Eszter SB Application Engine

Peter Verhas

Short Contents

Table of Contents

1 Introduction

This is the Users Guide for the Eszter SB Engine program. This program is a standalone web server that is used to execute ScriptBasic programs.

The Eszter SB Engine is a high performance, special-purpose web server that was developed to execute ScriptBasic scripts fast and efficiently.

The ScriptBasic interpreter can be used to run CGI programs utilizing any widely available general-purpose web servers. That solution is fine for most of the cases. In some cases however a high performance special purpose web server is needed.

Eszter SB Engine is a web server itself so there is no need to have another installed on the machine. You will see later that it is advised to use it together with Apache. Whenever an URL is requested it executes the ScriptBasic program inside the web server process as a separate thread. This kind of execution is extremely efficient and fast.

The single process-multi-thread implementation creates a good possibility to store data in memory between the consecutive hits. This data sharing is supported by the Eszter SB Engine and is delivered by the external module MT.

2 Using The Eszter SB Engine

To use the Eszter SB Engine there are two scenarios. You can use it as a stand alone web server or you can use it behind a general-purpose web server like the Apache web server.

2.1 CD publishing

As a standalone server the engine is typically used on small installation, like a multimedia CD published for Windows operating systems. Using the engine you can set up a configuration that automatically starts when the user inserts the CD and the browser accesses the information stored on the CD through the BASIC programs. This lets you store compressed or encrypted data on the CD in case uncompressed data would exceed CD size limit and decompress on the fly or implement search indexes to help the users find the proper information.

[Note: To make available configuration of ScriptBasic on Windows platforms the v1.0build22 changed the search path for configuration file. Now the first choice is file '`scriba.conf`' in the same directory as the binary.]

2.2 Pure client software

For the Windows environment you can develop pure client applications that run using the same technology as web applications. The application written in ScriptBasic runs on the client machine and thus can be used even on a remote laptop travelling cross the country and still accessing full functionality of your application. On Windows2000 you can even install a MySQL server for full processing functionality.

2.3 Web server applications

On UNIX operating system using Apache is very popular. You can use the ScriptBasic command line version to execute ScriptBasic programs as CGI processes. Although this approach is powerful there is another solution using the Eszter SB Engine. You can run the Eszter SB Engine on the same machine as the Apache web server or on a different machine and using the Apache rewrite and proxy modules you can proxy all `.bas` URL requests to the Eszter SB Application Engine. This way you have a full-featured general purpose web server and extremely fast execution of ScriptBasic programs without the need of spawning new processes for each request.

3 Installation Instructions

The Eszter SB Application Engine is installed together with the command line variation of ScriptBasic. In addition to that you may want to edit the configuration file to set the virtual directories and the other options of the engine.

For more information on how to configure the engine read the chapter See ⟨undefined⟩ [Configuration], page ⟨undefined⟩.

Optionally you may want to have the Eszter SB Application Engine to start automatically when the system is started up. To do this there is a sample `sh` script that starts and stops the engine. This is listed in the section See ⟨undefined⟩ [Start/Stop Sh Script], page ⟨undefined⟩. You can place this script in the file '`/etc/init.d/sbhttpd`' as well in '`/etc/rc2.d/K91sbhttpd`' and '`/etc/rc3.d/K91sbhttpd`'. This is the location for my Debian Linux installation. Other Linux or UNIX installations may require different locations for the script.

3.1 Start/Stop Sh Script

This `sh` script can be used to start and to stop the Eszter SB Application Engine. This script can be placed in the directory '`/etc/init.d`' or in the directories `/etc/rcX.d` to have the engine automatically start whenever the Linux box boots.

The script is not Linux specific; you can use it under other UNIX operating systems. The location of the file however may be different.

The source code of the file '`etc-init.d-sbhttpd`' is:

3.2 Configuring Apache for the Eszter SB Application Engine

You can use the Eszter SB Application Engine together with the Apache server. The reason to do this is that the engine is not a general purpose web server, like Apache. You want to have static pages, run Perl cgi scripts, PHP programs. These are not provided by the Eszter SB Application Engine. Therefore you need Apache.

Having Apache on your webserver you have two choices how to run ScriptBasic CGI programs.

You can run ScriptBasic programs as regular CGI programs. The command line version of the interpreter usually installed as '`/usr/bin/scriba`' is capable executing CGI programs. The CGI programs should have a starting line `#! /usr/bin/scriba -c` telling the UNIX operating system how to start ScriptBasic for the program. The option '`-c`' tells ScriptBasic that the program is executed in a CGI environment. If this option is used the error messages are HTML formatted and thus you can read it on the web browser. If you omit this option you will get "Internal Server Error" on the browser and you will be able to read the actual error in the Apache server error log.

