

WLhc – User Guide

For WLhc version 1.2

Table of Contents

1.	What is WLhc?	3
1.2	Target Audience.....	3
1.2	Features	3
2.	Software Requirements.....	4
3.	WLhc Architecture	5
4.	Installing WLhc	6
5.	Executing WLhc	8
6.	Configuring WLhc.....	11
7.	Running WLhc in Batch mode	16
8.	Extending WLhc.....	17

1. What is WLhc?

WLhc (**WebLogic health checker**) is a CLI-based WebLogic Server health-checking software. It comprises of a bunch of Korn Shell scripts, WLSshell and a line of perl ;). I'm grateful to Paco Gomez for developing the very useful WLSshell (<http://www.wlshell.net>) software which is a shell that can be used to query Java MBeans. Basically, WLhc is only a wrapper around WLSshell, with some added features like traps and alarms. WLhc is free software.

1.2 Target Audience

WLhc does not have fancy GUIs because it's CLI-based. It was developed out of my need for a simple, flexible, free tool which can just externally monitor a WebLogic domain and log alarms when pre-defined thresholds are breached. Beginners in WebLogic or users who are daunted by the UNIX command-line may find WLhc difficult to use. WLhc's primary target audience is users who possess the following:

- Good working knowledge of WebLogic Server and the architecture of the domain being monitored
- Familiarity with UNIX and a UNIX shell
- An understanding of the concept of Java MBeans and how to identify WebLogic Server MBeans

1.2 Features

- Monitors WebLogic Server versions 7.0, 8.1, 9.x and 10.x
- Interactive and non-interactive modes of execution
- Unintrusive, external monitoring: No application is required to be deployed on the WebLogic Server being monitored and no server restart required.
- Interactive execution provides a well-formatted dashboard of current runtime values of monitored parameters
- Non-interactive execution (e.g. cron) generates data files in csv format, thereby allowing graphing programs to generate graphs from

the data. The data files are retained for 30 days by default and can be used for troubleshooting and trend analysis.

- Quick and easy Re-configuration: An auxiliary script controlled by a template, facilitates rebuilding the configuration file for a monitored server. This script will come in handy when changes are made to a WebLogic Server and/or when you wish to monitor new parameters.
- Auxiliary scripts obtain data on all runtime MBeans and their attributes for a specified WebLogic Server.
- Traps can be easily configured using simple rules in configuration files
- Traps may be enabled/disabled globally for one or more WebLogic Servers or for individual parameters
- Clear and structured logging for alarms and errors, thereby facilitating the use of log scanners

