

Administrator's Guide

WebtoB 5 Fix#4

TMAXSOFT

Copyright

Copyright 2023. TmaxSoft Co., Ltd. All Rights Reserved.

Restricted Rights Legend

All TmaxSoft Software (Tmax WebtoB®) and documents are protected by copyright laws and international convention. TmaxSoft software and documents are made available under the terms of the TmaxSoft License Agreement and this document may only be distributed or copied in accordance with the terms of this agreement. No part of this document may be transmitted, copied, deployed, or reproduced in any form or by any means, electronic, mechanical, or optical, without the prior written consent of TmaxSoft Co., Ltd. Nothing in this software document and agreement constitutes a transfer of intellectual property rights regardless of whether or not such rights are registered) or any rights to TmaxSoft trademarks, logos, or any other brand features.

This document is for information purposes only. The company assumes no direct or indirect responsibilities for the contents of this document, and does not guarantee that the information contained in this document satisfies certain legal or commercial conditions. The information contained in this document is subject to change without prior notice due to product upgrades or updates. The company assumes no liability for any errors in this document.

Trademarks

Tmax WebtoB® is a registered trademark of TmaxSoft Co., Ltd. Other products, titles or services may be registered trademarks of their respective companies.

Open Source Software Notice

Some modules or files of this product are subject to the terms of the following licenses. : OpenSSL, ZLIB, PCRE, APACHE1.0, APACHE1.1, APACHE2.0, JSON-C, BSD, RSA, PHP, Paul Hsieh's hash

Detailed Information related to the license can be found in the following directory:

`${INSTALL_PATH}/lib/licenses`

Document History

Product Version	Guide Version	Date	Remarks
WebtoB 5 Fix#4	2.1.5	2023-12-21	-

Contents

1. Introduction	1
1.1. Structure and Operation	1
1.1.1. Comparison with Conventional Web Servers	1
1.1.2. WebtoB Processes and Threads	2
1.2. Features	3
1.2.1. Caching	3
1.2.2. WBAPI	3
1.2.3. Calling TP-Monitor Tmax Service	4
1.2.4. Extension Management	4
1.2.5. Logging	4
1.3. Management Tool	4
1.4. Environment Variables	4
1.5. Directory Structure	5
1.6. License Specific Functions	6
2. Quick Start	7
3. Configuration	10
3.1. Overview	10
3.1.1. Configuration File Structure	10
3.1.2. Configuration File Compilation	13
3.1.3. Configuration Item Description Rules	14
3.2. DOMAIN Section	14
3.2.1. Configuration Items	15
3.2.2. Example	16
3.3. NODE Section	17
3.3.1. Configuration Items	17
3.3.2. Example	51
3.4. VHOST Section	51
3.4.1. Configuration Items	52
3.4.2. Example	62
3.5. HTH_THREAD Section	62
3.5.1. Configuration Items	62
3.5.2. Example	65
3.6. SVRGROUP Section	65
3.6.1. Configuration Items	65
3.6.2. Example	71
3.7. SERVER Section	71
3.7.1. Configuration Items	71
3.7.2. Example	82

3.8. SERVICE Section	83
3.8.1. Configuration Items	84
3.8.2. Example	85
3.9. DIRECTORY Section	85
3.9.1. Configuration Items	85
3.9.2. Example	88
3.10. URI Section	88
3.10.1. Configuration Items	89
3.10.2. Example	94
3.11. ALIAS Section	95
3.11.1. Configuration Items	95
3.11.2. Example	96
3.12. DIRINDEX Section	96
3.12.1. Configuration Items	96
3.12.2. Example	99
3.13. LOGGING Section	99
3.13.1. Configuration Items	99
3.13.2. Example	104
3.14. ACCESS Section	104
3.14.1. Configuration Items	104
3.14.2. Example	107
3.15. AUTHENT Section	107
3.15.1. Configuration Items	107
3.15.2. Example	108
3.16. EXT Section	109
3.16.1. Configuration Items	109
3.16.2. MIME-Type	113
3.16.3. Example	115
3.17. SSL Section	115
3.17.1. Configuration Items	115
3.17.2. CA Command	122
3.17.3. Example	123
3.18. PROXY_SSL Section	123
3.18.1. Configuration Items	123
3.18.2. Example	127
3.19. ERRORDOCUMENT Section	127
3.19.1. Configuration Items	127
3.19.2. Example	128
3.20. EXPIRES Section	128
3.20.1. Configuration Items	129
3.20.2. Example	131