Using the plain old CGI execution will execute each BASIC program in a newly created process. This means process starting overhead, all database connections are initiated by the program, and no memory stored session variables are available.

The other approach is to install and configure the Apache server including the modules PROXY and REWRITE. In this case the Eszter SB Application Engine runs the BASIC programs. This allows BASIC programs run in the same process in separate threads thus eliminating the overhead of process creation. This also allows access to memory stored session or application variables and database connection pooling. (Note however that the only database supported currently is MySQL and the external MySQL module does not have database connection pooling capability.)

These are standard modules, which are not compiled and configured with the default Apache installation. However they are needed by many installations and thus the Apache documentation contains examples how to compile and install Apache including these modules.

Once you successfully installed Apache including this module you have to alter the configuration file to tell Apache how to use these modules. To have the modules loaded into Apache you should have the two lines in your configuration file uncommented:

```
LoadModule proxy_module /usr/lib/apache/1.3/mod_proxy.so
LoadModule rewrite_module /usr/lib/apache/1.3/mod_rewrite.so
```

The actual path to the shared object files may be different. After the modules are configured you have to tell Apache that all URLs that have the file extension `.bas` should be proxy to the Eszter SB Application Engine. As an example I include here the lines of the configuration of my Linux station.

```
<VirtualHost 192.168.0.2>
ServerAdmin webmaster.com
DocumentRoot /home/verhas/scriptbasic
ServerName www.scriptbasic
ServerAlias scriptbasic
ErrorLog /var/log/apache/scriptbasic/error.log
TransferLog /var/log/apache/scriptbasic/access.log
ScriptAlias /cgi-bin/ /home/verhas/scribas/
Alias /doc/ /home/verhas/scriptbasic/doc/
RewriteEngine on
RewriteLog /var/log/apache/rewrite.log
RewriteLogLevel 9
RewriteRule .*/redir/(.+)\.bas$ http://127.0.0.1:8080/cgi-bin/$1.bas [P]
</VirtualHost>
```

(This machine is no ever connected to the Internet, so you have no way to hack this Linux based on this information.)

4 Starting the engine

This chapter details how to start the engine on UNIX and on Windows NT.

4.1 Starting on UNIX

To start the engine on UNIX you just type

```
sbhttpd
```

on the command line. This will start the engine in the foreground. This may be needed when you want to debug some CGI scripts. Otherwise you should start the engine in the background saying:

```
sbhttpd -start
```

If you have installed the script listed in section See ⟨undefined⟩ [Start/Stop Sh Script], page ⟨undefined⟩ then the most preferred method is to type

```
/etc/init.d/sbhttpd start
```

to start,

```
/etc/init.d/sbhttpd stop
```

to stop, and

```
/etc/init.d/sbhttpd restart
```

to restart the engine.

In case you experience instability, the Engine crashing then you can start the engine using the command line option

```
sbhttpd -safe
```

This option start the engine in the background, but there is also a process remaining in the foreground and in case the real engine process stops the foreground process starts it again and again. Though we did not experience the engine crashing this method may give some way until the bug is localized, reported and we correct it.

4.2 Starting on Windows NT

Under Windows NT using the engine v10b23p1 and above you have to give the parameter

```
sbhttpd start
```

that informs the engine that the program was started from the command line and not as an installed service. Although you can use this option further following the version v10b30 it is not a must anymore. This version implements some heuristics that can distinguish the service start from the command line start fast.

You can install the executable file as a service under Windows NT invoking the command line

```
sbhttpd install
```

You can remove the installed service under Windows NT invoking the command line

```
sbhttpd remove
```

These options ('`-install`' and '`remove`') are not available under UNIX. To start the service and to alter the service options use the Windows NT Service Control Manager (SCM).

All other command line options that alter the behavior of the web server should be written after the options start or safe. Thus

```
sbhttpd p 88 start  WRONG
```

is incorrect.

4.3 Command line options

All command line options are ignored following the options '`install`' and '`remove`'. If any of the options '`safe`' or '`start`' is specified all other options should follow these options.

```
-p port
```

This option can be used to override the port that the engine is listening to. If this option is present the port number specified in the configuration file is neglected and the port number specified on the command line is used.

