

Nagios Object Definitions

Host Definition.....	1
Host Group Definition.....	7
Service Definition.....	8
Service Group Definition.....	12
Contact Definition.....	13
Contact Group Definition.....	15
Time Period Definition.....	16
Command Definition.....	17
Service Dependency Definition.....	17
Service Escalation Definition.....	19
Host Dependency Definition.....	20
Host Escalation Definition.....	21
Extended Host Information Definition.....	23
Extended Service Information Definition.....	24

Host Definition

Definition Format

Note: Directives followed by a * are required.

```
define host{
    host_name*                host_name
    alias*                    alias
    display_name                display_name
    address*                  address
    parents                      host_names
    hostgroups                   hostgroup_names
    check_command                command_name
    initial_state                [o,d,u]
    max_check_attempts*        #
    check_interval               #
    retry_interval               #
    active_checks_enabled        [0/1]
    passive_checks_enabled       [0/1]
    check_period*              timeperiod_name
    obsess_over_host              [0/1]
    check_freshness               [0/1]
    freshness_threshold           #
    event_handler                 command_name
    event_handler_enabled         [0/1]
    low_flap_threshold            #
    high_flap_threshold           #
    flap_detection_enabled        [0/1]
    flap_detection_options        [o,d,u]
    process_perf_data             [0/1]
    retain_status_information      [0/1]
}
```

```

retain_nonstatus_information      [0/1]
contacts*                       contacts
contact_groups*                 contact_groups
notification_interval*          #
first_notification_delay          #
notification_period*           timeperiod_name
notification_options              [d,u,r,f,s]
notifications_enabled             [0/1]
stalking_options                  [o,d,u]
notes                             note_string
notes_url                         url
action_url                        url
icon_image                        image_file
icon_image_alt                    alt_string
vrmml_image                       image_file
statusmap_image                   image_file
2d_coords                         x_coord,y_coord
3d_coords                          x_coord,y_coord,z_coord
}

```

Example Definition

```

define host{
    host_name          bogus-router
    alias              Bogus Router #1
    address            192.168.1.254
    parents            server-backbone
    check_command      check-host-alive
    check_interval     5
    retry_interval     1
    max_check_attempts 5
    check_period       24x7
    process_perf_data  0
    retain_nonstatus_information 0
    contact_groups     router-admins
    notification_interval 30
    notification_period 24x7
    notification_options d,u,r
}

```

Directive Descriptions

host_name	This directive is used to define a short name used to identify the host. It is used in host group and service definitions to reference this particular host. Hosts can have multiple services (which are monitored) associated with them. When used properly, the \$HOSTNAME\$ macro will contain this short name.
alias	This directive is used to define a longer name or description used to identify the host. It is provided in order to allow you to more easily identify a particular host. When used properly, the \$HOSTALIAS\$ macro will contain this alias/description.
address	This directive is used to define the address of the host. Normally, this is an IP address, although it could really be anything you want (so long as it can be used to check the status of the host). You can use a FQDN to identify the host instead of an IP address, but if DNS services are not 3333cc□le this could cause problems. When used properly, the \$HOSTADDRESS\$ macro will contain this address. Note: If you do not specify an

	address directive in a host definition, the name of the host will be used as its address. A word of caution about doing this, however - if DNS fails, most of your service checks will fail because the plugins will be unable to resolve the host name.
display_name	This directive is used to define an alternate name that should be displayed in the web interface for this host. If not specified, this defaults to the value you specify for the <i>host_name</i> directive. Note: The current CGIs do not use this option, although future versions of the web interface will.
parents	This directive is used to define a comma-delimited list of short names of the "parent" hosts for this particular host. Parent hosts are typically routers, switches, firewalls, etc. that lie between the monitoring host and a remote hosts. A router, switch, etc. which is closest to the remote host is considered to be that host's "parent". Read the "Determining Status and Reachability of Network Hosts" document located here for more information. If this host is on the same network segment as the host doing the monitoring (without any intermediate routers, etc.) the host is considered to be on the local network and will not have a parent host. Leave this value blank if the host does not have a parent host (i.e. it is on the same segment as the Nagios host). The order in which you specify parent hosts has no effect on how things are monitored.
hostgroups	This directive is used to identify the <i>short name(s)</i> of the hostgroup(s) that the host belongs to. Multiple hostgroups should be separated by commas. This directive may be used as an alternative to (or in addition to) using the <i>members</i> directive in hostgroup definitions.
check_command	This directive is used to specify the <i>short name</i> of the command that should be used to check if the host is up or down. Typically, this command would try and ping the host to see if it is "alive". The command must return a status of OK (0) or Nagios will assume the host is down. If you leave this argument blank, the host will <i>not</i> be actively checked. Thus, Nagios will likely always assume the host is up (it may show up as being in a "PENDING" state in the web interface). This is useful if you are monitoring printers or other devices that are frequently turned off. The maximum amount of time that the notification command can run is controlled by the host_check_timeout option.
initial_state	By default Nagios will assume that all hosts are in UP states when it starts. You can override the initial state for a host by using this directive. Valid options are: o = UP, d = DOWN, and u = UNREACHABLE.
max_check_attempts	This directive is used to define the number of times that Nagios will retry the host check command if it returns any state other than an OK state. Setting this value to 1 will cause Nagios to generate an alert without retrying the host check. Note: If you do not want to check the status of the host, you must still set this to a minimum value of 1. To bypass the host check, just leave the <i>check_command</i> option blank.
check_interval	This directive is used to define the number of "time units" between regularly scheduled checks of the host. Unless you've changed the interval_length directive from the default value of 60, this number will mean minutes. More information on this value can be found in the check scheduling documentation.
retry_interval	This directive is used to define the number of "time units" to wait before scheduling a re-check of the hosts. Hosts are rescheduled at the retry interval when they have changed to a non-UP state. Once the host has been retried max_check_attempts times without a change in its status, it will revert to being scheduled at its "normal" rate as defined by the check_interval value. Unless you've changed the interval_length directive from the default value of 60, this number will mean minutes. More information on this value can be found in the check scheduling documentation.
active_checks_enabled	This directive is used to determine whether or not active checks (either regularly scheduled or on-demand) of this host are enabled. Values: 0 = disable active host checks, 1 = enable active host checks (default).
passive_checks_enabled	This directive is used to determine whether or not passive checks are enabled for this host. Values: 0 = disable passive host checks, 1 = enable passive host checks (default).
check_period	This directive is used to specify the short name of the time period during which active checks of this host can be made.

