```
##############################################################################
#####
#
# NAGIOS.CFG - Sample Main Config File for Nagios 3.5.1
#
# Read the documentation for more information on this configuration
# file.  I've provided some comments here, but things may not be so
# clear without further explanation.
#
# Last Modified: 12-14-2008
#
##############################################################################
#####


# LOG FILE
# This is the main log file where service and host events are logged
# for historical purposes.  This should be the first option specified
# in the config file!!!

log_file=/var/log/nagios/nagios.log



# OBJECT CONFIGURATION FILE(S)
# These are the object configuration files in which you define hosts,
# host groups, contacts, contact groups, services, etc.
# You can split your object definitions across several config files
# if you wish (as shown below), or keep them all in a single config file.

# You can specify individual object config files as shown below:
cfg_file=/etc/nagios/objects/commands.cfg
cfg_file=/etc/nagios/objects/contacts.cfg
cfg_file=/etc/nagios/objects/timeperiods.cfg
cfg_file=/etc/nagios/objects/templates.cfg
cfg_file=/etc/nagios/objects/hostgroup.cfg

# Definitions for monitoring the local (Linux) host
cfg_file=/etc/nagios/objects/localhost.cfg

# Definitions for monitoring a Windows machine
cfg_file=/etc/nagios/objects/windows.cfg

# Definitions for monitoring a router/switch
cfg_file=/etc/nagios/objects/hosts.cfg

# Definitions for monitoring a network printer
#cfg_file=/etc/nagios/objects/printer.cfg

# Definitions des services
cfg_file=/etc/nagios/objects/services.cfg

# You can also tell Nagios to process all config files (with a .cfg
# extension) in a particular directory by using the cfg_dir
# directive as shown below:

#cfg_dir=/etc/nagios/servers
#cfg_dir=/etc/nagios/printers
```

```
#cfg_dir=/etc/nagios/switches
#cfg_dir=/etc/nagios/routers

cfg_dir=/etc/nagios/conf.d



# OBJECT CACHE FILE
# This option determines where object definitions are cached when
# Nagios starts/restarts.  The CGIs read object definitions from
# this cache file (rather than looking at the object config files
# directly) in order to prevent inconsistencies that can occur
# when the config files are modified after Nagios starts.

object_cache_file=/var/log/nagios/objects.cache



# PRE-CACHED OBJECT FILE
# This options determines the location of the precached object file.
# If you run Nagios with the -p command line option, it will preprocess
# your object configuration file(s) and write the cached config to this
# file.  You can then start Nagios with the -u option to have it read
# object definitions from this precached file, rather than the standard
# object configuration files (see the cfg_file and cfg_dir options
above).
# Using a precached object file can speed up the time needed to (re)start
# the Nagios process if you've got a large and/or complex configuration.
# Read the documentation section on optimizing Nagios to find our more
# about how this feature works.

precached_object_file=/var/log/nagios/objects.precache



# RESOURCE FILE
# This is an optional resource file that contains $USERx$ macro
# definitions. Multiple resource files can be specified by using
# multiple resource_file definitions.  The CGIs will not attempt to
# read the contents of resource files, so information that is
# considered to be sensitive (usernames, passwords, etc) can be
# defined as macros in this file and restrictive permissions (600)
# can be placed on this file.

resource_file=/etc/nagios/private/resource.cfg



# STATUS FILE
# This is where the current status of all monitored services and
# hosts is stored.  Its contents are read and processed by the CGIs.
# The contents of the status file are deleted every time Nagios
#  restarts.

status_file=/var/log/nagios/status.dat
```

```
# STATUS FILE UPDATE INTERVAL
# This option determines the frequency (in seconds) that
# Nagios will periodically dump program, host, and
# service status data.

status_update_interval=10



# NAGIOS USER
# This determines the effective user that Nagios should run as.
# You can either supply a username or a UID.

nagios_user=nagios



# NAGIOS GROUP
# This determines the effective group that Nagios should run as.
# You can either supply a group name or a GID.

nagios_group=nagios



# EXTERNAL COMMAND OPTION
# This option allows you to specify whether or not Nagios should check
# for external commands (in the command file defined below).  By default
# Nagios will *not* check for external commands, just to be on the
# cautious side.  If you want to be able to use the CGI command interface
# you will have to enable this.
# Values: 0 = disable commands, 1 = enable commands

check_external_commands=1



# EXTERNAL COMMAND CHECK INTERVAL
# This is the interval at which Nagios should check for external
commands.
