

# Nagios Web Service Checker

## V1.2

Document version v1.2.2

### Table of content

1. Introduction .....	3
1.1. What is Nagios? .....	3
1.2. What is Nagios Web Service Checker? .....	3
1.3. Why should I use it? .....	3
1.4. What do I need?.....	3
2. Installing Nagios Web Service Checker .....	4
2.1. Installing the Webservice.....	4
2.2. Checking your installation of the web-service .....	4
2.3. Installing the PERL-Script .....	4
2.4. Adding definitions for Nagios .....	5
2.5. Additional configuration issues.....	5
3. Webservice check methods.....	6
3.1. cpu.....	6
3.2. disk .....	6
3.3. disks .....	6
3.4. File Age <sup>v1.2</sup> .....	7
3.5. FileCount <sup>v1.2</sup> .....	8
3.6. FileSize <sup>v1.2</sup> .....	8
3.7. generic_WMI.....	9
3.8. memory .....	9
3.9. ntevent.....	10
3.10. ntevents.....	11
3.11. process .....	11
3.12. services.....	11
3.13. uptime.....	12
4. Performance data <sup>v1.2</sup> .....	12
5. Config options in Web.Config .....	12
5.1. Authentication <sup>v1.2</sup> .....	12
5.1.1. Default user <sup>v1.2</sup> .....	13
5.1.2. Default user (old style).....	13
5.1.3. User for a specific server .....	13
5.1.4. User for a specific domain <sup>v1.2</sup> .....	13
5.1.5. Application pool identity of IIS <sup>v1.2</sup> .....	14

5.2.	Access for the generic WMI function .....	14
5.3.	Behaviour of check "services" .....	14
5.4.	Behaviour of check "uptime" .....	14
6.	Configuration samples.....	15
6.1.	host- & service definition issues .....	15
7.	Contributions .....	16
8.	Other Stuff .....	16

# 1. Introduction

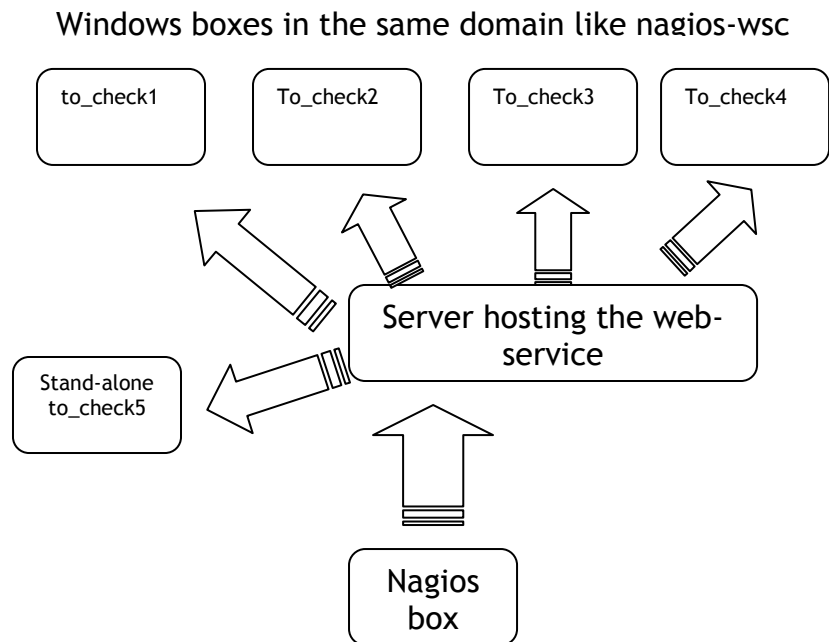
## 1.1. What is Nagios?

Nagios is a open source, host, network and service monitor that runs under UNIX or Linux. For more information, please check <http://www.nagios.org/>

## 1.2. What is Nagios Web Service Checker?

Nagios Web Service Checker (or nagios-wsc) is a utility I've written made up of two parts.

- The first part is a .NET based web service written in VB.NET that is capable of querying other windows machines on behalf of Nagios, using a Microsofts technology *Windows Management Instrumentation* (or WMI)
- The second part is a perl plug-in that allow Nagios to 'talk to' the Web Service and get the results back.