2. Software Requirements

- Korn Shell (/bin/ksh)
- Solaris 8/9/10 with nawk in PATH and /usr/ucb/ps

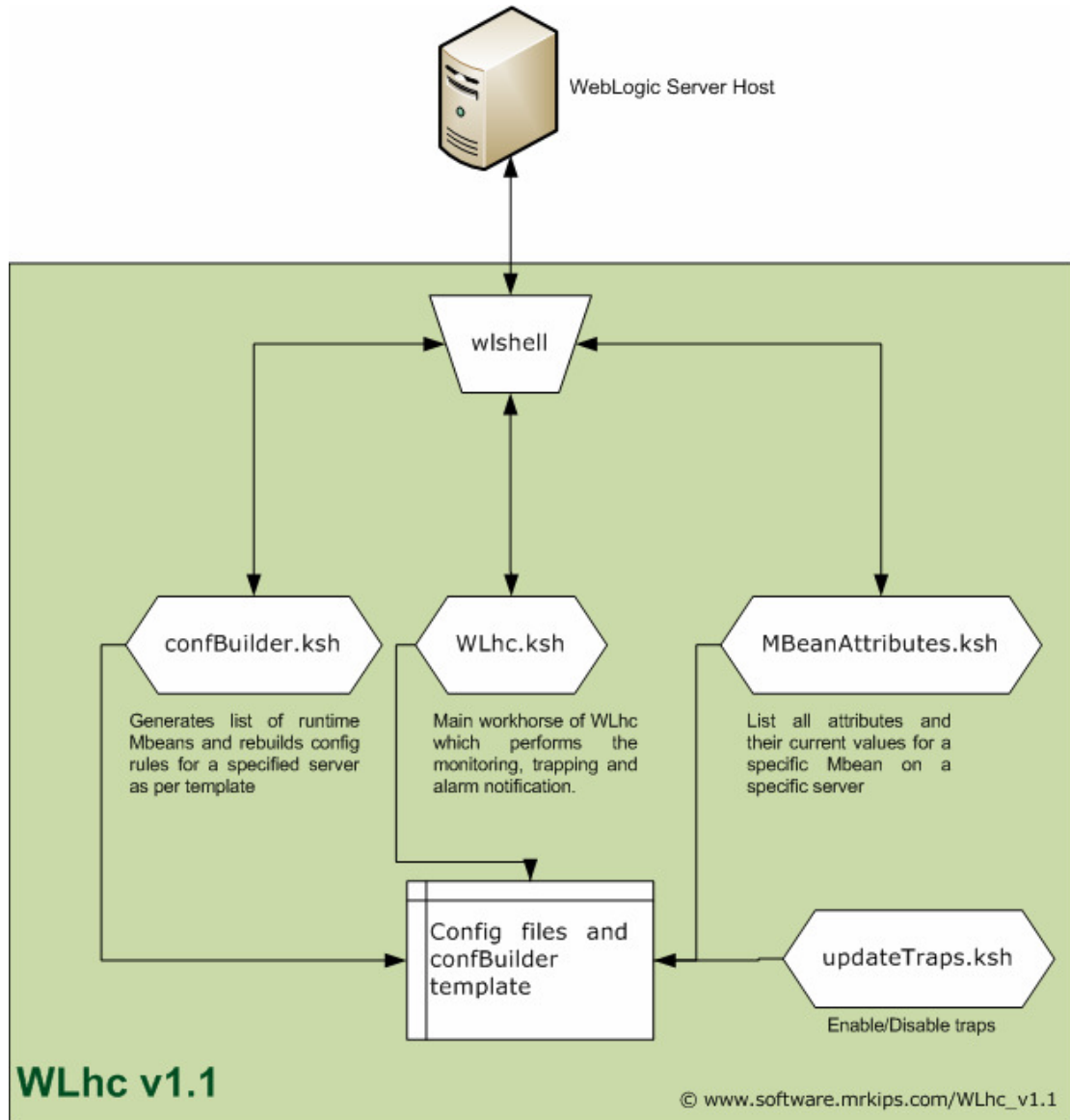
OR

Any Linux distribution with gawk in PATH

- Perl (perl in PATH)

Note: WLhc v1.2 may also work on other UNIX flavours with Korn Shell and appropriate awk and ps utilities, but it's been tested only with Solaris 9/10, RHEL 5.1 and Ubuntu Desktop 9.04

3. WLhc Architecture



ERRATA: In the diagram above, WLhc v1.1 should read as WLhc v1.2 and the copyright URL is www.it.cybergav.in/WLhc

4. Installing WLhc

The installation bundle for WLhc version 1.0 release is WLhc_v1.2.tar.gz. The installation steps are given below:

1. Uncompress and extract the installation files using any one of the following commands:

```
gzcat WLhc_v1.2.tar.gz | tar xvf -
```

OR

```
tar xvfz WLhc_v1.2.tar.gz
```

OR

```
gunzip WLhc_v1.2.tar.gz; tar xvf WLhc_v1.2.tar
```

After doing the above, you will see files `install.ksh`, `WLhc.cfg` and `WLhc_v1.2_installation.tar` extracted.

2. Configure installation parameters by editing the `WLhc.cfg` file and populating required values.

NOTE: Ensure that the `JAVA_DIR` parameter is configured with the JVM directory for the appropriate JVM (JVM supported by the version of WebLogic Server being monitored. For example, JVM 1.4.x for WebLogic 8.1 and JVM 1.5.x+ for WebLogic 9.x/10.x).