# This value works of the interval_length you specify later.  If you
leave
# that at its default value of 60 (seconds), a value of 1 here will cause
# Nagios to check for external commands every minute.  If you specify a
# number followed by an "s" (i.e. 15s), this will be interpreted to mean
# actual seconds rather than a multiple of the interval_length variable.
# Note: In addition to reading the external command file at regularly
# scheduled intervals, Nagios will also check for external commands after
# event handlers are executed.
# NOTE: Setting this value to -1 causes Nagios to check the external
# command file as often as possible.

#command_check_interval=15s
command_check_interval=-1
```

```
# EXTERNAL COMMAND FILE
# This is the file that Nagios checks for external command requests.
# It is also where the command CGI will write commands that are submitted
# by users, so it must be writeable by the user that the web server
# is running as (usually 'nobody').  Permissions should be set at the
# directory level instead of on the file, as the file is deleted every
# time its contents are processed.

command_file=/var/spool/nagios/cmd/nagios.cmd



# EXTERNAL COMMAND BUFFER SLOTS
# This settings is used to tweak the number of items or "slots" that
# the Nagios daemon should allocate to the buffer that holds incoming
# external commands before they are processed.  As external commands
# are processed by the daemon, they are removed from the buffer.

external_command_buffer_slots=4096



# LOCK FILE
# This is the lockfile that Nagios will use to store its PID number
# in when it is running in daemon mode.

lock_file=/var/run/nagios.pid



# TEMP FILE
# This is a temporary file that is used as scratch space when Nagios
# updates the status log, cleans the comment file, etc.  This file
# is created, used, and deleted throughout the time that Nagios is
# running.

temp_file=/var/log/nagios/nagios.tmp



# TEMP PATH
# This is path where Nagios can create temp files for service and
# host check results, etc.

temp_path=/tmp



# EVENT BROKER OPTIONS
# Controls what (if any) data gets sent to the event broker.
# Values:  0      = Broker nothing
#         -1      = Broker everything
#         <other> = See documentation

event_broker_options=-1
```

```
# EVENT BROKER MODULE(S)
# This directive is used to specify an event broker module that should
# by loaded by Nagios at startup.  Use multiple directives if you want
# to load more than one module.  Arguments that should be passed to
# the module at startup are seperated from the module path by a space.
#
#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
# WARNING !!! WARNING !!! WARNING !!! WARNING !!! WARNING !!! WARNING
#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#
# Do NOT overwrite modules while they are being used by Nagios or Nagios
# will crash in a fiery display of SEGFAULT glory.  This is a
bug/limitation
# either in dlopen(), the kernel, and/or the filesystem.  And maybe
Nagios...
#
# The correct/safe way of updating a module is by using one of these
methods:
#    1. Shutdown Nagios, replace the module file, restart Nagios
#    2. Delete the original module file, move the new module file into
place, restart Nagios
#
# Example:
#
#   broker_module=<modulepath> [moduleargs]

#broker_module=/somewhere/module1.o
#broker_module=/somewhere/module2.o arg1 arg2=3 debug=0



# LOG ROTATION METHOD
# This is the log rotation method that Nagios should use to rotate
# the main log file. Values are as follows..
#     n     = None - don't rotate the log
#     h     = Hourly rotation (top of the hour)
#     d     = Daily rotation (midnight every day)
#     w     = Weekly rotation (midnight on Saturday evening)
#     m     = Monthly rotation (midnight last day of month)

log_rotation_method=d



# LOG ARCHIVE PATH
# This is the directory where archived (rotated) log files should be
# placed (assuming you've chosen to do log rotation).

log_archive_path=/var/log/nagios/archives



# LOGGING OPTIONS
# If you want messages logged to the syslog facility, as well as the
# Nagios log file set this option to 1.  If not, set it to 0.

use_syslog=1
```

```
# NOTIFICATION LOGGING OPTION
# If you don't want notifications to be logged, set this value to 0.
# If notifications should be logged, set the value to 1.

log_notifications=1



# SERVICE RETRY LOGGING OPTION
# If you don't want service check retries to be logged, set this value
# to 0.  If retries should be logged, set the value to 1.

log_service_retries=1



# HOST RETRY LOGGING OPTION
# If you don't want host check retries to be logged, set this value to
# 0.  If retries should be logged, set the value to 1.

log_host_retries=1



# EVENT HANDLER LOGGING OPTION
# If you don't want host and service event handlers to be logged, set
# this value to 0.  If event handlers should be logged, set the value
# to 1.

log_event_handlers=1



# INITIAL STATES LOGGING OPTION
# If you want Nagios to log all initial host and service states to
# the main log file (the first time the service or host is checked)
# you can enable this option by setting this value to 1.  If you
# are not using an external application that does long term state
# statistics reporting, you do not need to enable this option.  In
# this case, set the value to 0.

log_initial_states=0



# EXTERNAL COMMANDS LOGGING OPTION
# If you don't want Nagios to log external commands, set this value
# to 0.  If external commands should be logged, set this value to 1.
# Note: This option does not include logging of passive service
# checks - see the option below for controlling whether or not
# passive checks are logged.

log_external_commands=1
```

```
# PASSIVE CHECKS LOGGING OPTION
# If you don't want Nagios to log passive host and service checks, set
# this value to 0.  If passive checks should be logged, set
# this value to 1.

log_passive_checks=1