obsess_over_host	This directive determines whether or not checks for the host will be "obsessed" over using the ochp_command .
check_freshness	This directive is used to determine whether or not freshness checks are enabled for this host. Values: 0 = disable freshness checks, 1 = enable freshness checks (default).
freshness_threshold	This directive is used to specify the freshness threshold (in seconds) for this host. If you set this directive to a value of 0, Nagios will determine a freshness threshold to use automatically.
event_handler	This directive is used to specify the <i>short name</i> of the command that should be run whenever a change in the state of the host is detected (i.e. whenever it goes down or recovers). Read the documentation on event handlers for a more detailed explanation of how to write scripts for handling events. The maximum amount of time that the event handler command can run is controlled by the event_handler_timeout option.
event_handler_enabled	This directive is used to determine whether or not the event handler for this host is enabled. Values: 0 = disable host event handler, 1 = enable host event handler.
low_flap_threshold	This directive is used to specify the low state change threshold used in flap detection for this host. More information on flap detection can be found here . If you set this directive to a value of 0, the program-wide value specified by the low_host_flap_threshold directive will be used.
high_flap_threshold	This directive is used to specify the high state change threshold used in flap detection for this host. More information on flap detection can be found here . If you set this directive to a value of 0, the program-wide value specified by the high_host_flap_threshold directive will be used.
flap_detection_enabled	This directive is used to determine whether or not flap detection is enabled for this host. More information on flap detection can be found here . Values: 0 = disable host flap detection, 1 = enable host flap detection.
flap_detection_options	This directive is used to determine what host states the flap detection logic will use for this host. Valid options are a combination of one or more of the following: o = UP states, d = DOWN states, u = UNREACHABLE states.
process_perf_data	This directive is used to determine whether or not the processing of performance data is enabled for this host. Values: 0 = disable performance data processing, 1 = enable performance data processing.
retain_status_information	This directive is used to determine whether or not status-related information about the host is retained across program restarts. This is only useful if you have enabled state retention using the retain_state_information directive. Value: 0 = disable status information retention, 1 = enable status information retention.
retain_nonstatus_information	This directive is used to determine whether or not non-status information about the host is retained across program restarts. This is only useful if you have enabled state retention using the retain_state_information directive. Value: 0 = disable non-status information retention, 1 = enable non-status information retention.
contacts	This is a list of the <i>short names</i> of the contacts that should be notified whenever there are problems (or recoveries) with this host. Multiple contacts should be separated by commas. Useful if you want notifications to go to just a few people and don't want to configure contact groups . You must specify at least one contact or contact group in each host definition.
contact_groups	This is a list of the <i>short names</i> of the contact groups that should be notified whenever there are problems (or recoveries) with this host. Multiple contact groups should be separated by commas. You must specify at least one contact or contact group in each host definition.
notification_interval	This directive is used to define the number of "time units" to wait before re-notifying a contact that this service is <i>still</i> down or unreachable. Unless you've changed the interval_length directive from the default value of 60, this number will mean minutes. If you set this value to 0, Nagios will <i>not</i> re-notify contacts about problems for this host - only one problem notification will be sent out.
first_notification_delay	This directive is used to define the number of "time units" to wait before sending out

	the first problem notification when this host enters a non-UP state. Unless you've changed the interval_length directive from the default value of 60, this number will mean minutes. If you set this value to 0, Nagios will start sending out notifications immediately.
notification_period	This directive is used to specify the short name of the time period during which notifications of events for this host can be sent out to contacts. If a host goes down, becomes unreachable, or recovers during a time which is not covered by the time period, no notifications will be sent out.
notification_options	This directive is used to determine when notifications for the host should be sent out. Valid options are a combination of one or more of the following: d = send notifications on a DOWN state, u = send notifications on an UNREACHABLE state, r = send notifications on recoveries (OK state), f = send notifications when the host starts and stops flapping , and s = send notifications when scheduled downtime starts and ends. If you specify n (none) as an option, no host notifications will be sent out. If you do not specify any notification options, Nagios will assume that you want notifications to be sent out for all possible states. Example: If you specify d,r in this field, notifications will only be sent out when the host goes DOWN and when it recovers from a DOWN state.
notifications_enabled	This directive is used to determine whether or not notifications for this host are enabled. Values: 0 = disable host notifications, 1 = enable host notifications.
stalking_options	This directive determines which host states "stalking" is enabled for. Valid options are a combination of one or more of the following: o = stalk on UP states, d = stalk on DOWN states, and u = stalk on UNREACHABLE states. More information on state stalking can be found here .
notes	This directive is used to define an optional string of notes pertaining to the host. If you specify a note here, you will see the it in the extended information CGI (when you are viewing information about the specified host).
notes_url	This variable is used to define an optional URL that can be used to provide more information about the host. If you specify an URL, you will see a red folder icon in the CGIs (when you are viewing host information) that links to the URL you specify here. Any valid URL can be used. If you plan on using relative paths, the base path will be the same as what is used to access the CGIs (i.e. <code>/cgi-bin/nagios/</code>). This can be very useful if you want to make detailed information on the host, emergency contact methods, etc. available to other support staff.
action_url	This directive is used to define an optional URL that can be used to provide more actions to be performed on the host. If you specify an URL, you will see a red "splat" icon in the CGIs (when you are viewing host information) that links to the URL you specify here. Any valid URL can be used. If you plan on using relative paths, the base path will be the same as what is used to access the CGIs (i.e. <code>/cgi-bin/nagios/</code>).
icon_image	This variable is used to define the name of a GIF, PNG, or JPG image that should be associated with this host. This image will be displayed in the various places in the CGIs. The image will look best if it is 40x40 pixels in size. Images for hosts are assumed to be in the logos/ subdirectory in your HTML images directory (i.e. <code>/usr/local/nagios/share/images/logos</code>).
icon_image_alt	This variable is used to define an optional string that is used in the ALT tag of the image specified by the <code><icon_image></code> argument.
vrml_image	This variable is used to define the name of a GIF, PNG, or JPG image that should be associated with this host. This image will be used as the texture map for the specified host in the statuswrl CGI. Unlike the image you use for the <code><icon_image></code> variable, this one should probably <i>not</i> have any transparency. If it does, the host object will look a bit wierd. Images for hosts are assumed to be in the logos/ subdirectory in your HTML images directory (i.e. <code>/usr/local/nagios/share/images/logos</code>).
statusmap_image	This variable is used to define the name of an image that should be associated with this host in the statusmap CGI. You can specify a JPEG, PNG, and GIF image if you want, although I would strongly suggest using a GD2 format image, as other image formats

	will result in a lot of wasted CPU time when the statusmap image is generated. GD2 images can be created from PNG images by using the pngtogd2 utility supplied with Thomas Boutell's gd library . The GD2 images should be created in <i>uncompressed</i> format in order to minimize CPU load when the statusmap CGI is generating the network map image. The image will look best if it is 40x40 pixels in size. You can leave these option blank if you are not using the statusmap CGI. Images for hosts are assumed to be in the logos/ subdirectory in your HTML images directory (i.e. <i>/usr/local/nagios/share/images/logos</i>).
2d_coords	This variable is used to define coordinates to use when drawing the host in the statusmap CGI. Coordinates should be given in positive integers, as they correspond to physical pixels in the generated image. The origin for drawing (0,0) is in the upper left hand corner of the image and extends in the positive x direction (to the right) along the top of the image and in the positive y direction (down) along the left hand side of the image. For reference, the size of the icons drawn is usually about 40x40 pixels (text takes a little extra space). The coordinates you specify here are for the upper left hand corner of the host icon that is drawn. Note: Don't worry about what the maximum x and y coordinates that you can use are. The CGI will automatically calculate the maximum dimensions of the image it creates based on the largest x and y coordinates you specify.
3d_coords	This variable is used to define coordinates to use when drawing the host in the statuswrl CGI. Coordinates can be positive or negative real numbers. The origin for drawing is (0.0,0.0,0.0). For reference, the size of the host cubes drawn is 0.5 units on each side (text takes a little more space). The coordinates you specify here are used as the center of the host cube.