3. Execute the installation script as follows:

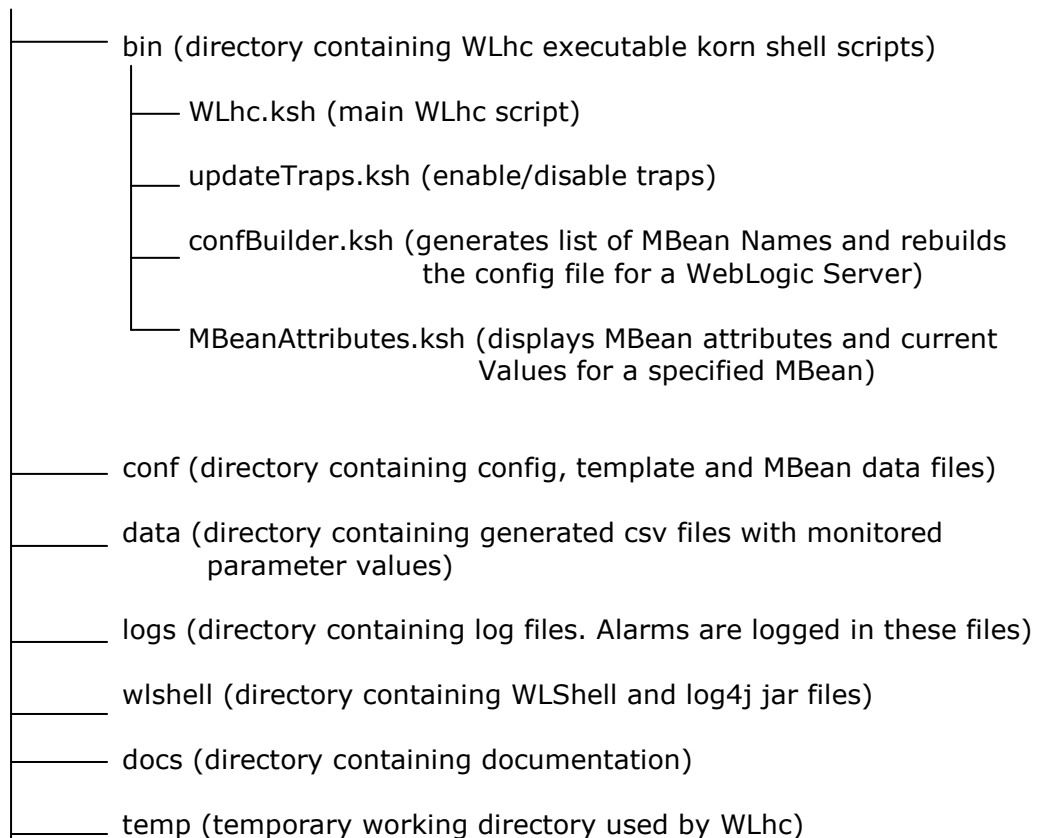
```
ksh install.ksh
```

When the installation script is executed, you will be asked for a WebLogic user username and password. This WebLogic user will be used to query MBeans and it is **strongly recommended** to use a WebLogic user which has been assigned **only Monitor role**, as the password will be present in plain-text format in the config files. You will then be prompted on whether you wish to create monitoring rules during installation. On large domains (several resources), it is recommended

NOT to create rules for the Admin server during installation as WLhc reads all domain MBeans for the admin server. However, you may choose to create rules for a Managed Server during installation.

The directory structure of the WLhc installation directory post-installation is given below:

WLhc Installation Directory



Successful installation of WLhc will ensure that a configuration file is created for each WebLogic Server to be monitored. If you chose to create monitoring rules for a server during installation, then a template in the conf directory (confBuilder.temp) is used to create the rules.

5. Executing WLhc

If you chose to create monitoring rules for a server during installation, then you may execute WLhc.ksh to monitor a specific server, immediately after a successful installation.

Details of how to execute the WLhc korn shell scripts are given below:

WLhc.ksh: This is the main script in WLhc used for monitoring (health-checking) WebLogic Servers. This script can be used in **interactive** mode for a real-time snapshot of monitored parameter values or in **non-interactive** mode (via cron) for continual monitoring of configured parameters. Executing WLhc.ksh in non-interactive mode will switch ON data file and log file generation. i.e. monitored parameter values will be recorded in csv format in data files and alarms for pre-configure traps will be logged in log files.

```
USAGE : ksh WLhc.ksh -s <server> [-i] [-n]
```

where, -s => The name of the WebLogic server you wish to monitor (only 1 server allowed)

-i => script run in interactive mode [default]. This argument is optional.