Picture 1

## 1.3. Why should I use it?

To my knowledge, you would previously have had to install a piece of software or service on each Windows box you want to query. If you have lots of windows servers, then this can be pain to install on each of these.

Using nagios-wsc, you only need to install one piece of software on an IIS web server with the .Net framework installed - if you've got lots of windows servers, one of them somewhere will be able to run this software.

## 1.4. What do I need?

To make best use of this software, you'll need:

- Nagios running on a UNIX/Linux box
- A windows machine running (at least) IIS 5 and .NET 1.1 framework installed.
- A username/password that will be allowed access to query the WMI information on each server.

## 2. Installing Nagios Web Service Checker

### 2.1. *Installing the Webservice*

Go to <http://www.sourceforge.net/projects/nagios-wsc> and download the latest versions of nagios-wsc and nagios-wsc-plugin.

Firstly, we'll set up the web service:

- Extract the contents of the nagios-wsc.vn.n.zip file to a local directory.
- Create a virtual directory on the IIS server that points to the nagios folder just extracted - ensure that it is created as an application.
- Consider very carefully restricting access to this virtual directory because nagios-wsc can return lots of information, which will be very useful in the hands of the wrong person. Try to limit the access to this directory by denying access to anyone except the internal IP address of your querying host(s). Do this on the "Directory security" tab and enter appropriate details in the "IP Address and Domain Name restrictions"
- In order to use WMI to retrieve information from other servers, you'll need a valid username/password. You need to enter this in the web.config file (near the bottom) in the nagios folder.

### 2.2. *Checking your installation of the web-service*

Use your favorite browser and navigate to the URL where the service is installed.

- Test the installation by using the following link (default URL):  
[http://name\\_of\\_your\\_server/nagios/service1.asmx](http://name_of_your_server/nagios/service1.asmx).  
With any luck, you'll get a list of methods (or things you can query) being displayed. (If this doesn't happen - report the problem in the forums on the <http://www.sourceforge.net/projects/nagios-wsc> project page and I'll try and help you out.)
- Choose the "disks" method. On the next screen, you enter "localhost" as server name and in the param field, type "1000,250". Finally, click the Invoke button. All being well you should get some XML that contains the result of the query.

That should conclude the setting up of the nagios-wsc web service. Now we need to set up Nagios to talk to it.

### 2.3. *Installing the PERL-Script*

Put the check\_wsc.pl file from the nagios-wsc-plugin zip file into the nagios libexec folder with the other plug-ins. You also can keep the perl-script in any other place. If you prefer to do so, you'll have to change the definition in Nagios checkcommand.cfg (see below).

## 2.4. Adding definitions for Nagios

### Checkcommands.cfg

Add the following lines in checkcommands.cfg: (command\_line should be all on one single line)

```
define command {
    command_name check_wsc
    command_line $USER1$/check_wsc.pl -H $HOSTADDRESS$ -r
    server_hosting_.net_service -p $ARG2$ -t $ARG1$
}
```

If you installed the perl script in a different location than the standard plug-ins then your definition should look like this:

```
define command {
    command_name check_wsc
    command_line <full_path_to_>/check_wsc.pl -H $HOSTADDRESS$ -r
    server_hosting_.net_service -p $ARG2$ -t $ARG1$
}
```

To choose a domain-authentication use the following form: **v1.2**

```
define command {
    command_name check_wsc
    command_line <full_path_to_>/check_wsc.pl -H '<domain>\$HOSTADDRESS$'
    -r server_hosting_.net_service -p $ARG2$ -t $ARG1$
}
```

e.g.

```
command_line $USER1$/check_wsc.pl -H 'MYDOMAIN\$HOSTADDRESS$' -r
server_hosting_.net_service -p '$ARG2$' -t '$ARG1$'
```

### services.cfg

Assume you have a host definition named “to\_check1”. Edit the services.cfg to make use this command just defined - example entries are show below:

```
Define_command {
    service_name      disks
    check_command     check_wsc!disks!
    host_name         to_check1
    ...
}
```

This above entry will check all fixed, local disks on the machine “to\_check1” for free disk space and warn if any have less than 1000MB free, or be critical if there is less than 250MB free. Details about the possible service checks see chapter 3 “Webservice check methods”.

## 2.5. Additional configuration issues

Chapter 6 (Configuration samples) describes some more issues of configuring Nagios and nagios-wsc.