For example:

```
sbhttpd start p 88
```

will listen on the port 88 even if the configuration file specifies the port 8080.

```
-h ip address
```

This option can be used to override the ip number that the http server is bound onto. The address has to be specified with IP number (not name) in the `x.x.x.x` format.

If you do not specify any IP address and there is also no ip address configured in the configuration file then the http daemon will listen on all available ip numbers of the machine. There can only be a single ip number specified and the ip stack of the machine should be configured so that one of the network cards handle this ip address.

5 Stopping the engine

To stop the engine you can use three methods:

Delete the pid file of the engine and wait until the engine stops gracefully. Please read the section See ⟨undefined⟩ [pid], page ⟨undefined⟩ on how to stop the server gracefully.

`kill` the smallest pid `sbhttpd` process, which pid should also be listed in the pid file.

`kill -9` the process.

The graceful stop allows the engine to serve the pending HTTP request, but not accepting any more, closing the log files. The engine in the current version does not handle `HUP` or `TERM` signals.

If you have installed the script listed in the section See ⟨undefined⟩ [Start/Stop Sh Script], page ⟨undefined⟩ you can stop the engine saying

```
/etc/init.d/sbhttpd stop
```

This script performs all three methods starting with the graceful and continuing with the more drastic methods in case the previous did not work.

6 Configuration

The engine reads the same configuration file as the command line version of ScriptBasic and several option keys alter its behavior. Here we detail only those keys that are specific to the engine. For further information on the configuration file format and the keys that control general ScriptBasic behavior please read the Users Guide.

All special keys that control the Eszter SB Application Engine are under the key `httpd` in the configuration file. For example the configuration file (text format) fragment:

```
httpd (
  port 8080
  home "C:\\MyProjects\\sb\\source\\examples\\"
  proxyip 0
  ip "127.0.0.1"
)
```

define that the server should listen on the port `8080`, the home directory for the executable BASIC programs is '`\MyProjects\sb\source\examples\`' (that is the directory we use under Windows NT on the development station) and the client ip should not be proxy-ed (explained later).

Note that this configuration sample is not enough to start the Eszter SB Engine. It should be carefully configured otherwise it may stop reporting configuration errors. For fastest result without reading this chapter edit the sample configuration file, compile it and give it a try.

In the following sections we detail each configuration key that Eszter SB Engine uses to control its behavior. All these keys should be under the key `httpd`.

Note that all directory names should be expressed as absolute full path to the given directories or files. The reason for this is that there is no guarantee of the default directory. Even an executed script can alter the current directory using the command `chdir` and the next execution may not find the input file, may not find the pid file and thus will not be able to execute the next script or even stops the engine.

6.1 threads

```
threads 20
```

This key defines how many threads may run at a time in the engine. This means that this many connections can be handled by the Eszter SB Egnine. There is no significant performance penalty to increase this number. The threads are started whenever a connection should be served and are stopped when the serving is finished. The only performance penalty is a small amount of memory used to administer the thread local data array. You should set this value according to the traffic of the site. The value of 10,000 should be large enough. Setting the limit lower will prevent the connections degrading your performance. Excess connection trials will be dropped but the ones that got through will be served.

6.2 listenbacklog

```
listenbacklog 3
```

This key defines how many connections are queued up in the socket listen queue at most. Because the engine starts new threads for the incoming connections there is no need to set this parameter too high. The value 3 should be fine even on a high traffic server. We have experience with a server with 1.5million hits per day having this value 3.

6.3 port

port 80

This key defines the port that the engine is listening.

6.4 ip

ip "127.0.0.1"

This key defines the IP address that the engine is listening. If this value is not configured and no IP address is specified on the command line then the engine will listen on all IP addresses that the machine is configured to serve. Using this key a single IP address can be configured.

The IP number should be defined as a string in the form of `x.x.x.x`

For example defining this key to be `"127.0.0.1"` will mean that the server will only be available from the local machine. This can be a safe installation mode for BASIC web programs that run on a single machine most probably from a CD installed program and serve only the person sitting in front of the PC.

6.5 home

```
home "C:\\MyProjects\\sb\\source\\examples\\"
```

This key defines the home directory. On UNIX as well as on Windows NT you can use forward slashes in the directory names. However using backward slashes you have to duplicate them.

This key defines the home directory where the BASIC programs are when no any virtual directory definition matches the URL. Note that all files are executed as BASIC source code not caring the file name extension.

6.6 proxyip

```
proxyip 0
```

This key should be zero or 1. If the key is 0 the client IP that the BASIC program sees is the real client connecting to the engine. If this key is 1 the client IP is the IP number provided by the proxy server (e.g. Apache proxy module).