Host Group Definition

Definition Format

Note: Directives followed by a * are required.

```
define hostgroup{
    hostgroup_name*           hostgroup_name
    alias*                   alias
    members                   hosts
    hostgroup_members        hostgroups
    notes                     note_string
    notes_url                 url
    action_url                url
}
```

Example Definition

```
define hostgroup{
    hostgroup_name           novell-servers
    alias                    Novell Servers
    members                  netware1, netware2, netware3, netware4
}
```

Directive Descriptions

hostgroup_name	This directive is used to define a short name used to identify the host group.
alias	This directive is used to define is a longer name or description used to identify the host group. It is provided in order to allow you to more easily identify a particular host group.
members	This is a list of the <i>short names</i> of hosts that should be included in this group. Multiple host names should be separated by commas. This directive may be used as an alternative to (or in

	addition to) the <i>hostgroups</i> directive in host definitions .
hostgroup_members	This optional directive can be used to include hosts from other "sub" host groups in this host group. Specify a comma-delimited list of short names of other host groups whose members should be included in this group.
notes	This directive is used to define an optional string of notes pertaining to the host. If you specify a note here, you will see the it in the extended information CGI (when you are viewing information about the specified host).
notes_url	This variable is used to define an optional URL that can be used to provide more information about the host group. If you specify an URL, you will see a red folder icon in the CGIs (when you are viewing hostgroup information) that links to the URL you specify here. Any valid URL can be used. If you plan on using relative paths, the base path will the the same as what is used to access the CGIs (i.e. <i>/cgi-bin/nagios/</i>). This can be very useful if you want to make detailed information on the host group, emergency contact methods, etc. available to other support staff.
action_url	This directive is used to define an optional URL that can be used to provide more actions to be performed on the host group. If you specify an URL, you will see a red "splat" icon in the CGIs (when you are viewing hostgroup information) that links to the URL you specify here. Any valid URL can be used. If you plan on using relative paths, the base path will the the same as what is used to access the CGIs (i.e. <i>/cgi-bin/nagios/</i>).

Service Definition

Definition Format

Note: Directives followed by a * are required.

```
define service{
    host_name*                host_name
    hostgroup_name            hostgroup_name
    service_description*     service_description
    display_name              display_name
    servicegroups             servicegroup_names
    is_volatile                [0/1]
    check_command*           command_name
    initial_state              [o,w,u,c]
    max_check_attempts*      #
    check_interval*          #
    retry_interval*          #
    active_checks_enabled     [0/1]
    passive_checks_enabled    [0/1]
    check_period*           timeperiod_name
    obsess_over_service        [0/1]
    check_freshness            [0/1]
    freshness_threshold        #
    event_handler              command_name
    event_handler_enabled      [0/1]
    low_flap_threshold         #
    high_flap_threshold        #
    flap_detection_enabled     [0/1]
    flap_detection_options     [o,w,c,u]
    process_perf_data          [0/1]
    retain_status_information   [0/1]
```

```

retain_nonstatus_information           [0/1]
notification_interval*              #
first_notification_delay               #
notification_period*               timeperiod_name
notification_options                   [w,u,c,r,f,s]
notifications_enabled                  [0/1]
contacts*                           contacts
contact_groups*                     contact_groups
stalking_options                       [o,w,u,c]
notes                                  note_string
notes_url                              url
action_url                             url
icon_image                             image_file
icon_image_alt                         alt_string
}

```

Example Definition

```

define service{
    host_name                linux-server
    service_description      check-disk-sda1
    check_command             check-disk!/dev/sda1
    max_check_attempts       5
    check_interval           5
    retry_interval           3
    check_period             24x7
    notification_interval    30
    notification_period      24x7
    notification_options     w,c,r
    contact_groups           linux-admins
}

```

Directive Descriptions

host_name	This directive is used to specify the <i>short name(s)</i> of the host(s) that the service "runs" on or is associated with. Multiple hosts should be separated by commas.
hostgroup_name	This directive is used to specify the <i>short name(s)</i> of the hostgroup(s) that the service "runs" on or is associated with. Multiple hostgroups should be separated by commas. The hostgroup_name may be used instead of, or in addition to, the host_name directive.
service_description	This directive is used to define the description of the service, which may contain spaces, dashes, and colons (semicolons, apostrophes, and quotation marks should be avoided). No two services associated with the same host can have the same description. Services are uniquely identified with their <i>host_name</i> and <i>service_description</i> directives.
display_name	This directive is used to define an alternate name that should be displayed in the web interface for this service. If not specified, this defaults to the value you specify for the <i>service_description</i> directive. Note: The current CGIs do not use this option, although future versions of the web interface will.
servicegroups	This directive is used to identify the <i>short name(s)</i> of the servicegroup(s) that the service belongs to. Multiple servicegroups should be separated by commas. This directive may be used as an alternative to using the <i>members</i> directive in servicegroup definitions.
is_volatile	This directive is used to denote whether the service is "volatile". Services are normally <i>not</i> volatile. More information on volatile service and how they differ from normal services can be found here . Value: 0 = service is not volatile, 1 = service is volatile.

check_command	This directive is used to specify the <i>short name</i> of the command that Nagios will run in order to check the status of the service. The maximum amount of time that the service check command can run is controlled by the service_check_timeout option.
initial_state	By default Nagios will assume that all services are in OK states when it starts. You can override the initial state for a service by using this directive. Valid options are: o = OK, w = WARNING, u = UNKNOWN, and c = CRITICAL.
max_check_attempts	This directive is used to define the number of times that Nagios will retry the service check command if it returns any state other than an OK state. Setting this value to 1 will cause Nagios to generate an alert without retrying the service check again.
check_interval	This directive is used to define the number of "time units" to wait before scheduling the next "regular" check of the service. "Regular" checks are those that occur when the service is in an OK state or when the service is in a non-OK state, but has already been rechecked max_check_attempts number of times. Unless you've changed the interval_length directive from the default value of 60, this number will mean minutes. More information on this value can be found in the check scheduling documentation.
retry_interval	This directive is used to define the number of "time units" to wait before scheduling a re-check of the service. Services are rescheduled at the retry interval when they have changed to a non-OK state. Once the service has been retried max_check_attempts times without a change in its status, it will revert to being scheduled at its "normal" rate as defined by the check_interval value. Unless you've changed the interval_length directive from the default value of 60, this number will mean minutes. More information on this value can be found in the check scheduling documentation.
active_checks_enabled	This directive is used to determine whether or not active checks of this service are enabled. Values: 0 = disable active service checks, 1 = enable active service checks (default).
passive_checks_enabled	This directive is used to determine whether or not passive checks of this service are enabled. Values: 0 = disable passive service checks, 1 = enable passive service checks (default).
check_period	This directive is used to specify the short name of the time period during which active checks of this service can be made.
obsess_over_service	This directive determines whether or not checks for the service will be "obsessed" over using the ocsp_command .
check_freshness	This directive is used to determine whether or not freshness checks are enabled for this service. Values: 0 = disable freshness checks, 1 = enable freshness checks (default).
freshness_threshold	This directive is used to specify the freshness threshold (in seconds) for this service. If you set this directive to a value of 0, Nagios will determine a freshness threshold to use automatically.
event_handler	This directive is used to specify the <i>short name</i> of the command that should be run whenever a change in the state of the service is detected (i.e. whenever it goes down or recovers). Read the documentation on event handlers for a more detailed explanation of how to write scripts for handling events. The maximum amount of time that the event handler command can run is controlled by the event_handler_timeout option.
event_handler_enabled	This directive is used to determine whether or not the event handler for this service is enabled. Values: 0 = disable service event handler, 1 = enable service event handler.
low_flap_threshold	This directive is used to specify the low state change threshold used in flap detection for this service. More information on flap detection can be found here . If you set this directive to a value of 0, the program-wide value specified by the low_service_flap_threshold directive will be used.
high_flap_threshold	This directive is used to specify the high state change threshold used in flap detection for this service. More information on flap detection can be found here . If you set this directive to a value of 0, the program-wide value specified by the high_service_flap_threshold directive will be used.
flap_detection_enabled	This directive is used to determine whether or not flap detection is enabled for this