## 3. Webservice check methods

### 3.1. *cpu*

This will compare the average cpu utilization and return a status based on the values entered. If you are running this check against a multiprocessor-system it will check the average load over all cpus.

```
check_wsc!cpu!"<warning_value>,<critical_value>"
```

All values are percent. If the average cpu load will rise above the corresponding values, the check will issue a warning or critical state.

### 3.2. *disk*

This can be used to check a single disk for the used disk space.

Size-value accept integer values and optional **v1.2**

k for kilo- (1000), K for Kilo- (1024), M for Mega-, G for Gigabytes, % for percentage  
Default: MegaBytes

#### Form 1

nagios-wsc will compare the available disk space of the disk specified with the values. If the disk has available disk space less than the warning value, it will return a warning. However, if the disk has available disk space less than the critical value, it will return a critical status.

```
check_wsc!disk!"drive_letter,warning_value[k|K|M|G],critical_value[k|K|M|G]"
```

#### Form 2

The second form is used to check the available disk space compared to the total one in percent.

```
check_wsc!disk!"drive_letter,warning_value%,critical_value%"  
values accept integer representing percent of total disk space
```

### 3.3. *disks*

nagios-wsc will compare the available disk space of all local fixed disks with the values. If any drive has available disk space less than the warning value, it will return a warning. However, if any drive has available disk space less than the critical value, it will return a critical status.

Size-value accept integer values and optional **v1.2**

k for kilo- (1000), K for Kilo- (1024), M for Mega-, G for Gigabytes, % for percentage  
Default: MegaBytes

#### Form 1

Values are integers representing Megabytes

```
check_wsc!disks!"warning_value,critical_value"  
e.g. disks!"1000,250"
```

## Form 2

Values are integers representing percentage of the available space compared to the total disk space.

```
check_wsc!disks!“warning_value%,critical_value%”  
e.g. disks!“80%,90%”
```

## Form 3

This form allows defining a list of disks which should be checked:

```
check_wsc!disks!“warning_value,critical_value,drive-letter:[|driveletter:]...”  
e.g. disks!10%,5%,D:|E:|F:|G:
```

This will check disks D,E,F,G. Warning threshold - 10% of a disk size, critical - 5% of a disk size. It can be used with the form for checking MB as.

## Form 4 <sup>v1.2</sup>

This form allows defining a list of disks which should be checked:

```
check_wsc!disks!“warning_value,critical_value,exclude:drive-  
letter:[|driveletter:]...”  
e.g. disks!10%,5%, exclude:C:|D:
```

This will check all disks except C and D. Warning threshold - 10% of a disk size, critical - 5% of a disk size. It can be used with the form for checking MB as.

This form is very useful if you do check a lot of drives and there's just one which doesn't fit the standard rule (e.g. backup disk).

## 3.4. File Age <sup>v1.2</sup>

This method checks the last modification date of files. The age of the file is compared checked against the specified time period. A warning or error state is returned if the file was not modified during this period. Optionally an “not OK” state is returned if the file was modified.

### Syntax

```
check_wsc!FileAge!“file-specification,timeperiod,options”  
e.g. FileAge!“c:\windows\temp\*.log,1d,/warn”
```

File-specification: Full path of the file, star convention for filename is allowed

timeperiode: <integer>m|h|d representing minutes, hours, days

options: /newer Warning/Error state if file is newer

or /older Warning/Error state if file was not modified within time period

/warn Issue a warning state (default: error state)

Examples:

```
check_wsc!FileAge!“c:\windows\temp\*.log,1d,/warn”
```

Warn if one of the files “c:\windows\temp\\*.log” is older than one day

```
check_wsc!FileAge!“c:\windows\temp\*.log,48h,/newer”
```

Issue a “State Critical” if one of the files “c:\windows\temp\\*.log” modified within the 48 hours.

## Compatibility

This check is currently not compatible with WMI on Windows2000 servers!

### 3.5. *FileCount*<sup>v1.2</sup>

This method checks the number of files in a directory.

#### Syntax

```
check_wsc!FileCount!“file-specification,warning,critical,options”
```

File-specification: Full path of the file, star convention for filename is allowed

Level: Positive values: Warning / critical if count is greater the values

Negative values: Warning / critical if count is below the values

Examples:

```
check_wsc! FileCount!“c:\windows\temp\*.log,2,5”
```

Warn if one more then 2 files of “c:\windows\temp\\*.log” exist, critical if more than 5.

```
check_wsc! FileCount!“c:\windows\temp\*.log,-10,-5”
```

Warning if less then 10 files of “c:\windows\temp\\*.log”, critical if less than 5.