# GLOBAL HOST AND SERVICE EVENT HANDLERS
# These options allow you to specify a host and service event handler
# command that is to be run for every host or service state change.
# The global event handler is executed immediately prior to the event
# handler that you have optionally specified in each host or
# service definition. The command argument is the short name of a
# command definition that you define in your host configuration file.
# Read the HTML docs for more information.

#global_host_event_handler=somecommand
#global_service_event_handler=somecommand



# SERVICE INTER-CHECK DELAY METHOD
# This is the method that Nagios should use when initially
# "spreading out" service checks when it starts monitoring.  The
# default is to use smart delay calculation, which will try to
# space all service checks out evenly to minimize CPU load.
# Using the dumb setting will cause all checks to be scheduled
# at the same time (with no delay between them)!  This is not a
# good thing for production, but is useful when testing the
# parallelization functionality.
#     n     = None - don't use any delay between checks
#     d     = Use a "dumb" delay of 1 second between checks
#     s     = Use "smart" inter-check delay calculation
#       x.xx    = Use an inter-check delay of x.xx seconds

service_inter_check_delay_method=s



# MAXIMUM SERVICE CHECK SPREAD
# This variable determines the timeframe (in minutes) from the
# program start time that an initial check of all services should
# be completed.  Default is 30 minutes.

max_service_check_spread=30



# SERVICE CHECK INTERLEAVE FACTOR
# This variable determines how service checks are interleaved.
# Interleaving the service checks allows for a more even
# distribution of service checks and reduced load on remote
# hosts.  Setting this value to 1 is equivalent to how versions
# of Nagios previous to 0.0.5 did service checks.  Set this
# value to s (smart) for automatic calculation of the interleave
# factor unless you have a specific reason to change it.
```

```
#       s       = Use "smart" interleave factor calculation
#       x       = Use an interleave factor of x, where x is a
#                 number greater than or equal to 1.

service_interleave_factor=s



# HOST INTER-CHECK DELAY METHOD
# This is the method that Nagios should use when initially
# "spreading out" host checks when it starts monitoring.  The
# default is to use smart delay calculation, which will try to
# space all host checks out evenly to minimize CPU load.
# Using the dumb setting will cause all checks to be scheduled
# at the same time (with no delay between them)!
#     n     = None - don't use any delay between checks
#     d     = Use a "dumb" delay of 1 second between checks
#     s     = Use "smart" inter-check delay calculation
#       x.xx    = Use an inter-check delay of x.xx seconds

host_inter_check_delay_method=s



# MAXIMUM HOST CHECK SPREAD
# This variable determines the timeframe (in minutes) from the
# program start time that an initial check of all hosts should
# be completed.  Default is 30 minutes.

max_host_check_spread=30



# MAXIMUM CONCURRENT SERVICE CHECKS
# This option allows you to specify the maximum number of
# service checks that can be run in parallel at any given time.
# Specifying a value of 1 for this variable essentially prevents
# any service checks from being parallelized.  A value of 0
# will not restrict the number of concurrent checks that are
# being executed.

max_concurrent_checks=0



# HOST AND SERVICE CHECK REAPER FREQUENCY
# This is the frequency (in seconds!) that Nagios will process
# the results of host and service checks.

check_result_reaper_frequency=10



# MAX CHECK RESULT REAPER TIME
# This is the max amount of time (in seconds) that  a single
# check result reaper event will be allowed to run before
# returning control back to Nagios so it can perform other
```

```
# duties.

max_check_result_reaper_time=30



# CHECK RESULT PATH
# This is directory where Nagios stores the results of host and
# service checks that have not yet been processed.
#
# Note: Make sure that only one instance of Nagios has access
# to this directory!

check_result_path=/var/log/nagios/spool/checkresults



# MAX CHECK RESULT FILE AGE
# This option determines the maximum age (in seconds) which check
# result files are considered to be valid.  Files older than this
# threshold will be mercilessly deleted without further processing.

max_check_result_file_age=3600



# CACHED HOST CHECK HORIZON
# This option determines the maximum amount of time (in seconds)
# that the state of a previous host check is considered current.
# Cached host states (from host checks that were performed more
# recently that the timeframe specified by this value) can immensely
# improve performance in regards to the host check logic.
# Too high of a value for this option may result in inaccurate host
# states being used by Nagios, while a lower value may result in a
# performance hit for host checks.  Use a value of 0 to disable host
# check caching.

cached_host_check_horizon=15



# CACHED SERVICE CHECK HORIZON
# This option determines the maximum amount of time (in seconds)
# that the state of a previous service check is considered current.
# Cached service states (from service checks that were performed more
# recently that the timeframe specified by this value) can immensely
# improve performance in regards to predictive dependency checks.
# Use a value of 0 to disable service check caching.

cached_service_check_horizon=15



# ENABLE PREDICTIVE HOST DEPENDENCY CHECKS
# This option determines whether or not Nagios will attempt to execute
```

```
# checks of hosts when it predicts that future dependency logic test
# may be needed.  These predictive checks can help ensure that your
# host dependency logic works well.
# Values:
#  0 = Disable predictive checks
#  1 = Enable predictive checks (default)

enable_predictive_host_dependency_checks=1