	service. More information on flap detection can be found here . Values: 0 = disable service flap detection, 1 = enable service flap detection.
flap_detection_options	This directive is used to determine what service states the flap detection logic will use for this service. Valid options are a combination of one or more of the following: o = OK states, w = WARNING states, c = CRITICAL states, u = UNKNOWN states.
process_perf_data	This directive is used to determine whether or not the processing of performance data is enabled for this service. Values: 0 = disable performance data processing, 1 = enable performance data processing.
retain_status_information	This directive is used to determine whether or not status-related information about the service is retained across program restarts. This is only useful if you have enabled state retention using the retain_state_information directive. Value: 0 = disable status information retention, 1 = enable status information retention.
retain_nonstatus_information	This directive is used to determine whether or not non-status information about the service is retained across program restarts. This is only useful if you have enabled state retention using the retain_state_information directive. Value: 0 = disable non-status information retention, 1 = enable non-status information retention.
notification_interval	This directive is used to define the number of "time units" to wait before re-notifying a contact that this service is <i>still</i> in a non-OK state. Unless you've changed the interval_length directive from the default value of 60, this number will mean minutes. If you set this value to 0, Nagios will <i>not</i> re-notify contacts about problems for this service - only one problem notification will be sent out.
first_notification_delay	This directive is used to define the number of "time units" to wait before sending out the first problem notification when this service enters a non-OK state. Unless you've changed the interval_length directive from the default value of 60, this number will mean minutes. If you set this value to 0, Nagios will start sending out notifications immediately.
notification_period	This directive is used to specify the short name of the time period during which notifications of events for this service can be sent out to contacts. No service notifications will be sent out during times which is not covered by the time period.
notification_options	This directive is used to determine when notifications for the service should be sent out. Valid options are a combination of one or more of the following: w = send notifications on a WARNING state, u = send notifications on an UNKNOWN state, c = send notifications on a CRITICAL state, r = send notifications on recoveries (OK state), f = send notifications when the service starts and stops flapping , and s = send notifications when scheduled downtime starts and ends. If you specify n (none) as an option, no service notifications will be sent out. If you do not specify any notification options, Nagios will assume that you want notifications to be sent out for all possible states. Example: If you specify w,r in this field, notifications will only be sent out when the service goes into a WARNING state and when it recovers from a WARNING state.
notifications_enabled	This directive is used to determine whether or not notifications for this service are enabled. Values: 0 = disable service notifications, 1 = enable service notifications.
contacts	This is a list of the <i>short names</i> of the contacts that should be notified whenever there are problems (or recoveries) with this service. Multiple contacts should be separated by commas. Useful if you want notifications to go to just a few people and don't want to configure contact groups . You must specify at least one contact or contact group in each service definition.
contact_groups	This is a list of the <i>short names</i> of the contact groups that should be notified whenever there are problems (or recoveries) with this service. Multiple contact groups should be separated by commas. You must specify at least one contact or contact group in each service definition.
stalking_options	This directive determines which service states "stalking" is enabled for. Valid options are a combination of one or more of the following: o = stalk on OK states, w = stalk on WARNING states, u = stalk on UNKNOWN states, and c = stalk on CRITICAL states. More information on state stalking can be found here .

notes	This directive is used to define an optional string of notes pertaining to the service. If you specify a note here, you will see the it in the extended information CGI (when you are viewing information about the specified service).
notes_url	This directive is used to define an optional URL that can be used to provide more information about the service. If you specify an URL, you will see a red folder icon in the CGIs (when you are viewing service information) that links to the URL you specify here. Any valid URL can be used. If you plan on using relative paths, the base path will be the same as what is used to access the CGIs (i.e. <i>/cgi-bin/nagios/</i>). This can be very useful if you want to make detailed information on the service, emergency contact methods, etc. available to other support staff.
action_url	This directive is used to define an optional URL that can be used to provide more actions to be performed on the service. If you specify an URL, you will see a red "splat" icon in the CGIs (when you are viewing service information) that links to the URL you specify here. Any valid URL can be used. If you plan on using relative paths, the base path will be the same as what is used to access the CGIs (i.e. <i>/cgi-bin/nagios/</i>).
icon_image	This variable is used to define the name of a GIF, PNG, or JPG image that should be associated with this service. This image will be displayed in the status and extended information CGIs. The image will look best if it is 40x40 pixels in size. Images for services are assumed to be in the logos/ subdirectory in your HTML images directory (i.e. <i>/usr/local/nagios/share/images/logos</i>).
icon_image_alt	This variable is used to define an optional string that is used in the ALT tag of the image specified by the <i><icon_image></i> argument. The ALT tag is used in the status , extended information and statusmap CGIs.

Service Group Definition

Definition Format

Note: Directives followed by a * are required.

```
define servicegroup{
    servicegroup_name*           servicegroup_name
    alias*                       alias
    members                       services
    servicegroup_members         servicegroups
    notes                         note_string
    notes_url                     url
    action_url                   url
}
```

Example Definition

```
define servicegroup{
    servicegroup_name           dbservices
    alias                       Database Services
    members                     ms1,SQL Server,ms1,SQL Server Agent,ms1,SQL DTC
}
```

Directive Descriptions

servicegroup_name	This directive is used to define a short name used to identify the service group.
alias	This directive is used to define is a longer name or description used to identify the service group. It is provided in order to allow you to more easily identify a particular service group.

members	<p>This is a list of the <i>descriptions</i> of services (and the names of their corresponding hosts) that should be included in this group. Host and service names should be separated by commas. This directive may be used as an alternative to the <i>servicegroups</i> directive in service definitions. The format of the member directive is as follows (note that a host name must precede a service name/description):</p> <pre>members=<host1>,<service1>,<host2>,<service2>,...,<hostn>,<servicen></pre>
servicegroup_members	<p>This optional directive can be used to include services from other "sub" service groups in this service group. Specify a comma-delimited list of short names of other service groups whose members should be included in this group.</p>
notes	<p>This directive is used to define an optional string of notes pertaining to the service group. If you specify a note here, you will see the it in the extended information CGI (when you are viewing information about the specified service group).</p>
notes_url	<p>This directive is used to define an optional URL that can be used to provide more information about the service group. If you specify an URL, you will see a red folder icon in the CGIs (when you are viewing service group information) that links to the URL you specify here. Any valid URL can be used. If you plan on using relative paths, the base path will be the same as what is used to access the CGIs (i.e. <i>/cgi-bin/nagios/</i>). This can be very useful if you want to make detailed information on the service group, emergency contact methods, etc. available to other support staff.</p>
action_url	<p>This directive is used to define an optional URL that can be used to provide more actions to be performed on the service group. If you specify an URL, you will see a red "splat" icon in the CGIs (when you are viewing service group information) that links to the URL you specify here. Any valid URL can be used. If you plan on using relative paths, the base path will be the same as what is used to access the CGIs (i.e. <i>/cgi-bin/nagios/</i>).</p>

Contact Definition

Definition Format

Note: Directives followed by a * are required.

```
define contact {
    contact_name*                contact_name
    alias                          alias
    contactgroups                   contactgroup_names
    host_notifications_enabled*  [0/1]
    service_notifications_enabled*  [0/1]
    host_notification_period*    timeperiod_name
    service_notification_period* timeperiod_name
    host_notification_options*  [d,u,r,f,s,n]
    service_notification_options* [w,u,c,r,f,s,n]
    host_notification_commands* command_name
    service_notification_commands* command_name
    email                           email_address
    pager                           pager_number or pager_email_gateway
    addressx                         additional_contact_address
    can_submit_commands              [0/1]
    retain_status_information         [0/1]
    retain_nonstatus_information     [0/1]
}
```

