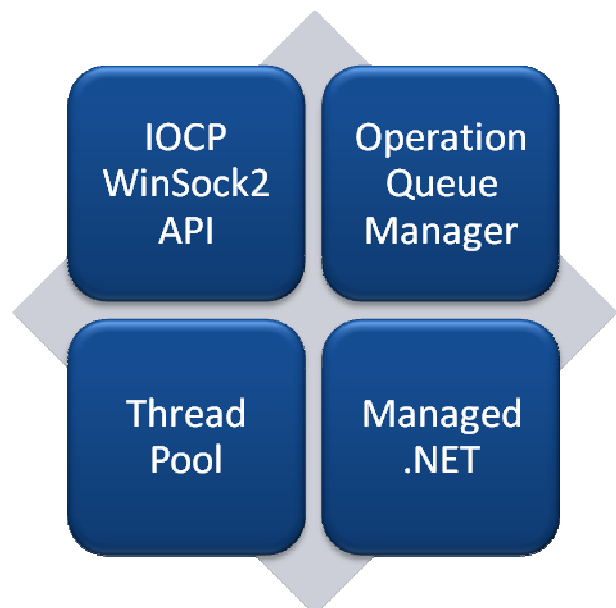
Priority Based Operations Processing Feature

XF.Server component provides you with an ability to create a server application that processes operation according to its priority. This feature allows you to process a request with higher priority level immediately, even if the server is heavily loaded and average response time is about 3-5 seconds. Definitely priority based operation processing feature will help to create responsive servers, which can process time critical requests immediately.

Architecture Overview

Main parts of the XF.Server architecture can be seen on the picture.

- Server component uses system level API (IOCP, WinSock2) to process network I/O
- Operation Queue Manager controls the order in which the operations are processed according to their priorities.
- Thread pool contains worker threads which process the requests. The number of running threads is usually equal to the number of processors, but can be higher for some time, if one of the running threads was blocked.
- Managed .NET layer, routes messages between server business logic in .NET and server core.



This documentation is still in progress, please refer to it later.

You can ask your questions or write about incorrect information in the document using the [support forum](#)