

User's Guide

Multiprocessor C#

.net Platform based

This document describes the MC#.Cluster platform for Windows Operating Systems. It also contains information on how to use cluster management utilities like **mcsboot** and **mcs halt** and shows how to compile and run programs written in MC# language.

System requirements

Microsoft .Net 2.0, Microsoft Installer 3.1

Supported operating systems

Theoretically it should be run on all kind of Windows platforms that support Microsoft.Net 2.0
Tested under the following operating systems:

- ✓ Windows Server 2003 Standard Edition
- ✓ Windows XP Home Edition
- ✓ Windows XP Professional Edition

Check your version

Before the installation of this system please make sure that you have the latest version of distributives. You can download latest version on the project site in **Downloads** section:

<http://u.pereslavl.ru/~vadim/MCSharp/>

System requirements

Your feedbacks about the project as well as comments, suggestions and ideas are greatly appreciated. You can send them through our site: <http://u.pereslavl.ru/~vadim/MCSharp/>

If you have some problems with the system or its installation please contact directly with developers of the system:

Yury P. Serdyuk

Yury@serdyuk.botik.ru

ICQ: #330402764

Compilers

Vadim B. Guzev

vadim@u.pereslavl.ru

ICQ: #62951762

Runtimes, Web site & Setup projects

Your feedbacks really help us to improve the system.

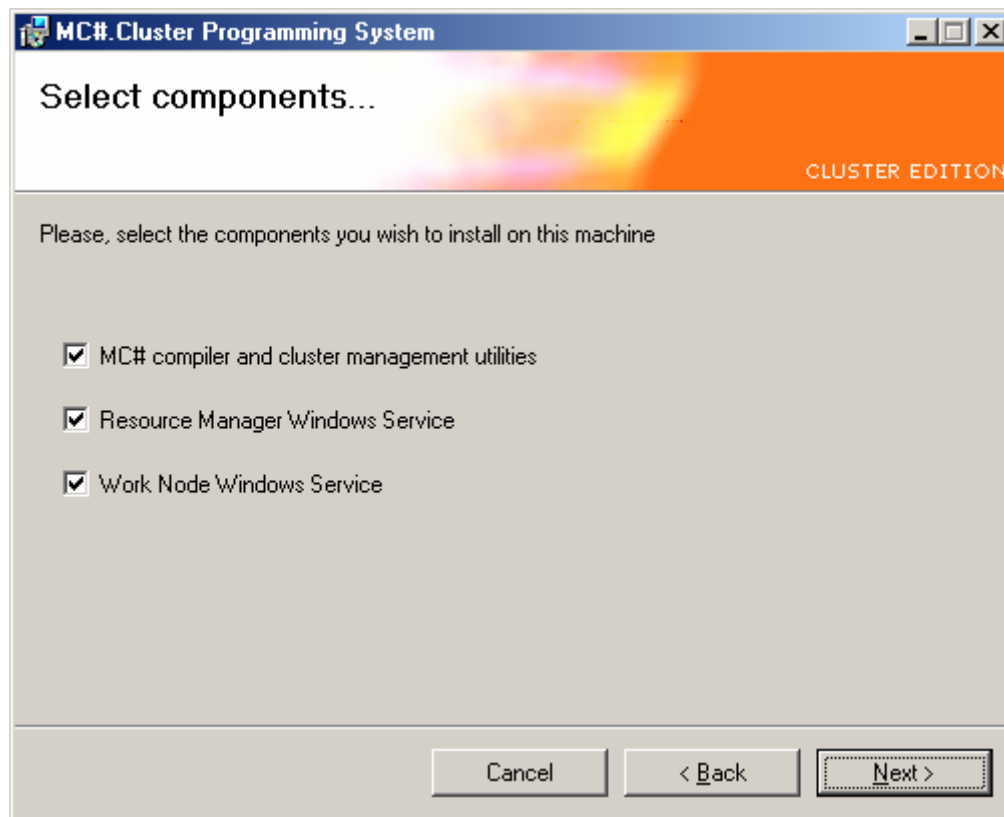
Installation

The **MC#.Cluster** system can be installed by running installation package **MC#.Cluster Programming System**.

Note:

While executing in distributed mode runtime system is using server sockets with random port numbers starting from 10000 till 50000. Before the installations make sure you're on your machines firewalls are either disabled or configured correspondingly.

During the installation you will be asked to choose which components should be installed:



1. **MC# compiler and cluster management utilities** - this component contains cluster management utilities, MC# compiler, documentation and some code examples. This component must be installed on the same node as **MC#.Cluster - Resource Manager** package, i.e. on the frontend of the cluster. If you just want to try MC# samples in local mode then you don't need to install two other packages.
2. **Resource Manager Windows Service** - this component must be installed on the frontend machine of the cluster. After the installation of this component the new Windows-service called **MC#.Cluster - Resource Manager** will be added to the system. This service is responsible for load-balancing of the cluster. Don't start it manually - instead use **mcsboot**, **mcs halt** and **mcsnodes** utilities.