# ENABLE PREDICTIVE SERVICE DEPENDENCY CHECKS
# This option determines whether or not Nagios will attempt to execute
# checks of service when it predicts that future dependency logic test
# may be needed.  These predictive checks can help ensure that your
# service dependency logic works well.
# Values:
#  0 = Disable predictive checks
#  1 = Enable predictive checks (default)

enable_predictive_service_dependency_checks=1



# SOFT STATE DEPENDENCIES
# This option determines whether or not Nagios will use soft state
# information when checking host and service dependencies. Normally
# Nagios will only use the latest hard host or service state when
# checking dependencies. If you want it to use the latest state
(regardless
# of whether its a soft or hard state type), enable this option.
# Values:
#  0 = Don't use soft state dependencies (default)
#  1 = Use soft state dependencies

soft_state_dependencies=0



# TIME CHANGE ADJUSTMENT THRESHOLDS
# These options determine when Nagios will react to detected changes
# in system time (either forward or backwards).

#time_change_threshold=900



# AUTO-RESCHEDULING OPTION
# This option determines whether or not Nagios will attempt to
# automatically reschedule active host and service checks to
# "smooth" them out over time.  This can help balance the load on
# the monitoring server.
# WARNING: THIS IS AN EXPERIMENTAL FEATURE - IT CAN DEGRADE
# PERFORMANCE, RATHER THAN INCREASE IT, IF USED IMPROPERLY

auto_reschedule_checks=0
```

```
# AUTO-RESCHEDULING INTERVAL
# This option determines how often (in seconds) Nagios will
# attempt to automatically reschedule checks.  This option only
# has an effect if the auto_reschedule_checks option is enabled.
# Default is 30 seconds.
# WARNING: THIS IS AN EXPERIMENTAL FEATURE - IT CAN DEGRADE
# PERFORMANCE, RATHER THAN INCREASE IT, IF USED IMPROPERLY

auto_rescheduling_interval=30



# AUTO-RESCHEDULING WINDOW
# This option determines the "window" of time (in seconds) that
# Nagios will look at when automatically rescheduling checks.
# Only host and service checks that occur in the next X seconds
# (determined by this variable) will be rescheduled. This option
# only has an effect if the auto_reschedule_checks option is
# enabled.  Default is 180 seconds (3 minutes).
# WARNING: THIS IS AN EXPERIMENTAL FEATURE - IT CAN DEGRADE
# PERFORMANCE, RATHER THAN INCREASE IT, IF USED IMPROPERLY

auto_rescheduling_window=180



# SLEEP TIME
# This is the number of seconds to sleep between checking for system
# events and service checks that need to be run.

sleep_time=0.25



# TIMEOUT VALUES
# These options control how much time Nagios will allow various
# types of commands to execute before killing them off.  Options
# are available for controlling maximum time allotted for
# service checks, host checks, event handlers, notifications, the
# ocsp command, and performance data commands.  All values are in
# seconds.

service_check_timeout=60
host_check_timeout=30
event_handler_timeout=30
notification_timeout=30
ocsp_timeout=5
perfdata_timeout=5



# RETAIN STATE INFORMATION
# This setting determines whether or not Nagios will save state
# information for services and hosts before it shuts down.  Upon
# startup Nagios will reload all saved service and host state
# information before starting to monitor.  This is useful for
# maintaining long-term data on state statistics, etc, but will
```

```
# slow Nagios down a bit when it (re)starts.  Since its only
# a one-time penalty, I think its well worth the additional
# startup delay.

retain_state_information=1



# STATE RETENTION FILE
# This is the file that Nagios should use to store host and
# service state information before it shuts down.  The state
# information in this file is also read immediately prior to
# starting to monitor the network when Nagios is restarted.
# This file is used only if the retain_state_information
# variable is set to 1.

state_retention_file=/var/log/nagios/retention.dat



# RETENTION DATA UPDATE INTERVAL
# This setting determines how often (in minutes) that Nagios
# will automatically save retention data during normal operation.
# If you set this value to 0, Nagios will not save retention
# data at regular interval, but it will still save retention
# data before shutting down or restarting.  If you have disabled
# state retention, this option has no effect.

retention_update_interval=60



# USE RETAINED PROGRAM STATE
# This setting determines whether or not Nagios will set
# program status variables based on the values saved in the
# retention file.  If you want to use retained program status
# information, set this value to 1.  If not, set this value
# to 0.

use_retained_program_state=1



# USE RETAINED SCHEDULING INFO
# This setting determines whether or not Nagios will retain
# the scheduling info (next check time) for hosts and services
# based on the values saved in the retention file.  If you
# If you want to use retained scheduling info, set this
# value to 1.  If not, set this value to 0.

use_retained_scheduling_info=1



# RETAINED ATTRIBUTE MASKS (ADVANCED FEATURE)
# The following variables are used to specify specific host and
# service attributes that should *not* be retained by Nagios during
# program restarts.
```

```
#
# The values of the masks are bitwise ANDs of values specified
# by the "MODATTR_" definitions found in include/common.h.
# For example, if you do not want the current enabled/disabled state
# of flap detection and event handlers for hosts to be retained, you
# would use a value of 24 for the host attribute mask...
# MODATTR_EVENT_HANDLER_ENABLED (8) + MODATTR_FLAP_DETECTION_ENABLED (16)
= 24

# This mask determines what host attributes are not retained
retained_host_attribute_mask=0

# This mask determines what service attributes are not retained
retained_service_attribute_mask=0