## Compatibility

This check is currently not compatible with WMI on Windows2000 servers!

### 3.6. *FileSize*<sup>v1.2</sup>

This method checks the size of files.

#### Syntax

```
check_wsc!FileSize!“file-specification,timeperiod”
```

File-specification: Full path of the file, star convention for filename is allowed

Level: Positive values: Warning / critical if size is larger than the specified values

Negative values: Warning / critical if size is below the values

Size-value is accepting integer values and optional

k for kilo- (1000), K for Kilo- (1024), M for Mega-, G for Gigabytes

Default: MegaBytes

Examples:

```
check_wsc!FileSize!“c:\windows\temp\*.log,200K,500M”
```

Warn if one of the files “c:\windows\temp\\*.log” is larger than 200KB, critical if above 500MB.

```
check_wsc! FileSize!“c:\windows\temp\*.log,-1M,-500K”
```

Critical if one of the files “c:\windows\temp\\*.log” is smaller than 1M, critical if below 500KB.

## Compatibility

This check is currently not compatible with WMI on Windows2000 servers!

### 3.7. *generic\_WMI*

WMI\_Class: Specify a Win32\_xxx class to query. For a full list, check out the Microsoft site here:

[http://msdn.microsoft.com/library/default.asp?url=/library/enus/wmisdk/wmi/installed\\_applications\\_classes.asp](http://msdn.microsoft.com/library/default.asp?url=/library/enus/wmisdk/wmi/installed_applications_classes.asp). Select\_Fields: if left blank, all fields are returned; otherwise input a commaseparated list of fields to return.

Where\_clause: If left blank, all records are returned; otherwise specify a condition to apply to records nagios-wsc will check the win32 class entered and return the values required in as an xml document.

Example:

WMI\_Class: win32\_logicaldisk,

Select\_fields: filesystem

Where\_clause: drivetype=3

These parameters will return the filesystem of all local fixed disks

Note: Please take care with this as this proxy method is very powerful and can return lots of interesting information to anyone that is capable of invoking it. It will be useful to hackers if you have your nagios-wsc directory accessible to the internet.

To reduce this risk, there are two entries in the web.config file.

- “valid\_callers” will specify IP addresses of hosts that can invoke this particular method (comma-separated)
- “valid\_calls” specifies which WMI Classes can be invoked (again commaseparated )

## Remark

Currently this check is for testing purposes only. You need version v1.2 of the PERL-script *check\_wsc.pl* or higher to receive the XML result back to the script.

### 3.8. *memory*

This one checks memory and pagefile size. You can miss any parameters you want. For example, if you miss memwarning and memcritical, the method will check only virtual memory size.

Size-value accept integer values and optional <sup>v1.2</sup>  
k for kilo- (1000), K for Kilo- (1024), M for Mega-, G for Gigabytes, % for percentage  
Default: Kilobytes

```
check_wsc!Memory!("[memwarning], [memcritical], [vmemwarning],  
[vmemcritical]")
```

- memwarning, memcritical - thresholds for physical memory
- vmemwarning, vmemcritical - thresholds for virtual memory

Example: 50%, 499, 10, 50%.

### 3.9. ntevent

This is used to check the value which is returned by a special event entry.

```
check_wsc!Ntevent!(eventtypes, logfile, sourcenames, eventcodes,  
searchstring, period, type of returned state if found, type of returned  
state if not found,options)
```

- eventtypes - list of EventTypes (integer) to query, for example: "1|2" or just "3" (Type 1 = Critical, 2 = Warning, 3 = Informational).
- logfile - list of LogFiles to query, for example: "Application|System" or just "Security"
- sourcenames - source to query, for example: Service Control Manager|eventlog
- eventcodes - event codes to query, for example: 51|7023
- searchstring - string for searching in event body
- period - period in minutes to query
- type of returned state if found - state returned if some events found with query, i - OK, w - Warning, c - Critical, default is "c"
- type of returned state if not found - state returned if no events found with query, "i"
- options - /W2K search for string to be done compatible with WMI for hosts to be checked having a Windows 2000 operating system; Default. Windows 2003 style (using LIKE operator).