-n => script run in non-interactive mode. This argument is optional.

NOTE:

- (1) Always test the WLhc.ksh script interactively before setting it up for non-interactive monitoring via cron or whatever scheduling software you use.
- (2) Ensure you measure the execution time for the WLhc.ksh script when executed interactively, so that you can schedule non-interactive execution with appropriate execution frequency. For example, on a domain with around 500 MBeans per server, I could set up non-interactive execution of WLhc.ksh to run every minute.

Auxiliary Scripts:

confBuilder.ksh: This script does the following:

- (1) Initializes a server's config file: This refers to ensuring that the "WLhc config" section containing details such as server name, connection URL, username, password, etc. is built.
- (2) Runtime MBean Listing: This refers to querying the specified server to retrieve a list of all the server's runtime MBeans.
- (3) Rule Creation: This refers to creating the monitoring rules for a specified server. Existing rules will not be overwritten. Rules are created based on the template available in conf/confBuilder.temp.

```
USAGE: ksh confBuilder.ksh -s <server> -t <server type> [-b]
```

where,

<server> = The name of the WebLogic server whose MBeans you wish to list.

<server type> = ADMIN or MANAGED

-b = Option to build the server's monitoring rules (controlled by confBuilder.temp). Optional and default is NO build.

If you have skipped details of a Weblogic server in WLhc.cfg during installation, you can add required details to WLhc.cfg (conf/WLhc.cfg) and use confBuilder.ksh to build the server's configuration.

MBeanAttributes.ksh:

NOTE: This script must be run only after the server for which it is executed already has its configuration file initialized.

This script displays the list of MBean attributes and their current values for a specified MBean on a specified WebLogic server. You will require executing this script whenever you wish to determine all the attributes for a specific MBean to help you configure Server rules.

```
USAGE: ksh MBeanAttributes.ksh <server> '<MBean Object Name>'
```

where,

<server> = The name of the WebLogic server whose MBeans you wish to list.

<MBean Object Name> = Object Name of the MBean whose attributes you wish to list. Select this object name from the Runtime dat files in the conf directory.

Use single quotes (' ') to enclose this parameter.

updateTraps.ksh: This script enables or disables traps for a specified server. It also provides you an option to define the time for which traps may be disabled without generating warnings. For example, if you have a planned release deployment, you may choose to disable traps for the duration of the release. You can do this by using the '-i' option with updateTraps.ksh and specify the interval in minutes. After this interval has elapsed, WLhc will log warnings in the relevant log file to help you remember that the traps need to be re-enabled.

USAGE : `ksh ./updateTraps.ksh -d|-e [-i <interval>] [-a]`

where, -d => disable traps

-e => enable traps

-i => Interval (in minutes) between checks for disabled traps.

-a => disable or enable traps for ALL config files in this WLhc

installation

6. Configuring WLhc

Each server being monitored by WLhc must have its own configuration file (<server>.cfg in the conf directory). During installation, sample configuration files are generated for each server, based on a template (conf/confBuilder.temp). These sample files contain rules for monitoring frequently monitored parameters. This section will guide you on extending the sample configuration files to customize monitoring to meet your requirements.

The configuration file consists of 3 sections:

1. **WLhc config:** This section contains key parameters which control how WLhc connects to the server being monitored, trap behaviour and housekeeping.