# These two masks determine what process attributes are not retained.
# There are two masks, because some process attributes have host and
service
# options.  For example, you can disable active host checks, but leave
active
# service checks enabled.
retained_process_host_attribute_mask=0
retained_process_service_attribute_mask=0

# These two masks determine what contact attributes are not retained.
# There are two masks, because some contact attributes have host and
# service options.  For example, you can disable host notifications for
# a contact, but leave service notifications enabled for them.
retained_contact_host_attribute_mask=0
retained_contact_service_attribute_mask=0



# INTERVAL LENGTH
# This is the seconds per unit interval as used in the
# host/contact/service configuration files.  Setting this to 60 means
# that each interval is one minute long (60 seconds).  Other settings
# have not been tested much, so your mileage is likely to vary...

interval_length=60



# CHECK FOR UPDATES
# This option determines whether Nagios will automatically check to
# see if new updates (releases) are available.  It is recommend that you
# enable this option to ensure that you stay on top of the latest
critical
# patches to Nagios.  Nagios is critical to you - make sure you keep it
in
# good shape.  Nagios will check once a day for new updates. Data
collected
# by Nagios Enterprises from the update check is processed in accordance
# with our privacy policy - see http://api.nagios.org for details.

check_for_updates=1
```

```
# BARE UPDATE CHECK
# This option deterines what data Nagios will send to api.nagios.org when
# it checks for updates.  By default, Nagios will send information on the
# current version of Nagios you have installed, as well as an indicator
as
# to whether this was a new installation or not.  Nagios Enterprises uses
# this data to determine the number of users running specific version of
# Nagios.  Enable this option if you do not want this information to be
sent.

bare_update_check=0



# AGGRESSIVE HOST CHECKING OPTION
# If you don't want to turn on aggressive host checking features, set
# this value to 0 (the default).  Otherwise set this value to 1 to
# enable the aggressive check option.  Read the docs for more info
# on what aggressive host check is or check out the source code in
# base/checks.c

use_aggressive_host_checking=0



# SERVICE CHECK EXECUTION OPTION
# This determines whether or not Nagios will actively execute
# service checks when it initially starts.  If this option is
# disabled, checks are not actively made, but Nagios can still
# receive and process passive check results that come in.  Unless
# you're implementing redundant hosts or have a special need for
# disabling the execution of service checks, leave this enabled!
# Values: 1 = enable checks, 0 = disable checks

execute_service_checks=1



# PASSIVE SERVICE CHECK ACCEPTANCE OPTION
# This determines whether or not Nagios will accept passive
# service checks results when it initially (re)starts.
# Values: 1 = accept passive checks, 0 = reject passive checks

accept_passive_service_checks=1



# HOST CHECK EXECUTION OPTION
# This determines whether or not Nagios will actively execute
# host checks when it initially starts.  If this option is
# disabled, checks are not actively made, but Nagios can still
# receive and process passive check results that come in.  Unless
# you're implementing redundant hosts or have a special need for
# disabling the execution of host checks, leave this enabled!
# Values: 1 = enable checks, 0 = disable checks

execute_host_checks=1
```

```
# PASSIVE HOST CHECK ACCEPTANCE OPTION
# This determines whether or not Nagios will accept passive
# host checks results when it initially (re)starts.
# Values: 1 = accept passive checks, 0 = reject passive checks

accept_passive_host_checks=1



# NOTIFICATIONS OPTION
# This determines whether or not Nagios will sent out any host or
# service notifications when it is initially (re)started.
# Values: 1 = enable notifications, 0 = disable notifications

enable_notifications=1



# EVENT HANDLER USE OPTION
# This determines whether or not Nagios will run any host or
# service event handlers when it is initially (re)started.  Unless
# you're implementing redundant hosts, leave this option enabled.
# Values: 1 = enable event handlers, 0 = disable event handlers

enable_event_handlers=1



# PROCESS PERFORMANCE DATA OPTION
# This determines whether or not Nagios will process performance
# data returned from service and host checks.  If this option is
# enabled, host performance data will be processed using the
# host_perfdata_command (defined below) and service performance
# data will be processed using the service_perfdata_command (also
# defined below).  Read the HTML docs for more information on
# performance data.
# Values: 1 = process performance data, 0 = do not process performance
data

process_performance_data=0



# HOST AND SERVICE PERFORMANCE DATA PROCESSING COMMANDS
# These commands are run after every host and service check is
# performed.  These commands are executed only if the
# enable_performance_data option (above) is set to 1.  The command
# argument is the short name of a command definition that you
# define in your host configuration file.  Read the HTML docs for
# more information on performance data.

#host_perfdata_command=process-host-perfdata
#service_perfdata_command=process-service-perfdata
```

```
# HOST AND SERVICE PERFORMANCE DATA FILES
# These files are used to store host and service performance data.
# Performance data is only written to these files if the
# enable_performance_data option (above) is set to 1.

#host_perfdata_file=/tmp/host-perfdata
#service_perfdata_file=/tmp/service-perfdata