Eszter SB Engine can be used as a standalone web server or as an auxiliary application server that happens to communicate using the http protocol. In this second case the original HTTP request arrives to the primary web server (for example the Apache web server) and the request is proxy-ed to the Eszeter SB Engine. In this case the client IP that the engine sees is the IP number of the server where the Apache is running. This is not the real client IP. However some programs use the client IP number. There is a standard way proxies use to report the original client IP address. This is the http header field `X-Forwarded-For`.

If this key is true then the environment is altered before the BASIC programs are executed to hold the IP number reported in this header field instead of the IP number from the socket.

6.7 user

```
user "www-data"
```

This key specifies the user that the Eszter SB Application Engine uses to execute. This key has effect only under UNIX and is ignored under Windows NT.

When the Eszter SB Application Engine has performed the initialization tasks it calls the system function `setuid` to change the effective user. After this point all BASIC programs are going to be executed in the security context of the named user.

The name of the user for this operation has to be specified in this configuration key.

If the configuration key is missing the Eszter SB Application Engine will be executing in the context of the original user.

It is recommended that you specify this user and install the Eszter SB Application Engine owned by `root` and with the setuid bit set. This way you can secure the panic log file, which is opened as root. The other log files, however can not be secured this way as they are closed and reopened time to time by the engine.

To ensure that the engine really runs in the context of the specified user I performed the following test:

```
# chown root:root /home/verhas/scribas/echo.bas
# chmod 700 /home/verhas/scribas/echo.bas
# touch /home/verhas/scribas/echo.bas
# sbhttpd

# cat /var/log/scriba/err.log
2001.09.09 08:38:09 /home/verhas/scribas/echo.bas(0): error &H42:The file can not b
# chown www-data:www-data /home/verhas/scribas/echo.bas
# sbhttpd
```

After starting the engine (from hand `sbhttpd`) I used my browser to invoke the script '`echo.bas`'. I simply pressed control C to stop it. The command `touch` is necessary otherwise the already executed script is not read, but executed from the compiled version from the cache directory.

6.8 pid

```
pid (
    file "\\MyProjects\\sb\\httpdlog\\pid.txt"
    delay 10
    wait (
      period 10
      length 1
      )
    )
```

This key should have sub-keys. The key `pid.file` should define the pid file that the program writes. When the program starts it creates this file and writes the pid of the process into this file.

This file can be used to stop the engine. There are two ways to stop the engine. One is to use the pid and terminate the process using the command kill on UNIX or the task manager under Windows NT. This is the brute force solution and may result in data loss or data inconsistency if BASIC programs are still running.

The graceful way is to delete the created pid file.

If the pid file was created the engine checks the existence of the file every delay seconds (10 seconds in the example above). The code tries to open the file for reading. If the file exists the engine closes it and sleeps for another delay seconds.

If the file was deleted or its security attributes were set so that the engine can not read it the engine sets its internal state from running to stopping. This will inform the main thread to neglect all incoming connections, wait for the already running threads to stop and then to terminate the process.

The main thread waits period times sleeping length seconds between the periods for the threads to stop. If all threads stop the main thread does not wait longer but terminates the process. If some the thread is still executing the main thread kills those still running threads exiting the process.

It is also a graceful termination of the web server on Windows NT to use the Service Control Manager to stop the service in case the server was installed and started as a service.

6.9 vdirs

```
vdirs (
    dir "/user/:e:\\MyProjects\\sb\\source\\examples\\user\\"
    dir "/cgi-bin/:e:\\MyProjects\\sb\\source\\examples\\"
    )
```

This key with its subkeys should define the virtual directories. The definitions are searched from the first towards the last and whenever a definition matches the actual URL the search finishes.

The string should contain the virtual directory path including the starting and closing slash a colon separating the virtual directory and the real directory and finally the real directory.

6.10 run

```
run (
  start   "e:\\MyProjects\\sb\\source\\examples\\runstart.bas"
  restart "e:\\MyProjects\\sb\\source\\examples\\runrestart.bas"
  )
```

This key with the subkeys define the programs that has to be started by the Eszter SB Engine before starting listening on the defined port for incoming HTTP connections. All programs that follow the key start are started only once when the Engine is started. All programs that follow the keyword restart are started the same way as the programs with the keyword start, but are also restarted whenever they stop so long as long the engine is running.

These programs run asynchronously in separate threads. A typical use of such a program is to periodically check old sessions in the module MT and delete the sessions that were not deleted by the CGI programs.