You can miss any parameters.

Just for example, in my nagios I'm using this method for checking 37 event from Ntfs (User hit their quota limit):

```
check_wsc!Ntevent!3,System,Ntfs,37,,5,w,i
```

### 3.10. ntevents

The method checks the appearance of NT Events for defined period of time.

The methods return status depends on most important returned event. For example if there would be some event with type "Error" than the method will return Critical, for Warning the method return Warning.

```
check_wsc!ntevents!(logfiles,sourcenames,eventcodes,period) "
```

- logfiles - list of LogFiles to query, for example: "Application|System" or just "Security"
- sourcenames - source to query, for example: Service Control Manager|eventlog
- eventcodes - event codes to query, for example: 51|7023
- period - period in minutes for which events will be queried, for example: 10

You can miss any parameters. For example, if you miss logfiles than the method will check all Log Files on remote system.

Sample: Common case, check for important events on Windows Server:

```
check_wsc!ntevents!System,Service Control Manager|NetBT|System  
Error|W32Time|Server|LDM|dmio|disk,,360
```

### 3.11. process

The method checks the number of processes running.

#### Form 1

The warning or critical state is raised if the count of the specified process is below the configured values.

```
Check_wsc!process!warning,critical
```

#### Form 2

```
Check_wsc!process
```

Using this syntax, the method just checks whether the process exists on the remote system.

Example: check\_wsc!process!svchost.exe!10,5

If svchost.exe is running at less than 10 times a warning state is raised, below 5 a critical state is issued.

### 3.12. services

This will check all services listed and return their status. If any services specified are recognized and not running, then a critical state will be returned. Note: specifying a service that does not exist on the server should not cause an error and will not cause a warning or critical state.

Note: When entering names of services to check for, you need to use the “system name”, rather than the descriptive name.

```
check_wsc!services!ListOfServices
```

ListOfServices is a comma-separated list of services to be checked (e.g. “alerter, w32time”). “\*” will check all services. This can be used for verifying the name of services.

For display options on the Nagios web-pages see section 5 “Config options in Web.Config”.

### 3.13. uptime

nagios-wsc will attempt to determine how long the server has been up for and compare to the values entered. The state returned will change if the current value is lower than the configured values.

```
Check_wsc!uptime!“warning_value,critical_value”
```

The values accept integers representing hours. For instance configuring “24,5” will check the uptime of a machine and warn if it’s been up for less than 24 hours or be critical if it’s been up for less than 5 hours.

For display options on the Nagios web-pages see section 5 “Config options in Web.Config”

## 4. Performance data <sup>v1.2</sup>

The following commands return performance data:

- cpu: average load over all cpus
- disk percentage of free space
- services services running, service not running
- process number of processes detected
- uptime number of hours

## 5. Config options in Web.Config

To grant access for nagios-wsc to the servers to be checked, you have to specify a user in the Web.Config file. All keys are located in the <appSettings>-section of </system.web>.

### 5.1. Authentication <sup>v1.2</sup>

Authentication can be configured in several ways:

- Standard authentication
- Authentication for a specific server

- Application pool identity of IIS (since v1.2)
- Authentication for domains (since v1.2)

Order in which the authentication is selected:

1. domain definition
2. server definition
3. default user
4. default user

### 5.1.1. Default user <sup>v1.2</sup>

Use the following key to specify the user which will be used to get access to remote machines.

```
<add key="default_auth" value="wsc-test:pwd" />
```

### 5.1.2. Default user (old style)

Use the following keys to specify the user which will be used to get access to remote machines.

```
<add key="sysusername" value="wsc-test" />
<add key="syspassword" value="pwd" />
```

This form is still supported.

### 5.1.3. User for a specific server

```
<add key="servername" value="user:password" />
    e.g. <add key="server_to_check1" value="UserAtServer:password" />
```

The name of the server must be written in the form which is sent during the request.

It corresponds to the parameter which is passed via the -H switch.

### 5.1.4. User for a specific domain <sup>v1.2</sup>

```
<add key="dom:domainname" value="user:password" />
    e.g. <add key="dom:MYDOMAIN" value="domainuser:password" />
```

The name of the domain and server must be written in the form which is sent during the request.