3. **Work Node Windows Service** - this package must be installed on all other nodes of the cluster (although you can still install it on front-end machine). After the installation is completed the special Windows-service will be installed called **MC#.Cluster - Work Node** and it will be started automatically (it will also start automatically when the system reboots).

To install the MC#.Cluster system please run the installer on the nodes specified above and follow the instructions of the installer.

NOTE:

You can create clusters from any Windows machines in the same network that has installed the same version of Microsoft.Net (2.0) platform. The only limitation is that you can run MC# programs on cluster only from the disks that are mapped in the same way on all machines of the cluster (including frontend). I.e. that means that in order to run program "A.exe" that is located in folder "C:\MyFolderPath\" on frontend it must be also located in the same folder on all other nodes of the cluster! You can either create copies of your program on all machines in the same folders of the cluster or you can create a shared folder or use some other mechanisms of folders synchronization.

If you need more information on how to setup shared folder please refer to the document "*{Installation Folder}\Programmer's Guide\en.Setting up shared folder.pdf*", which must be available on your system after installation of **MC#.Cluster - Compiler & Utils** package.

Programs, compiled with MC# compiler can be executed either autonomously (in local mode) or in distributed mode on the cluster. In the second case you should specify the number of processors - with a special key ``/np'` (or ``-np'`).

By default, user's programs are executed in local mode, i.e. as simple Windows .exe applications. In this case it isn't necessary to initialize the cluster. It can be very useful for writing and debugging applications locally, before you launch them on productions servers.

If you want to run the program on multiple machines you need to initialize the cluster.

Initialization of Runtime

After installation you'll be able to find a special **mcsboot** utility which must be used for cluster initialization. It has only one optional parameter - the name of the file, which contains the names of the cluster nodes (each node must be on the separate line). By default if you don't specify the file with nodes only the current node will be included in the cluster.

mcsboot usage example:

```
mcsboot c:\Path\To\File\With\Nodes
```

or

```
mcsboot
```

In the second case only the local host will be included into the cluster.

Make sure that **MC#.Cluster - Work Node** service is started on all machines specified in the nodes list passed to **mcsboot** utility. Only nodes with working **MC#.Cluster - Work Node** services will be included in the list of cluster nodes. If there is no node in the list with running work node service then the initialization of the cluster is considered to be failed.

Finalization of Runtime

When you finished your work with the cluster, you should finalize the Runtime on the nodes and frontend. Command **mcs halt** is dedicated for this task. In case when you need to stop cluster where some applications are being executed, option ``-force'` should be used.

Additional tools

To find out which nodes are being used by MC# Runtime now use **mcsnodes** command. It prints the list of nodes.

Compilation of programs written in MC#

In this package there is a compiler called **mcs c** for compilation of MC# programs. There are two stages of compilation:

- ✓ MC#-code is translated into C# code
- ✓ C# code is compiled by **csc** compiler (from .Net platform).

Note:

| All parameters for **mcs c** compiler are passed to C# compiler (csc)

Examples of usage:

```
mcsc fib.mcs
mcsc fib.mcs /out:myfib.exe
mcsc /t:library fib.mcs
```

You can compile MC# programs into DLLs

```
mcsc a.mcs /t:library /out:a.dll
```

Then you can include it into the program

```
mcsc /r:a.dll b.mcs
```

You can mix MC# and C# files

```
mcsc a.mcs b.cs
```

Or compile several MC# files at once

```
mcsc a.mcs b.mcs
```

Starting user's compiled programs

Local mode:

```
fib.exe 35
```

This will start program in local mode, i.e. movable methods will be executed as new threads. No network operations should occur.

Distributed mode:

```
fib.exe 35 /np 2
```

This will start program in distributed mode on 2 processors. In this case movable methods will be transferred across the network. You can even specify the number of nodes that is larger than the number of physical nodes/processors. In this case on some nodes of the cluster numerous copies of processes will be started.

In case when you need to display the collected statistics you can use ``/withstats'` key