Example Definition

```

define contact{
    contact_name           jdoe
    alias                  John Doe
    host_notifications_enabled      1
    service_notifications_enabled  1
    service_notification_period    24x7
    host_notification_period      24x7
    service_notification_options   w,u,c,r
    host_notification_options      d,u,r
    service_notification_commands  notify-by-email
    host_notification_commands     host-notify-by-email
    email                  jdoe@localhost.localdomain
    pager                  555-5555@pagergateway.localhost.localdomain
    address1               xxxxx.xyyy@icq.com
    address2               555-555-5555
    can_submit_commands    1
}

```

Directive Descriptions

contact_name	This directive is used to define a short name used to identify the contact. It is referenced in contact group definitions. Under the right circumstances, the \$CONTACTNAME\$ macro will contain this value.
alias	This directive is used to define a longer name or description for the contact. Under the right circumstances, the \$CONTACTALIAS\$ macro will contain this value. If not specified, the <i>contact_name</i> will be used as the alias.
contactgroups	This directive is used to identify the <i>short name(s)</i> of the contactgroup(s) that the contact belongs to. Multiple contactgroups should be separated by commas. This directive may be used as an alternative to (or in addition to) using the <i>members</i> directive in contactgroup definitions.
host_notifications_enabled	This directive is used to determine whether or not the contact will receive notifications about host problems and recoveries. Values: 0 = don't send notifications, 1 = send notifications.
service_notifications_enabled	This directive is used to determine whether or not the contact will receive notifications about service problems and recoveries. Values: 0 = don't send notifications, 1 = send notifications.
host_notification_period	This directive is used to specify the short name of the time period during which the contact can be notified about host problems or recoveries. You can think of this as an "on call" time for host notifications for the contact. Read the documentation on time periods for more information on how this works and potential problems that may result from improper use.
service_notification_period	This directive is used to specify the short name of the time period during which the contact can be notified about service problems or recoveries. You can think of this as an "on call" time for service notifications for the contact. Read the documentation on time periods for more information on how this works and potential problems that may result from improper use.
host_notification_commands	This directive is used to define a list of the <i>short names</i> of the commands used to notify the contact of a <i>host</i> problem or recovery. Multiple notification commands should be separated by commas. All notification commands are executed when the contact needs to be notified. The maximum amount of time that a notification command can run is controlled by the notification_timeout option.
host_notification_options	This directive is used to define the host states for which notifications can be sent out to this contact. Valid options are a combination of one or more of the following: d = notify on DOWN host states, u = notify on UNREACHABLE host states, r = notify on host recoveries (UP states), f = notify when the host starts and stops flapping , and

	s = send notifications when host or service scheduled downtime starts and ends. If you specify n (none) as an option, the contact will not receive any type of host notifications.
service_notification_options	This directive is used to define the service states for which notifications can be sent out to this contact. Valid options are a combination of one or more of the following: w = notify on WARNING service states, u = notify on UNKNOWN service states, c = notify on CRITICAL service states, r = notify on service recoveries (OK states), and f = notify when the service starts and stops flapping . If you specify n (none) as an option, the contact will not receive any type of service notifications.
service_notification_commands	This directive is used to define a list of the <i>short names</i> of the commands used to notify the contact of a <i>service</i> problem or recovery. Multiple notification commands should be separated by commas. All notification commands are executed when the contact needs to be notified. The maximum amount of time that a notification command can run is controlled by the notification_timeout option.
email	This directive is used to define an email address for the contact. Depending on how you configure your notification commands, it can be used to send out an alert email to the contact. Under the right circumstances, the \$CONTACTEMAIL\$ macro will contain this value.
pager	This directive is used to define a pager number for the contact. It can also be an email address to a pager gateway (i.e. pagejoe@pagenet.com). Depending on how you configure your notification commands, it can be used to send out an alert page to the contact. Under the right circumstances, the \$CONTACTPAGERS\$ macro will contain this value.
addressx	Address directives are used to define additional "addresses" for the contact. These addresses can be anything - cell phone numbers, instant messaging addresses, etc. Depending on how you configure your notification commands, they can be used to send out an alert to the contact. Up to six addresses can be defined using these directives (<i>address1</i> through <i>address6</i>). The \$CONTACTADDRESSx\$ macro will contain this value.
can_submit_commands	This directive is used to determine whether or not the contact can submit external commands to Nagios from the CGIs. Values: 0 = don't allow contact to submit commands, 1 = allow contact to submit commands.
retain_status_information	This directive is used to determine whether or not status-related information about the contact is retained across program restarts. This is only useful if you have enabled state retention using the retain_state_information directive. Value: 0 = disable status information retention, 1 = enable status information retention.
retain_nonstatus_information	This directive is used to determine whether or not non-status information about the contact is retained across program restarts. This is only useful if you have enabled state retention using the retain_state_information directive. Value: 0 = disable non-status information retention, 1 = enable non-status information retention.

Contact Group Definition

Definition Format

Note: Directives followed by a * are required.

```
define contactgroup{
    contactgroup_name*           contactgroup_name
    alias*                       alias
    members                       contacts
    contactgroup_members          contactgroups
}
```

Example Definition

```
define contactgroup{
    contactgroup_name      novell-admins
    alias                  Novell Administrators
    members                jdoe,rtobert,tzach
}
```

Directive Descriptions

contactgroup_name	This directive is a short name used to identify the contact group.
alias	This directive is used to define a longer name or description used to identify the contact group.
members	This optional directive is used to define a list of the <i>short names</i> of contacts that should be included in this group. Multiple contact names should be separated by commas. This directive may be used as an alternative to (or in addition to) using the <i>contactgroups</i> directive in contact definitions.
contactgroup_members	This optional directive can be used to include contacts from other "sub" contact groups in this contact group. Specify a comma-delimited list of short names of other contact groups whose members should be included in this group.

Time Period Definition

Definition Format

Note: Directives followed by a * are required.

```
define timeperiod{
    timeperiod_name*      timeperiod_name
    alias*                alias
    [weekday]              timeranges
    [exception]            timeranges
    exclude                [timeperiod1,timeperiod2,...,timeperiodn]
}
```

Example Definitions

```
define timeperiod{
    timeperiod_name      nonworkhours
    alias                Non-Work Hours
    sunday               00:00-24:00           ; Every Sunday of every week
    monday               00:00-09:00,17:00-24:00 ; Every Monday of every week
    tuesday              00:00-09:00,17:00-24:00 ; Every Tuesday of every week
    wednesday            00:00-09:00,17:00-24:00 ; Every Wednesday of every week
    thursday             00:00-09:00,17:00-24:00 ; Every Thursday of every week
    friday               00:00-09:00,17:00-24:00 ; Every Friday of every week
    saturday             00:00-24:00           ; Every Saturday of every week
}
```

Directive Descriptions

timeperiod_name	This directives is the short name used to identify the time period.
alias	This directive is a longer name or description used to identify the time period.
[weekday]	The weekday directives (" <i>sunday</i> " through " <i>saturday</i> ")are comma-delimited lists of time ranges that are "valid" times for a particular day of the week. Notice that there are seven different days for which you can define time ranges (Sunday through Saturday). Each time range is in the form of HH:MM-HH:MM , where hours are specified on a 24 hour clock. For example, 00:15-24:00

	means 12:15am in the morning for this day until 12:00am midnight (a 23 hour, 45 minute total time range). If you wish to exclude an entire day from the timeperiod, simply do not include it in the timeperiod definition.
[exception]	You can specify several different types of exceptions to the standard rotating weekday schedule. Exceptions can take a number of different forms including single days of a specific or generic month, single weekdays in a month, or single calendar dates. You can also specify a range of days/dates and even specify skip intervals to obtain functionality described by "every 3 days between these dates". Rather than list all the possible formats for exception strings, I'll let you look at the example timeperiod definitions above to see what's possible. :-) Weekdays and different types of exceptions all have different levels of precedence, so its important to understand how they can affect each other. More information on this can be found in the documentation on timeperiods .
exclude	This directive is used to specify the short names of other timeperiod definitions whose time ranges should be excluded from this timeperiod. Multiple timeperiod names should be separated with a comma.