# HOST AND SERVICE PERFORMANCE DATA FILE TEMPLATES
# These options determine what data is written (and how) to the
# performance data files.  The templates may contain macros, special
# characters (\t for tab, \r for carriage return, \n for newline)
# and plain text.  A newline is automatically added after each write
# to the performance data file.  Some examples of what you can do are
# shown below.

#host_perfdata_file_template=[HOSTPERFDATA]\t$TIMET$\t$HOSTNAME$\t$HOSTEX
ECUTIONTIME$\t$HOSTOUTPUT$\t$HOSTPERFDATA$
#service_perfdata_file_template=[SERVICEPERFDATA]\t$TIMET$\t$HOSTNAME$\t$
SERVICEDESC$\t$SERVICEEXECUTIONTIME$\t$SERVICELATENCY$\t$SERVICEOUTPUT$\t
$SERVICEPERFDATA$



# HOST AND SERVICE PERFORMANCE DATA FILE MODES
# This option determines whether or not the host and service
# performance data files are opened in write ("w") or append ("a")
# mode. If you want to use named pipes, you should use the special
# pipe ("p") mode which avoid blocking at startup, otherwise you will
# likely want the defult append ("a") mode.

#host_perfdata_file_mode=a
#service_perfdata_file_mode=a



# HOST AND SERVICE PERFORMANCE DATA FILE PROCESSING INTERVAL
# These options determine how often (in seconds) the host and service
# performance data files are processed using the commands defined
# below.  A value of 0 indicates the files should not be periodically
# processed.

#host_perfdata_file_processing_interval=0
#service_perfdata_file_processing_interval=0



# HOST AND SERVICE PERFORMANCE DATA FILE PROCESSING COMMANDS
# These commands are used to periodically process the host and
# service performance data files.  The interval at which the
# processing occurs is determined by the options above.

#host_perfdata_file_processing_command=process-host-perfdata-file
#service_perfdata_file_processing_command=process-service-perfdata-file
```

```
# HOST AND SERVICE PERFORMANCE DATA PROCESS EMPTY RESULTS
# THese options determine wether the core will process empty perfdata
# results or not. This is needed for distributed monitoring, and
intentionally
# turned on by default.
# If you don't require empty perfdata - saving some cpu cycles
# on unwanted macro calculation - you can turn that off. Be careful!
# Values: 1 = enable, 0 = disable

#host_perfdata_process_empty_results=1
#service_perfdata_process_empty_results=1


# OBSESS OVER SERVICE CHECKS OPTION
# This determines whether or not Nagios will obsess over service
# checks and run the ocsp_command defined below.  Unless you're
# planning on implementing distributed monitoring, do not enable
# this option.  Read the HTML docs for more information on
# implementing distributed monitoring.
# Values: 1 = obsess over services, 0 = do not obsess (default)

obsess_over_services=0



# OBSESSIVE COMPULSIVE SERVICE PROCESSOR COMMAND
# This is the command that is run for every service check that is
# processed by Nagios.  This command is executed only if the
# obsess_over_services option (above) is set to 1.  The command
# argument is the short name of a command definition that you
# define in your host configuration file. Read the HTML docs for
# more information on implementing distributed monitoring.

#ocsp_command=somecommand



# OBSESS OVER HOST CHECKS OPTION
# This determines whether or not Nagios will obsess over host
# checks and run the ochp_command defined below.  Unless you're
# planning on implementing distributed monitoring, do not enable
# this option.  Read the HTML docs for more information on
# implementing distributed monitoring.
# Values: 1 = obsess over hosts, 0 = do not obsess (default)

obsess_over_hosts=0



# OBSESSIVE COMPULSIVE HOST PROCESSOR COMMAND
# This is the command that is run for every host check that is
# processed by Nagios.  This command is executed only if the
# obsess_over_hosts option (above) is set to 1.  The command
# argument is the short name of a command definition that you
# define in your host configuration file. Read the HTML docs for
# more information on implementing distributed monitoring.
```

```
#ochp_command=somecommand



# TRANSLATE PASSIVE HOST CHECKS OPTION
# This determines whether or not Nagios will translate
# DOWN/UNREACHABLE passive host check results into their proper
# state for this instance of Nagios.  This option is useful
# if you have distributed or failover monitoring setup.  In
# these cases your other Nagios servers probably have a different
# "view" of the network, with regards to the parent/child relationship
# of hosts.  If a distributed monitoring server thinks a host
# is DOWN, it may actually be UNREACHABLE from the point of
# this Nagios instance.  Enabling this option will tell Nagios
# to translate any DOWN or UNREACHABLE host states it receives
# passively into the correct state from the view of this server.
# Values: 1 = perform translation, 0 = do not translate (default)

translate_passive_host_checks=0



# PASSIVE HOST CHECKS ARE SOFT OPTION
# This determines whether or not Nagios will treat passive host
# checks as being HARD or SOFT.  By default, a passive host check
# result will put a host into a HARD state type.  This can be changed
# by enabling this option.
# Values: 0 = passive checks are HARD, 1 = passive checks are SOFT

passive_host_checks_are_soft=0