Given below are details on WLhc config variables:

SL#	WLhc CONFIG VARIABLE	DESCRIPTION
1	INSTALL_DIR	The absolute path to the directory in which you want to install the WLhc software. This directory must exist.
2	JAVA_DIR	The absolute path to the directory containing the "java" binary used to start the WebLogic servers
3	WEBLOGIC_JAR	The absolute path for the weblogic.jar used by the WebLogic servers
4	CONN_URL	The connection URL for the WebLogic Admin Server in the format <IP>:<Port> or <Domain Name>:<Port> (no angular brackets)
5	USERNAME	The username for the WebLogic user used to query MBeans (for security, a user with only Monitor role is recommended)
6	PASSWORD	The password for the WebLogic user specified above
7	SERVER	The name of the WebLogic Server being monitored
8	TOTAL_ALLOWABLE_CONCURRENCY	The maximum number of WLSHELL processes allowed to run concurrently. The default value is 10. This parameter protects the host in case too many java processes are

		stuck and hanging around
9	SERVER_ALLOWABLE_CONCURRENCY	The maximum number of instances of WLhc allowed to connect to a WebLogic Server. The default value is 3. This parameter protects the WebLogic Server being monitored, in case several users attempt to use WLhc concurrently.
10	ENABLE_TRAPS	Switch to enable or disable traps. This parameter can be changed via the updateTraps.ksh script.
11	DISABLEDTRAPCHECK_INTERVAL	Interval (in minutes) between checks for disabled traps
12	DATA_RETENTION	The retention period, in days, for data files. Default is 30.
13	LOG_RETENTION	The retention period, in days, for log files. Default is 30.

2. Server rules:

- Every rule must be contained within a single line.
- ~ is the field separator for a rule.
- The line containing a rule must not terminate with a ~.

SYNTAX:

SL#~MBean OBJECT NAME~MBean ATTRIBUTE~FRIENDLY NAME~TRAP
OPERATOR~(LOWER) THRESHOLD~UPPER THRESHOLD

where,

SL# (mandatory) => An alphanumeric identifier which must always begin with 'S', followed by a number, the number being incremented by 1 for the next rule. For example, if 3 server rules are configured, then the SL# can be S1, S2 and S3.

MBean OBJECT NAME (mandatory) => The entire MBean Object Name for the MBean representing the parameter you want to monitor. You can select the appropriate MBean Name for a server by browsing the list of names in the

Runtime MBean data files located in the conf directory. The chosen MBean Name must be copied and pasted in the server rule as is.

MBEAN ATTRIBUTE (mandatory) => The MBean attribute (for a specific Runtime MBean) which you wish to monitor. You can select which attribute you want to monitor by executing the MBeanAttributes.ksh script with appropriate arguments. When executed, you will see a list of attributes and their current values within parentheses. The chosen attribute (for monitoring) must be copied and pasted in the server rule as is.

FRIENDLY NAME (mandatory) => Some MBean attributes do not have friendly or short names. So, you can choose a name of your liking for the parameter you wish to monitor or you can use the same name as the MBean attribute.

TRAP OPERATOR (optional) => This is the operator used to configure a trap on a monitored parameter.

The following operators are valid:

TRAP OPERATOR	DESCRIPTION	OPERAND TYPE
-lt	less than	numeric
-le	less than or equal to	numeric
-gt	greater than	numeric
-ge	greater than or equal to	numeric
-eq	equal to	numeric
-ne	not equal to	numeric
=	equal to	string
!=	not equal to	string
between	between lower and upper thresholds	numeric

The trap operator is optional. If the trap operator is not specified, then the parameter or MBean attribute specified in that rule will not be trapped and the lower and upper thresholds, if specified, will be ignored. However, its values will still be obtained and recorded.

(LOWER) THRESHOLD (only required if TRAP OPERATOR is specified) => This is the threshold value for the monitored parameter if configuring a trap on a single operand or the lower threshold value for the monitored parameter if configuring a trap when the trap operator 'between' is used.

UPPER THRESHOLD (only required if TRAP OPERATOR is 'between') => This is the upper threshold value for the monitored parameter if configuring a trap when the trap operator 'between' is used.

3. JMS Destination rules:

- Every rule must be contained within a single line.
- ~ is the field separator for a rule.
- The line containing a rule must not terminate with a ~.
- There is no requirement to specify which parameters need to be monitored. **By default, WLhc will monitor consumers, 'messages current' and 'messages pending' for all JMS destinations specified in rules.** Values for 'messages received' are also recorded, but not monitored.
- The threshold specified is applicable to both 'messages current' and 'messages pending'. You cannot set different thresholds for 'messages current' and 'messages pending'.