Command Definition

Definition Format

Note: Directives followed by a * are required.

```
define command{
    command_name*           command_name
    command_line*         command_line
}
```

Example Definition

```
define command{
    command_name    check_pop
    command_line    /usr/local/nagios/libexec/check_pop -H $HOSTADDRESS$
}
```

Directive Descriptions

command_name	This directive is the short name used to identify the command. It is referenced in contact , host , and service definitions (in notification, check, and event handler directives), among other places.
command_line	<p>This directive is used to define what is actually executed by Nagios when the command is used for service or host checks, notifications, or event handlers. Before the command line is executed, all valid macros are replaced with their respective values. See the documentation on macros for determining when you can use different macros. Note that the command line is <i>not</i> surrounded in quotes. Also, if you want to pass a dollar sign (\$) on the command line, you have to escape it with another dollar sign.</p> <p>NOTE: You may not include a semicolon (;) in the <i>command_line</i> directive, because everything after it will be ignored as a config file comment. You can work around this limitation by setting one of the \$USERS\$ macros in your resource file to a semicolon and then referencing the appropriate \$USERS\$ macro in the <i>command_line</i> directive in place of the semicolon.</p> <p>If you want to pass arguments to commands during runtime, you can use \$ARGn\$ macros in the <i>command_line</i> directive of the command definition and then separate individual arguments from the command name (and from each other) using bang (!) characters in the object definition directive (host check command, service event handler command, etc) that references the command. More information on how arguments in command definitions are processed during runtime can be found in the documentation on macros.</p>

Service Dependency Definition

Definition Format

Note: Directives followed by a * are required. However, you must supply at least one type of criteria for the definition to be of much use.

```
define servicedependency{
    dependent_host_name*           host_name
    dependent_hostgroup_name       hostgroup_name
    dependent_service_description* service_description
    host_name*                   host_name
    hostgroup_name                 hostgroup_name
    service_description*         service_description
    inherits_parent                [0/1]
    execution_failure_criteria     [o,w,u,c,p,n]
    notification_failure_criteria  [o,w,u,c,p,n]
    dependency_period              timeperiod_name
}
```

Example Definition

```
define servicedependency{
    host_name           WWW1
    service_description Apache Web Server
    dependent_host_name WWW1
    dependent_service_description Main Web Site
    execution_failure_criteria n
    notification_failure_criteria w, u, c
}
```

Directive Descriptions

dependent_host_name	This directive is used to identify the <i>short name(s)</i> of the host(s) that the <i>dependent</i> service "runs" on or is associated with. Multiple hosts should be separated by commas. Leaving this directive blank can be used to create "same host" dependencies .
dependent_hostgroup_name	This directive is used to specify the <i>short name(s)</i> of the hostgroup(s) that the <i>dependent</i> service "runs" on or is associated with. Multiple hostgroups should be separated by commas. The dependent_hostgroup may be used instead of, or in addition to, the dependent_host directive.
dependent_service_description	This directive is used to identify the <i>description</i> of the <i>dependent</i> service .
host_name	This directive is used to identify the <i>short name(s)</i> of the host(s) that the service <i>that is being depended upon</i> (also referred to as the master service) "runs" on or is associated with. Multiple hosts should be separated by commas.
hostgroup_name	This directive is used to identify the <i>short name(s)</i> of the hostgroup(s) that the service <i>that is being depended upon</i> (also referred to as the master service) "runs" on or is associated with. Multiple hostgroups should be separated by commas. The hostgroup_name may be used instead of, or in addition to, the host_name directive.
service_description	This directive is used to identify the <i>description</i> of the service <i>that is being depended upon</i> (also referred to as the master service).
inherits_parent	This directive indicates whether or not the dependency inherits dependencies of the service <i>that is being depended upon</i> (also referred to as the master service). In other words, if the master service is dependent upon other services and any one of those dependencies fail, this dependency will also fail.

execution_failure_criteria	This directive is used to specify the criteria that determine when the dependent service should <i>not</i> be actively checked. If the <i>master</i> service is in one of the failure states we specify, the <i>dependent</i> service will not be actively checked. Valid options are a combination of one or more of the following (multiple options are separated with commas): o = fail on an OK state, w = fail on a WARNING state, u = fail on an UNKNOWN state, c = fail on a CRITICAL state, and p = fail on a pending state (e.g. the service has not yet been checked). If you specify n (none) as an option, the execution dependency will never fail and checks of the dependent service will always be actively checked (if other conditions allow for it to be). Example: If you specify o,c,u in this field, the <i>dependent</i> service will not be actively checked if the <i>master</i> service is in either an OK, a CRITICAL, or an UNKNOWN state.
notification_failure_criteria	This directive is used to define the criteria that determine when notifications for the dependent service should <i>not</i> be sent out. If the <i>master</i> service is in one of the failure states we specify, notifications for the <i>dependent</i> service will not be sent to contacts. Valid options are a combination of one or more of the following: o = fail on an OK state, w = fail on a WARNING state, u = fail on an UNKNOWN state, c = fail on a CRITICAL state, and p = fail on a pending state (e.g. the service has not yet been checked). If you specify n (none) as an option, the notification dependency will never fail and notifications for the dependent service will always be sent out. Example: If you specify w in this field, the notifications for the <i>dependent</i> service will not be sent out if the <i>master</i> service is in a WARNING state.
dependency_period	This directive is used to specify the short name of the time period during which this dependency is valid. If this directive is not specified, the dependency is considered to be valid during all times.

Service Escalation Definition

Definition Format

Note: Directives followed by a * are required.

```
define serviceescalation{
    host_name*                host_name
    hostgroup_name
    service_description*    service_description
    contacts*                contacts
    contact_groups*         contactgroup_name
    first_notification*     #
    last_notification*      #
    notification_interval*  #
    escalation_period        timeperiod_name
    escalation_options      [w,u,c,r]
}
```

Example Definition

```
define serviceescalation{
    host_name                nt-3
    service_description      Processor Load
    first_notification       4
    last_notification        0
    notification_interval    30
    contact_groups           all-nt-admins,themanagers
}
```

