

```
#!/bin/bash
```

```
#####  
#                                                                 #  
# Copyright (C) 2011 Stefan-Michael Guenther <s.guenther@in-put.de> #  
#                                                                 #  
# This program is free software; you can redistribute it and/or modify #  
# it under the terms of the GNU General Public License as published by #  
# the Free Software Foundation; either version 3 of the License, or #  
# (at your option) any later version. #  
#                                                                 #  
# This program is distributed in the hope that it will be useful, #  
# but WITHOUT ANY WARRANTY; without even the implied warranty of #  
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the #  
# GNU General Public License for more details. #  
#                                                                 #  
# You should have received a copy of the GNU General Public License #  
# along with this program; if not, write to the Free Software #  
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA #  
#                                                                 #  
#####
```

```
#####  
#                                                                 #  
# Nagios plugin to monitor the output of zarafa-stats #  
# of a Zarafa Server (www.zarafaserver.de) #  
#                                                                 #  
# Version 0.2 #  
# - Modified the statement for getting VAL #  
# - Added a check, whether the right parameter for a check was used #  
#                                                                 #  
# Version 0.1 #  
#                                                                 #  
#####
```

```
#####  
#                                                                 #  
# commands.cfg #  
#                                                                 #  
# define command { #  
#   command_name check_zarafa #  
#   command_line $USER1$/check_zarafa --param $ARG1$ -w $ARG2$ -c $ARG3$ #  
# } #  
#                                                                 #  
#####
```

```
VERSION="Version 0.2"  
AUTHOR="(c) 2011 Stefan-Michael Guenther <s.guenther@in-put.de>"
```

```
# zarafa-stats  
ZARAFAPROG=/usr/bin/zarafa-stats
```

```
# Exit codes  
STATE_OK=0  
STATE_WARNING=1  
STATE_CRITICAL=2  
STATE_UNKNOWN=3
```

```
shopt -s extglob
```

```
#### Functions ####
```

```
# Print version information  
print_version()  
{  
    printf "\n\n$0 - $VERSION\n"  
}
```

```
#Print help information
```

```
print_help()
{
    print_version
    printf "$AUTHOR\n"
    printf "Monitor queuelen, queueage, thread, threads_idle of a Zarafa server\n"
}
/bin/cat <<EOT
```

## Options:

```
-h
    Print detailed help screen
-V
    Print version information
-v
    Verbose output

--param VALUE
    Set what to query: queueln, queueage, threads, threads_idle. Default is threads_idle

-w INTEGER
    Exit with WARNING status if above INTEGER

-c INTEGER
    Exit with CRITICAL status if above INTEGER
```

```
EOT
}
```

```
##### MAIN #####
```

```
# Warning threshold
thresh_warn=
# Critical threshold
thresh_crit=
# Parameter to monitor
query=threads_idle
```

```
# See if we have the Zarafa server installed and can execute zarafa-stats
```

```
if [[ ! -x "$ZARAFAPROG" ]]; then
    printf "\nIt appears that you don't have the Zarafa server installed!\n"
    exit $STATE_UNKOWN
fi
```

```
# Parse command line options
```

```
while [[ -n "$1" ]]; do
    case "$1" in

        -h | --help)
            print_help
            exit $STATE_OK
            ;;

        -V | --version)
            print_version
            exit $STATE_OK
            ;;

        -v | --verbose)
            : $(( verbosity++ ))
            shift
            ;;

        -w | --warning)
            if [[ -z "$2" ]]; then
                # Threshold not provided
                printf "\nOption $1 requires an argument"
                print_help
                exit $STATE_UNKNOWN
            elif [[ "$2" = +([0-9]) ]]; then
                # Threshold is an integer
```

```

        thresh=$2
    else
        # Threshold is not an integer
        printf "\nThreshold must be an integer"
        print_help
        exit $STATE_UNKNOWN
    fi
    thresh_warn=$thresh
    shift 2
    ;;

-c | --critical)
    if [[ -z "$2" ]]; then
        # Threshold not provided
        printf "\nOption '$1' requires an argument"
        print_help
        exit $STATE_UNKNOWN
    elif [[ "$2" = +([0-9]) ]]; then
        # Threshold is an integer
        thresh=$2
    else
        # Threshold is not an integer
        printf "\nThreshold must be an integer"
        print_help
        exit $STATE_UNKNOWN
    fi
    thresh_crit=$thresh
    shift 2
    ;;

-?)
    print_help
    exit $STATE_OK
    ;;

--param)
    if [[ -z "$2" ]]; then
        printf "\nOption $1 requires an argument"
        print_help
        exit $EXIT_UNKNOWN
    fi
    query=$2
    shift 2
    ;;

*)
    printf "\nInvalid option '$1'"
    print_help
    exit $STATE_UNKNOWN
    ;;

esac
done

# Check if a query was specified
if [[ -z "$query" ]]; then
    # No query specified
    printf "\nNo query specified"
    print_help
    exit $STATE_UNKNOWN
fi

# Check whether the right parameter is used: 0x3001001E instead of 0x6740001E
#
CHECK=`${ZARAFAPROG} --system | egrep -i ": $query$" | grep -c "0x3001001E"`

if [ $CHECK = 0 ];
then
    echo "Wrong parameter to check!"

```

```
    exit $STATE_CRITICAL
fi

# Get the value
# The shell doesn't like decimals. Therefore we have to use the last cut command, just in case
# the query asks for the queueage
VAL=`${ZARAFAPROG} --system | egrep -i -A2 " : $query$" | tail -1 | cut -f 2 -d ' ' | cut -f 1 -d '.'`

# Check if the thresholds has been set correctly
if [[ -z "$thresh_warn" || -z "$thresh_crit" ]]; then
    # One or both thresholds were not specified
    printf "\nThreshold not set"
    print_help
    exit $STATE_UNKNOWN
elif [[ "$thresh_crit" -lt "$thresh_warn" ]]; then
    # The warning threshold must be lower than the critical threshold
    printf "\nWarning threshold should be lower than critical"
    print_help
    exit $STATE_UNKNOWN
fi

# Verbose output
if [[ "$verbosity" -ge 2 ]]; then
    /bin/cat <<_EOT
Debugging information:
Warning threshold: $thresh_warn
Critical threshold: $thresh_crit
Verbosity level: $verbosity
Current $query: $VAL
_EOT
    printf "\n Complete output of zarafa-stats --system:\n"
    ${ZARAFAPROG}
    printf "\n\n"
fi

# And finally check the retrun value against our thresholds
if [[ "$SVAL" -gt "$thresh_crit" ]]; then
    # Value is above critical threshold
    echo "$query CRITICAL - Value is $SVAL"
    exit $STATE_CRITICAL

elif [[ "$SVAL" -gt "$thresh_warn" ]]; then
    # Value is above warning threshold
    echo "$query WARNING - Value is $SVAL"
    exit $STATE_WARNING

else
    # Value is ok
    echo "$query OK - Value is $SVAL"
    exit $STATE_OK
fi
exit 3
```