It corresponds to the parameter which is passed via the -H switch

For an example of a checkcommand using the domain-authentication see chapter 2.4 [Adding definitions for Nagios](#) above.

### 5.1.5. Application pool identity of IIS<sup>v1.2</sup>

The advantage of using the IIS6.0 feature of running an application pool with a specific user is that you do not have to configure defaults for username and password in the web.config file.

This is used when

- you do not specify a default authentication or default user
- the requested server to be checked is not defined in the web.config
- you do not ask for domain authentication

Then the check is executed with the permissions of the user, under which the web-service is running. How to define the application pool to run using a specific user can be read at

<http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/f05a7c2b-36b0-4b6e-ac7c-662700081f25.mspx> .

### 5.2. Access for the generic WMI function

These options define for which servers it is allowed to use the check “Generic WMI” as well as the parts of the WMI, which are allowed to be queried.

```
<add key="valid_callers" value="127.0.0.1" />
```

```
<add key="valid_calls" value="Win32_LogicalDisk,Win32_Service" />
```

„valid\_callers“ accepts IP-Addresses (seperated by comma), „valid\_calls“ a list of classes in the WMI which are permitted to be part of the query.

### 5.3. Behaviour of check “services”

```
<add key="ServicesOptions" value="warnmissing,smart" />
```

Possible values:

- “warnmissing” results in check warning if you requested a service to be checked, which is not defined on the remote server.
- “smart” does not output running services in the result section of nagios. Only services in a warning or critical state are displayed.

### 5.4. Behaviour of check “uptime”

```
<add key="UptimeOptions" value="smart" />
```

Possible values:

- “smart” - if state returned is OK only the current uptime is display in Nagios. Otherwise last boot time and local time on the server are displayed as well.

## 6. Configuration samples

### 6.1. *host- & service definition issues (IP vs. DNS)*

You might use one of the following forms to define your hosts in hosts.cfg:

```
Define_command {
    host_name    to_check1
    alias        alias_to_check1
    address      1.2.3.4
    ...
}
```

or

```
Define_command {
    host_name    to_check1
    alias        alias_to_check1
    address      DNS-name.domain
    ...
}
```

If your check-command and service-definition is defined like

```
define command {
    command_name check_wsc
    command_line $USER1$/check_wsc.pl -H $HOSTADDRESS$ -r
    server_hosting_.net_service -p $ARG2$ -t $ARG1$
}
Define_command {
    service_name    disks
    check_command check_wsc!disks!
    host_name    to_check1
    ...
}
```

the command executed by Nagios would be

```
<path_to_plugin>/check_wsc.pl -H 1.2.3.4 ...
server_hosting_.net_service -p '' -t 'disks'
```

or

```
<path_to_plugin>/check_wsc.pl -H DNS-name.domain ...
server_hosting_.net_service -p '' -t 'disks'
```

The first form does not rely on DNS-lookups.

Nagios and nagios-wsc will target their checks directly to the same IP-address. Even in a case of loss of connectivity to your DNS-server or impossibility of DNS-lookup inside the IP-segment of nagios-wsc, all checks can be well performed.

The later form relies on DNS-lookup which must resolve to a correct IP address in both cases.

This one even works in cases where the IP-address for accessing the server is different for Nagios and nagios-wsc. Then you will have to use this way, e.g. due to environments using NAT.

On the other hand it is not necessary to maintain the servers IP-address in several locations (DNS & host-config) in case of changes.

## Impacts to web.config for authentication for servers and domain

Because nagios-wsc uses the parameter given by the commandline-switch -H to lookup for non-default authentication, the way of defining your check-command and host also affects the way you have to define it in your web.config.

Using the IP-address from the example above has to result in an entry

```
<add key=" 1.2.3.4" value="user:password" />
```

whereas using the DNS-name variant has to lead to

```
<add key="DNS-name.domain" value="UserAtServer:password" />
```

## 7. Contributions

I want to thank the following people for contributing code and ideas to this project:

Mark Edgar, Frank J. Thomas, Konstantin Timokhin

## 8. Other Stuff

NAGIOS-WSC is a registered project at SourceForge.Net.

NAGIOS-WSC is published under GPL (general public license).

NAGIOS is a registered trademark of Ethan Galstad.

© 2007 by Wolfgang Wagner

© 2005 by Frank J. Thomas