# ORPHANED HOST/SERVICE CHECK OPTIONS
# These options determine whether or not Nagios will periodically
# check for orphaned host service checks.  Since service checks are
# not rescheduled until the results of their previous execution
# instance are processed, there exists a possibility that some
# checks may never get rescheduled.  A similar situation exists for
# host checks, although the exact scheduling details differ a bit
# from service checks.  Orphaned checks seem to be a rare
# problem and should not happen under normal circumstances.
# If you have problems with service checks never getting
# rescheduled, make sure you have orphaned service checks enabled.
# Values: 1 = enable checks, 0 = disable checks

check_for_orphaned_services=1
check_for_orphaned_hosts=1



# SERVICE FRESHNESS CHECK OPTION
# This option determines whether or not Nagios will periodically
# check the "freshness" of service results.  Enabling this option
# is useful for ensuring passive checks are received in a timely
# manner.
# Values: 1 = enabled freshness checking, 0 = disable freshness checking
```

```
check_service_freshness=1



# SERVICE FRESHNESS CHECK INTERVAL
# This setting determines how often (in seconds) Nagios will
# check the "freshness" of service check results.  If you have
# disabled service freshness checking, this option has no effect.

service_freshness_check_interval=60



# SERVICE CHECK TIMEOUT STATE
# This setting determines the state Nagios will report when a
# service check times out - that is does not respond within
# service_check_timeout seconds.  This can be useful if a
# machine is running at too high a load and you do not want
# to consider a failed service check to be critical (the default).
# Valid settings are:
# c - Critical (default)
# u - Unknown
# w - Warning
# o - OK

service_check_timeout_state=c



# HOST FRESHNESS CHECK OPTION
# This option determines whether or not Nagios will periodically
# check the "freshness" of host results.  Enabling this option
# is useful for ensuring passive checks are received in a timely
# manner.
# Values: 1 = enabled freshness checking, 0 = disable freshness checking

check_host_freshness=0



# HOST FRESHNESS CHECK INTERVAL
# This setting determines how often (in seconds) Nagios will
# check the "freshness" of host check results.  If you have
# disabled host freshness checking, this option has no effect.

host_freshness_check_interval=60




# ADDITIONAL FRESHNESS THRESHOLD LATENCY
# This setting determines the number of seconds that Nagios
# will add to any host and service freshness thresholds that
# it calculates (those not explicitly specified by the user).

additional_freshness_latency=15
```

```
# FLAP DETECTION OPTION
# This option determines whether or not Nagios will try
# and detect hosts and services that are "flapping".
# Flapping occurs when a host or service changes between
# states too frequently.  When Nagios detects that a
# host or service is flapping, it will temporarily suppress
# notifications for that host/service until it stops
# flapping.  Flap detection is very experimental, so read
# the HTML documentation before enabling this feature!
# Values: 1 = enable flap detection
#         0 = disable flap detection (default)

enable_flap_detection=1



# FLAP DETECTION THRESHOLDS FOR HOSTS AND SERVICES
# Read the HTML documentation on flap detection for
# an explanation of what this option does.  This option
# has no effect if flap detection is disabled.

low_service_flap_threshold=5.0
high_service_flap_threshold=20.0
low_host_flap_threshold=5.0
high_host_flap_threshold=20.0



# DATE FORMAT OPTION
# This option determines how short dates are displayed. Valid options
# include:
#     us          (MM-DD-YYYY HH:MM:SS)
#     euro        (DD-MM-YYYY HH:MM:SS)
#     iso8601           (YYYY-MM-DD HH:MM:SS)
#     strict-iso8601    (YYYY-MM-DDTHH:MM:SS)
#

date_format=us




# TIMEZONE OFFSET
# This option is used to override the default timezone that this
# instance of Nagios runs in.  If not specified, Nagios will use
# the system configured timezone.
#
# NOTE: In order to display the correct timezone in the CGIs, you
# will also need to alter the Apache directives for the CGI path
# to include your timezone.  Example:
#
#   <Directory "/usr/local/nagios/sbin/">
#      SetEnv TZ "Australia/Brisbane"
#      ...
#   </Directory>
```

```
#use_timezone=US/Mountain
#use_timezone=Australia/Brisbane



# P1.PL FILE LOCATION
# This value determines where the p1.pl perl script (used by the
# embedded Perl interpreter) is located.  If you didn't compile
# Nagios with embedded Perl support, this option has no effect.

p1_file=/usr/sbin/p1.pl



# EMBEDDED PERL INTERPRETER OPTION
# This option determines whether or not the embedded Perl interpreter
# will be enabled during runtime.  This option has no effect if Nagios
# has not been compiled with support for embedded Perl.
# Values: 0 = disable interpreter, 1 = enable interpreter

enable_embedded_perl=1



# EMBEDDED PERL USAGE OPTION
# This option determines whether or not Nagios will process Perl plugins
# and scripts with the embedded Perl interpreter if the plugins/scripts
# do not explicitly indicate whether or not it is okay to do so. Read
# the HTML documentation on the embedded Perl interpreter for more
# information on how this option works.

use_embedded_perl_implicitly=1