Directive Descriptions

host_name	This directive is used to identify the <i>short name(s)</i> of the host(s) that the service escalation should apply to or is associated with.
hostgroup_name	This directive is used to specify the <i>short name(s)</i> of the hostgroup(s) that the service escalation should apply to or is associated with. Multiple hostgroups should be separated by commas. The hostgroup_name may be used instead of, or in addition to, the host_name directive.
service_description	This directive is used to identify the <i>description</i> of the service the escalation should apply to.
first_notification	This directive is a number that identifies the <i>first</i> notification for which this escalation is effective. For instance, if you set this value to 3, this escalation will only be used if the service is in a non-OK state long enough for a third notification to go out.
last_notification	This directive is a number that identifies the <i>last</i> notification for which this escalation is effective. For instance, if you set this value to 5, this escalation will not be used if more than five notifications are sent out for the service. Setting this value to 0 means to keep using this escalation entry forever (no matter how many notifications go out).
contacts	This is a list of the <i>short names</i> of the contacts that should be notified whenever there are problems (or recoveries) with this service. Multiple contacts should be separated by commas. Useful if you want notifications to go to just a few people and don't want to configure contact groups . You must specify at least one contact or contact group in each service escalation definition.
contact_groups	This directive is used to identify the <i>short name</i> of the contact group that should be notified when the service notification is escalated. Multiple contact groups should be separated by commas. You must specify at least one contact or contact group in each service escalation definition.
notification_interval	This directive is used to determine the interval at which notifications should be made while this escalation is valid. If you specify a value of 0 for the interval, Nagios will send the first notification when this escalation definition is valid, but will then prevent any more problem notifications from being sent out for the host. Notifications are sent out again until the host recovers. This is useful if you want to stop having notifications sent out after a certain amount of time. Note: If multiple escalation entries for a host overlap for one or more notification ranges, the smallest notification interval from all escalation entries is used.
escalation_period	This directive is used to specify the short name of the time period during which this escalation is valid. If this directive is not specified, the escalation is considered to be valid during all times.
escalation_options	This directive is used to define the criteria that determine when this service escalation is used. The escalation is used only if the service is in one of the states specified in this directive. If this directive is not specified in a service escalation, the escalation is considered to be valid during all service states. Valid options are a combination of one or more of the following: r = escalate on an OK (recovery) state, w = escalate on a WARNING state, u = escalate on an UNKNOWN state, and c = escalate on a CRITICAL state. Example: If you specify w in this field, the escalation will only be used if the service is in a WARNING state.

Host Dependency Definition

Definition Format

Note: Directives followed by a * are required.

```
define hostdependency{
    dependent_host_name*           host_name
    dependent_hostgroup_name       hostgroup_name
    host_name*                   host_name
    hostgroup_name                 hostgroup_name
}
```

```

inherits_parent           [0/1]
execution_failure_criteria [o,d,u,p,n]
notification_failure_criteria [o,d,u,p,n]
dependency_period        timeperiod_name
}

```

Example Definition

```

define hostdependency{
    host_name                WWW1
    dependent_host_name      DBASE1
    notification_failure_criteria d,u
}

```

Directive Descriptions

dependent_host_name	This directive is used to identify the <i>short name(s)</i> of the <i>dependent</i> host(s) . Multiple hosts should be separated by commas.
dependent_hostgroup_name	This directive is used to identify the <i>short name(s)</i> of the <i>dependent</i> hostgroup(s) . Multiple hostgroups should be separated by commas. The <code>dependent_hostgroup_name</code> may be used instead of, or in addition to, the <code>dependent_host_name</code> directive.
host_name	This directive is used to identify the <i>short name(s)</i> of the host(s) <i>that is being depended upon</i> (also referred to as the master host). Multiple hosts should be separated by commas.
hostgroup_name	This directive is used to identify the <i>short name(s)</i> of the hostgroup(s) <i>that is being depended upon</i> (also referred to as the master host). Multiple hostgroups should be separated by commas. The <code>hostgroup_name</code> may be used instead of, or in addition to, the <code>host_name</code> directive.
inherits_parent	This directive indicates whether or not the dependency inherits dependencies of the host <i>that is being depended upon</i> (also referred to as the master host). In other words, if the master host is dependent upon other hosts and any one of those dependencies fail, this dependency will also fail.
execution_failure_criteria	This directive is used to specify the criteria that determine when the dependent host should <i>not</i> be actively checked. If the <i>master</i> host is in one of the failure states we specify, the <i>dependent</i> host will not be actively checked. Valid options are a combination of one or more of the following (multiple options are separated with commas): o = fail on an UP state, d = fail on a DOWN state, u = fail on an UNREACHABLE state, and p = fail on a pending state (e.g. the host has not yet been checked). If you specify n (none) as an option, the execution dependency will never fail and the dependent host will always be actively checked (if other conditions allow for it to be). Example: If you specify u,d in this field, the <i>dependent</i> host will not be actively checked if the <i>master</i> host is in either an UNREACHABLE or DOWN state.
notification_failure_criteria	This directive is used to define the criteria that determine when notifications for the dependent host should <i>not</i> be sent out. If the <i>master</i> host is in one of the failure states we specify, notifications for the <i>dependent</i> host will not be sent to contacts. Valid options are a combination of one or more of the following: o = fail on an UP state, d = fail on a DOWN state, u = fail on an UNREACHABLE state, and p = fail on a pending state (e.g. the host has not yet been checked). If you specify n (none) as an option, the notification dependency will never fail and notifications for the dependent host will always be sent out. Example: If you specify d in this field, the notifications for the <i>dependent</i> host will not be sent out if the <i>master</i> host is in a DOWN state.
dependency_period	This directive is used to specify the short name of the time period during which this dependency is valid. If this directive is not specified, the dependency is considered to be valid during all times.

Host Escalation Definition

Definition Format

Note: Directives followed by a * are required.

```
define hostescalation{
    host_name*                host_name
    hostgroup_name          hostgroup_name
    contacts*              contacts
    contact_groups*       contactgroup_name
    first_notification*    #
    last_notification*     #
    notification_interval* #
    escalation_period      timeperiod_name
    escalation_options    [d,u,r]
}
```

Example Definition

```
define hostescalation{
    host_name            router-34
    first_notification   5
    last_notification    8
    notification_interval 60
    contact_groups       all-router-admins
}
```

Directive Descriptions

host_name	This directive is used to identify the <i>short name</i> of the host that the escalation should apply to.
hostgroup_name	This directive is used to identify the <i>short name(s)</i> of the hostgroup(s) that the escalation should apply to. Multiple hostgroups should be separated by commas. If this is used, the escalation will apply to all hosts that are members of the specified hostgroup(s).
first_notification	This directive is a number that identifies the <i>first</i> notification for which this escalation is effective. For instance, if you set this value to 3, this escalation will only be used if the host is down or unreachable long enough for a third notification to go out.
last_notification	This directive is a number that identifies the <i>last</i> notification for which this escalation is effective. For instance, if you set this value to 5, this escalation will not be used if more than five notifications are sent out for the host. Setting this value to 0 means to keep using this escalation entry forever (no matter how many notifications go out).
contacts	This is a list of the <i>short names</i> of the contacts that should be notified whenever there are problems (or recoveries) with this host. Multiple contacts should be separated by commas. Useful if you want notifications to go to just a few people and don't want to configure contact groups . You must specify at least one contact or contact group in each host escalation definition.
contact_groups	This directive is used to identify the <i>short name</i> of the contact group that should be notified when the host notification is escalated. Multiple contact groups should be separated by commas. You must specify at least one contact or contact group in each host escalation definition.
notification_interval	This directive is used to determine the interval at which notifications should be made while this escalation is valid. If you specify a value of 0 for the interval, Nagios will send the first notification when this escalation definition is valid, but will then prevent any more problem notifications from being sent out for the host. Notifications are sent out again until the host recovers. This is useful if you want to stop having notifications sent out after a certain amount

	of time. Note: If multiple escalation entries for a host overlap for one or more notification ranges, the smallest notification interval from all escalation entries is used.
escalation_period	This directive is used to specify the short name of the time period during which this escalation is valid. If this directive is not specified, the escalation is considered to be valid during all times.
escalation_options	This directive is used to define the criteria that determine when this host escalation is used. The escalation is used only if the host is in one of the states specified in this directive. If this directive is not specified in a host escalation, the escalation is considered to be valid during all host states. Valid options are a combination of one or more of the following: r = escalate on an UP (recovery) state, d = escalate on a DOWN state, and u = escalate on an UNREACHABLE state. Example: If you specify d in this field, the escalation will only be used if the host is in a DOWN state.