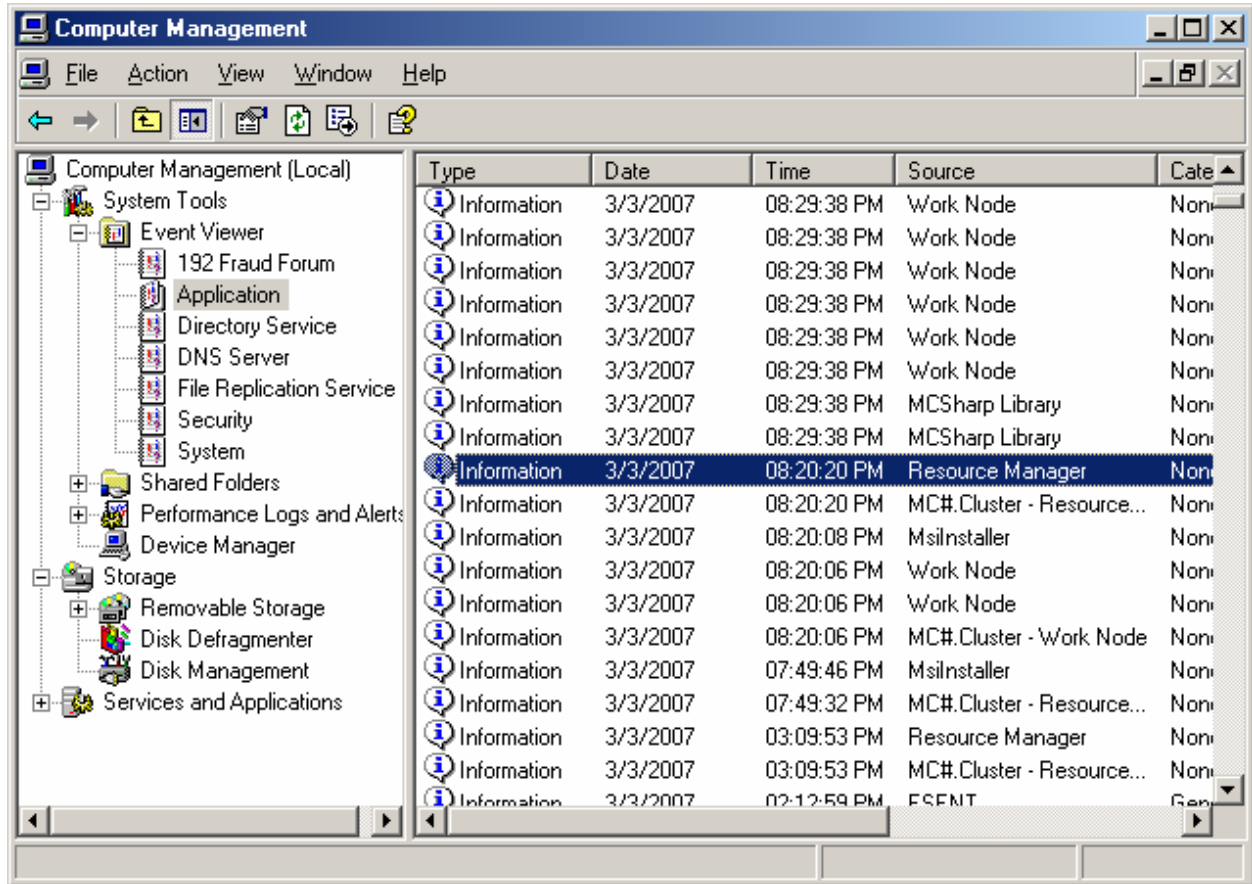
```
C:\Program Files\MC# Group\MC#.Cluster Programming Sys-
tem\examples\fib>Fib.exe 35 /np 2 /withstats
For n = 35 value is 14930352

=====
==MC# Statistics=====
Number of movable calls: 41
Number of channel messages: 41
Number of movable calls (across network): 15
Number of channel messages (across network): 15
Total size of movable calls (across network): 6359 bytes
Total size of channel messages (across network): 2954 bytes
Total time of movable calls serialization: 00:00:00.0156250
Total time of channel messages serialization: 00:00:00
Total size of transported messages: 10685 bytes
Total time of transporting messages: 00:00:00.0312500
Session initialization time: 00:00:02.5156250 / 2.515625 sec. / 2515.625
msec.
Total time: 00:00:03.0781250 / 3.078125 sec. / 3078.125 msec.
Number of movable calls (mc) per node:
  1. 2 x pcl | ***** 41 mc 100.00%
--- 1 of 1 nodes were used ---
```

When you are executing your program on several nodes the copy of the program is launched on these nodes with some additional parameters. These parameters notify the runtime system that method Main shouldn't be called on this node. When this parameter is specified the first thing that program does is notifying the Resource Manager that the program is now running and ready to receive commands. By default Runtime System is working in mode where execution of Main method on the client side begins when at least one of cluster nodes is initialized. But sometimes there are situations when you need to wait before the application is initialized on all nodes before method Main starts executing on client side (for example to uniformly distribute the load of the cluster). In this case you should use key ``/completeboot'`.

In case when you faces with some problems in Runtime System you can enable debug mode with the help of key ``/withdebug'` – this will help to see the messages being transferred between client application and resource manager.

All messages about exceptions which occur on nodes of the cluster as well as in Resource Manager Service you can find in Application event log of the given node. You can find it in the following way: right click on My Computer icon → Manage → Computer Management (Local) → System Tools → Event Viewer → Application.



By default the debugging information of resource manager and work node is disabled. You can enable it by changing the **IsDebug** parameter in corresponding configuration file in **bin** folder. Perhaps after that you'll need to restart reconfigured components and/or the whole cluster.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="IsDebug" value="true" />
    <add key="RMPort" value="25001" />
    <add key="WNPport" value="25002" />
  </appSettings>
</configuration>
```