# ILLEGAL OBJECT NAME CHARACTERS
# This option allows you to specify illegal characters that cannot
# be used in host names, service descriptions, or names of other
# object types.

illegal_object_name_chars=`~!$%^&*|'"<>?,()=



# ILLEGAL MACRO OUTPUT CHARACTERS
# This option allows you to specify illegal characters that are
# stripped from macros before being used in notifications, event
# handlers, etc.  This DOES NOT affect macros used in service or
# host check commands.
# The following macros are stripped of the characters you specify:
#     $HOSTOUTPUT$
#     $HOSTPERFDATA$
#     $HOSTACKAUTHOR$
#     $HOSTACKCOMMENT$
#     $SERVICEOUTPUT$
#     $SERVICEPERFDATA$
```

```
#     $SERVICEACKAUTHOR$
#     $SERVICEACKCOMMENT$

illegal_macro_output_chars=`~$&|'"<>



# REGULAR EXPRESSION MATCHING
# This option controls whether or not regular expression matching
# takes place in the object config files.  Regular expression
# matching is used to match host, hostgroup, service, and service
# group names/descriptions in some fields of various object types.
# Values: 1 = enable regexp matching, 0 = disable regexp matching

use_regexp_matching=0



# "TRUE" REGULAR EXPRESSION MATCHING
# This option controls whether or not "true" regular expression
# matching takes place in the object config files.  This option
# only has an effect if regular expression matching is enabled
# (see above).  If this option is DISABLED, regular expression
# matching only occurs if a string contains wildcard characters
# (* and ?).  If the option is ENABLED, regexp matching occurs
# all the time (which can be annoying).
# Values: 1 = enable true matching, 0 = disable true matching

use_true_regexp_matching=0



# ADMINISTRATOR EMAIL/PAGER ADDRESSES
# The email and pager address of a global administrator (likely you).
# Nagios never uses these values itself, but you can access them by
# using the $ADMINEMAIL$ and $ADMINPAGER$ macros in your notification
# commands.

admin_email=nagios@localhost
admin_pager=pagenagios@localhost



# DAEMON CORE DUMP OPTION
# This option determines whether or not Nagios is allowed to create
# a core dump when it runs as a daemon.  Note that it is generally
# considered bad form to allow this, but it may be useful for
# debugging purposes.  Enabling this option doesn't guarantee that
# a core file will be produced, but that's just life...
# Values: 1 - Allow core dumps
#         0 - Do not allow core dumps (default)

daemon_dumps_core=0



# LARGE INSTALLATION TWEAKS OPTION
# This option determines whether or not Nagios will take some shortcuts
```

```
# which can save on memory and CPU usage in large Nagios installations.
# Read the documentation for more information on the benefits/tradeoffs
# of enabling this option.
# Values: 1 - Enabled tweaks
#         0 - Disable tweaks (default)

use_large_installation_tweaks=0



# ENABLE ENVIRONMENT MACROS
# This option determines whether or not Nagios will make all standard
# macros available as environment variables when host/service checks
# and system commands (event handlers, notifications, etc.) are
# executed.  Enabling this option can cause performance issues in
# large installations, as it will consume a bit more memory and (more
# importantly) consume more CPU.
# Values: 1 - Enable environment variable macros (default)
#         0 - Disable environment variable macros

enable_environment_macros=1



# CHILD PROCESS MEMORY OPTION
# This option determines whether or not Nagios will free memory in
# child processes (processed used to execute system commands and host/
# service checks).  If you specify a value here, it will override
# program defaults.
# Value: 1 - Free memory in child processes
#        0 - Do not free memory in child processes

#free_child_process_memory=1



# CHILD PROCESS FORKING BEHAVIOR
# This option determines how Nagios will fork child processes
# (used to execute system commands and host/service checks).  Normally
# child processes are fork()ed twice, which provides a very high level
# of isolation from problems.  Fork()ing once is probably enough and will
# save a great deal on CPU usage (in large installs), so you might
# want to consider using this.  If you specify a value here, it will
# program defaults.
# Value: 1 - Child processes fork() twice
#        0 - Child processes fork() just once

#child_processes_fork_twice=1



# DEBUG LEVEL
# This option determines how much (if any) debugging information will
# be written to the debug file.  OR values together to log multiple
# types of information.
# Values:
#          -1 = Everything
#          0 = Nothing
```

```
#        1 = Functions
#          2 = Configuration
#          4 = Process information
#        8 = Scheduled events
#          16 = Host/service checks
#          32 = Notifications
#          64 = Event broker
#          128 = External commands
#          256 = Commands
#          512 = Scheduled downtime
#          1024 = Comments
#          2048 = Macros

debug_level=0



# DEBUG VERBOSITY
# This option determines how verbose the debug log out will be.
# Values: 0 = Brief output
#         1 = More detailed
#         2 = Very detailed

debug_verbosity=1



# DEBUG FILE
# This option determines where Nagios should write debugging information.

debug_file=/var/log/nagios/nagios.debug



# MAX DEBUG FILE SIZE
# This option determines the maximum size (in bytes) of the debug file.
If
# the file grows larger than this size, it will be renamed with a .old
# extension.  If a file already exists with a .old extension it will
# automatically be deleted.  This helps ensure your disk space usage
doesn't
# get out of control when debugging Nagios.

max_debug_file_size=1000000
```