Extended Host Information Definition

Definition Format

Note: Directives followed by a * are required. However, you need to supply at least one optional variable in each definition for it to be of much use.

```
define hostextinfo{
    host_name*           host_name
    notes                 note_string
    notes_url             url
    action_url            url
    icon_image            image_file
    icon_image_alt        alt_string
    vrmf_image            image_file
    statusmap_image       image_file
    2d_coords              x_coord,y_coord
    3d_coords              x_coord,y_coord,z_coord
}
```

Example Definition

```
define hostextinfo{
    host_name             netware1
    notes                 This is the primary Netware file server
    notes_url             http://localhost.localdomain/hostinfo.pl?host=netware1
    icon_image            novell40.png
    icon_image_alt        IntranetWare 4.11
    vrmf_image            novell40.png
    statusmap_image       novell40.gd2
    2d_coords              100,250
    3d_coords              100.0,50.0,75.0
}
```

Variable Descriptions

host_name	This variable is used to identify the <i>short name</i> of the host which the data is associated with.
notes	This directive is used to define an optional string of notes pertaining to the host. If you specify a note here, you will see the it in the extended information CGI (when you are viewing information about the specified host).
notes_url	This variable is used to define an optional URL that can be used to provide more information about the host. If you specify an URL, you will see a link that says "Extra Host Notes" in the

	extended information CGI (when you are viewing information about the specified host). Any valid URL can be used. If you plan on using relative paths, the base path will be the same as what is used to access the CGIs (i.e. <code>/cgi-bin/nagios/</code>). This can be very useful if you want to make detailed information on the host, emergency contact methods, etc. available to other support staff.
action_url	This directive is used to define an optional URL that can be used to provide more actions to be performed on the host. If you specify an URL, you will see a link that says "Extra Host Actions" in the extended information CGI (when you are viewing information about the specified host). Any valid URL can be used. If you plan on using relative paths, the base path will be the same as what is used to access the CGIs (i.e. <code>/cgi-bin/nagios/</code>).
icon_image	This variable is used to define the name of a GIF, PNG, or JPG image that should be associated with this host. This image will be displayed in the status and extended information CGIs. The image will look best if it is 40x40 pixels in size. Images for hosts are assumed to be in the logos/ subdirectory in your HTML images directory (i.e. <code>/usr/local/nagios/share/images/logos</code>).
icon_image_alt	This variable is used to define an optional string that is used in the ALT tag of the image specified by the <code><icon_image></code> argument. The ALT tag is used in the status , extended information and statusmap CGIs.
vrmf_image	This variable is used to define the name of a GIF, PNG, or JPG image that should be associated with this host. This image will be used as the texture map for the specified host in the statuswrl CGI. Unlike the image you use for the <code><icon_image></code> variable, this one should probably <i>not</i> have any transparency. If it does, the host object will look a bit wierd. Images for hosts are assumed to be in the logos/ subdirectory in your HTML images directory (i.e. <code>/usr/local/nagios/share/images/logos</code>).
statusmap_image	This variable is used to define the name of an image that should be associated with this host in the statusmap CGI. You can specify a JPEG, PNG, and GIF image if you want, although I would strongly suggest using a GD2 format image, as other image formats will result in a lot of wasted CPU time when the statusmap image is generated. GD2 images can be created from PNG images by using the pngtogd2 utility supplied with Thomas Boutell's gd library . The GD2 images should be created in <i>uncompressed</i> format in order to minimize CPU load when the statusmap CGI is generating the network map image. The image will look best if it is 40x40 pixels in size. You can leave these option blank if you are not using the statusmap CGI. Images for hosts are assumed to be in the logos/ subdirectory in your HTML images directory (i.e. <code>/usr/local/nagios/share/images/logos</code>).
2d_coords	This variable is used to define coordinates to use when drawing the host in the statusmap CGI. Coordinates should be given in positive integers, as they correspond to physical pixels in the generated image. The origin for drawing (0,0) is in the upper left hand corner of the image and extends in the positive x direction (to the right) along the top of the image and in the positive y direction (down) along the left hand side of the image. For reference, the size of the icons drawn is usually about 40x40 pixels (text takes a little extra space). The coordinates you specify here are for the upper left hand corner of the host icon that is drawn. Note: Don't worry about what the maximum x and y coordinates that you can use are. The CGI will automatically calculate the maximum dimensions of the image it creates based on the largest x and y coordinates you specify.
3d_coords	This variable is used to define coordinates to use when drawing the host in the statuswrl CGI. Coordinates can be positive or negative real numbers. The origin for drawing is (0.0,0.0,0.0). For reference, the size of the host cubes drawn is 0.5 units on each side (text takes a little more space). The coordinates you specify here are used as the center of the host cube.

Extended Service Information Definition

Definition Format

Note: Directives followed by a * are required. However, you need to supply at least one optional variable in each definition for it to be of much use.

```
define serviceextinfo {
```

```

host_name*                host_name
service_description*     service_description
notes                       note_string
notes_url                   url
action_url                  url
icon_image                  image_file
icon_image_alt              alt_string
}

```

Example Definition

```

define serviceextinfo{
    host_name                linux2
    service_description      Log Anomalies
    notes                    Security-related log anomalies
    notes_url                http://localhost.localdomain/serviceinfo.pl
    icon_image               security.png
    icon_image_alt           Security-Related Alerts
}

```

Variable Descriptions

host_name	This directive is used to identify the <i>short name</i> of the host that the service is associated with.
service_description	This directive is description of the service which the data is associated with.
notes	This directive is used to define an optional string of notes pertaining to the service. If you specify a note here, you will see the it in the extended information CGI (when you are viewing information about the specified service).
notes_url	This directive is used to define an optional URL that can be used to provide more information about the service. If you specify an URL, you will see a link that says "Extra Service Notes" in the extended information CGI (when you are viewing information about the specified service). Any valid URL can be used. If you plan on using relative paths, the base path will the the same as what is used to access the CGIs (i.e. <i>/cgi-bin/nagios/</i>). This can be very useful if you want to make detailed information on the service, emergency contact methods, etc. available to other support staff.
action_url	This directive is used to define an optional URL that can be used to provide more actions to be performed on the service. If you specify an URL, you will see a link that says "Extra Service Actions" in the extended information CGI (when you are viewing information about the specified service). Any valid URL can be used. If you plan on using relative paths, the base path will the the same as what is used to access the CGIs (i.e. <i>/cgi-bin/nagios/</i>).
icon_image	This variable is used to define the name of a GIF, PNG, or JPG image that should be associated with this host. This image will be displayed in the status and extended information CGIs. The image will look best if it is 40x40 pixels in size. Images for hosts are assumed to be in the logos/ subdirectory in your HTML images directory (i.e. <i>/usr/local/nagios/share/images/logos/</i>).
icon_image_alt	This variable is used to define an optional string that is used in the ALT tag of the image specified by the <code><icon_image></code> argument. The ALT tag is used in the status , extended information and statusmap CGIs.