SYNTAX:

SL#~MBean OBJECT NAME~FRIENDLY NAME~TRAP OPERATOR~(LOWER)
THRESHOLD~UPPER THRESHOLD

where,

SL# (mandatory) => An alphanumeric identifier which must always begin with 'Q', followed by a number, the number being incremented by 1 for the next rule. For example, if 3 server rules are configured, then the SL# can be Q1, Q2 and Q3.

MBean OBJECT NAME (mandatory) => The entire MBean Object Name for the MBean representing the JMS Destination you want to monitor. You can select the appropriate MBean Name for a JMS Destination by browsing the list of names in the Runtime MBean data files located in the conf directory. The chosen MBean Name must be copied and pasted in the JMS Destination rule as is.

FRIENDLY NAME (mandatory) => Some MBean attributes do not have friendly or short names. So, you can choose a name of your liking for the JMS Destination you wish to monitor.

TRAP OPERATOR (optional) => This is the operator used to configure a trap on a monitored JMS Destination.

The following operators are valid:

TRAP OPERATOR	DESCRIPTION	OPERAND TYPE
-lt	less than	numeric
-le	less than or equal to	numeric
-gt	greater than	numeric
-ge	greater than or equal to	numeric
-eq	equal to	numeric
-ne	not equal to	numeric
between	between lower and upper thresholds	numeric

The trap operator is optional. If the trap operator is not specified, then the JMS Destination specified in that rule will not be trapped and the lower and upper thresholds, if specified, will be ignored. However, JMS Destination attributes (messages current, messages pending, messages received and consumers) will still be obtained and recorded.

(LOWER) THRESHOLD (only required if TRAP OPERATOR is specified) => This is the threshold value for the 'messages current' and 'messages pending' attributes if configuring a trap on a single operand or the lower threshold value for the

'messages current' and 'messages pending' attributes if configuring a trap when the trap operator 'between' is used.

UPPER THRESHOLD (only required if TRAP OPERATOR is 'between') => This is the upper threshold value for the 'messages current' and 'messages pending' attributes if configuring a trap when the trap operator 'between' is used.

NOTE: If you wish to disable a trap for specific parameters, then just do NOT specify the trap operand and thresholds and end the configuration rule with the Friendly Name (not ~). You CANNOT use # at the beginning of a configuration rule to disable that parameter's trap.

7. Running WLhc in Batch mode

In order to periodically monitor WebLogic Servers, you must schedule WLhc to run in non-interactive mode. For example, if WLhc is installed in /software /WLhc and you wish to monitor 2 servers called wls1 and wls2 every 5 minutes, then you may set up cron jobs as follows (entry for each server on 1 line):

```
0,5,10,15,20,25,30,35,40,45,50,55 * * * * /software/WLhc/bin/WLhc.ksh -s wls1 -n >
/software/WLhc/logs/WLhc_wls1.cron.out 2> /software/WLhc/logs/WLhc_wls1.cron.err
```

```
0,5,10,15,20,25,30,35,40,45,50,55 * * * * /software/WLhc/bin/WLhc.ksh -s wls2 -n >
/software/WLhc/logs/WLhc_wls2.cron.out 2> /software/WLhc/logs/WLhc_wls2.cron.err
```

NOTE: Some cron versions (e.g. vixie cron), then shortcuts like */5 can be used to denote every 5 minutes. If WLhc works fine, the *.cron.out and *.cron.err files should be zero byte in size.

Setting up cron jobs as described above will ensure the following:

- (1) Data on parameters being monitored is appended to csv files and
- (2) Any alarms are logged

8. Extending WLhc

Here are some suggestions for extending the use of or building upon WLhc:

- Feed the generated data files (csv) regularly to a graphing program to obtain illustrative graphs of monitored parameters. This will be ideal for helpdesk staff and Operations Management.
- Use log scanning programs to parse the log files and raise alarms (e.g. via SNMP) when threshold breaches are logged.
- Rewrite WLhc in Ruby or Python to make it portable and platform independent ☺