6.11 client

```
client (
    allowed "127.0.0.1/255.255.255.255"
    allowed "16.94.58.4/0.0.0.0"

    denied "127.0.0.1/0.0.0.0"
    denied "16.192.68.5/255.255.0.0"
    )
```

You can define the client IPs that are allowed and denied to connect to the engine. This is the real client that is connecting to the listener on the socket and trying to get access via HTTP and not the client reported by any proxy header field. You can use this configuration option to allow only the web server machine to connect to the engine via the proxy module and disallow all external connections that may try to connect from external machines. You can also use this option to configure a client application not to be accessed by external computer.

A connection is accepted by the engine if there is at least one allowed configuration line that allows the connection and there is no any denied configuration line that denies the connections.

The string following the keywords allowed or denied should contains the IP number and the MASK separated by a slash. When the client IP is calculated first it is bitwise AND-ed by the MASK and compared to the IP number. Thus

```
allowed "127.0.0.1/255.255.255.255"
```

allows only connections from the `localhost` and

```
denied "127.0.0.1/0.0.0.0"
```

denies access for no-one (in other word this line is waste of CPU and text file, but is good as an example). The line

```
denied "16.192.68.5/255.255.0.0"
```

disallows all requests that come from an IP number 16.192.*.* that is some internal sub-network of the once existed firm: Digital Equipment Corporation.

6.12 errmsgdest

```
errmsgdest 3
```

This key defines where the error messages are sent. By default error messages are sent to nowhere. This key may have four different values:

0 send error messages to nowhere

1 send the error messages to the client browser

2 send the error messages to the error log file (default)

3 send the error messages both to the client browser and the error log file

Switching off all error messages may harden to find out what goes wrong, therefore it is not advisable to set this configuration value zero. The error log file is a good place for error messages. Sending error messages to the client browser may be convenient during development but may present security risk on a production server.

6.13 nolog

```
nolog 0
```

This key should have the value zero or one. For security reasons the engine does not start if it can not open its log files. The only exception is if you configure it explicitly saying not to try to open the log files using the line:

```
nolog 1
```

6.14 log

```
log (
    panic ( file "\\MyProjects\\sb\\httpdlog\\panic.log" span 0)
    app   ( file "\\MyProjects\\sb\\httpdlog\\app.log" span 0)
    err   ( file "\\MyProjects\\sb\\httpdlog\\err.log" span 0)
    hit   ( file "\\MyProjects\\sb\\httpdlog\\hit.log" span 0)
    stat  ( file "\\MyProjects\\sb\\httpdlog\\stat.log" span 0)
    )
```

These keys define the different log files. These are:

hit contains detailed information about each hit. This is the most frequently written log file.

stat contains information about the statistics of the internal parameters of the engine. This helps the system managers to set certain parameters of the engine optimally. This log is not used in the current version.

app contains information about the actions that the engine performs.

err contains information about errors.

panic contains information about severe errors that prevent the engine to run.

If the engine does not run check the panic log and search for the file '`panic.txt`'. This file is created in case even the panic log file can not be opened.

The sub-keys panic, hit, stat, app and err also contain sub-keys: file and optionally span. The sub-key file specifies the name of the file. This may contain `sprintf` formatting `%d` format strings, at most four. These will get the four-digit year, month, day and the hour (gm time). Note that the current version of the engine does not create the directory and fails writing the statistics file if the directory does not exists.

To create the file name is a time consuming process and therefore the engine needs configuration help to prevent file name recalculation for each log item. Therefore the value of the key span should specify how often the log file is rotated in number of seconds. If this is zero the log file is treated as non rotating and the file name is recalculated at nonspecific times.

6.15 msg404

```
msg404 """
<HTML>
<HEAD>
<TITLE>Error 404 page not found</TITLE>
</HEAD>
<BODY>
<FONT FACE="Verdana" SIZE="2">
<H1>Page not found</H1>
We regretfully inform you that the page you have requested can not be found on this
<p>
In case you are sure that this is a server configuration error, please contact
<FONT SIZE="3"><TT>root</TT></FONT>
</FONT>
</BODY>
</HTML>
 """
```

This key may specify alternative 404 error page HTML text.

6.16 code404

```
code404 "200 OK"
```

This key may define alternative 404 error page state. The default is 404.

6.17 run404

```
run404 "\\MyProjects\\sb\\source\\examples\\run404.bas"
```

This key may define alternative program in case of 404 error. If even this program is not loadable the msg404 text is used if defined.