3.21. TCPGW Section	131
3.21.1. Configuration Items	131
3.21.2. Example	135
3.22. REVERSE_PROXY_GROUP Section	135
3.22.1. Configuration Items	135
3.22.2. Example	141
3.23. REVERSE_PROXY Section	142
3.23.1. Configuration Items	142
3.23.2. Example	152
3.24. LOGLEVEL Section	152
3.24.1. Configuration Items	152
3.24.2. Example	154
3.25. HEADERS Section	155
3.25.1. Configuration Items	155
3.25.2. Example	157
3.26. PRECEDING_COMMAND Section	157
3.26.1. Configuration Items	157
3.26.2. Example	159
3.27. FILTER Section	159
3.27.1. Configuration Items	159
3.27.2. Example	160
3.28. USERLOGFORMAT Section	160
3.28.1. Configuration Items	160
3.28.2. Example	160
3.29. DISK_CACHE Section	161
3.29.1. Configuration Items	161
3.29.2. Example	163
3.30. LOG_HANDLER Section	163
3.30.1. Configuration Items	163
3.30.2. Example	165
3.31. IPGROUP Section	165
3.31.1. Configuration Items	165
3.31.2. Example	166
4. Advanced Configuration	167
4.1. Dynamic Content	167
4.1.1. CGI	167
4.1.2. SSI	168
4.1.3. PHP	170
4.2. Virtual Hosting	171
4.2.1. Virtual Host Structure	172
4.2.2. Mass Virtual Host	173

4.3. Log	174
4.3.1. Log File	174
4.3.2. Log Format	175
4.3.3. Common Log Format	175
4.3.4. Configuration	176
4.4. Integration with Tmax	178
4.4.1. Tmax Integration Configuration	179
4.4.2. Usage Example	180
4.5. Integration with JEUS	180
4.5.1. WebtoB and JEUS Integration	180
4.5.2. JEUS 6 Integration Configuration (Standard+)	180
4.5.3. Using Embedded Servlet Engine for Enterprise (JEUS 7)	184
4.6. Connecting to Other Web Application Servers	187
4.6.1. Using Reverse Proxy	187
4.6.2. Connecting to Multiple Web Application Servers (Enterprise+)	187
4.7. URLRewrite	192
4.7.1. WebtoB Configuration	193
4.7.2. URLRewriteConfig File Configuration	193
4.7.3. RewriteCond	193
4.7.4. RewriteRule	196
4.7.5. Examples	199
4.8. Filter Module Development (Enterprise+)	202
5. Starting Up and Shutting Down	207
5.1. Starting WebtoB	207
5.1.1. Before Starting WebtoB	207
5.1.2. wsboot	207
5.2. Shutting Down WebtoB	209
5.2.1. wsdwn	209
6. WebtoB Administration	211
6.1. wsadmin Console Management Program	211
6.1.1. Environment Configuration	213
6.1.2. Active State Information	215
6.1.3. Suspend and Resume Server Processes	224
6.1.4. Queue Purge	225
6.1.5. Dynamic Configuration Change	226
6.1.6. Client Disconnection	226
6.1.7. Others	227
6.2. wsmon Console Monitoring Program	233
6.3. Commands	234
6.3.1. wscfl	235
6.3.2. wsuncfl	236

6.3.3. wsgst	236
6.3.4. wsracd	238
6.3.5. wsmkppd	239
7. WebtoB Tuning	241
7.1. HTH and HTH_THREAD Configuration	241
7.1.1. HTH	241
7.1.2. HTH Thread	242
7.1.3. Example	242
7.2. Server Process Configuration	243
7.2.1. Example	243
7.3. Tuning Configuration for BRUN	245
7.3.1. HttpOutBufSize and FlowControl	246
7.3.2. How to Reduce BRUNs	246
8. WebtoB Security	248
8.1. Authentication Method	248
8.2. SSL	248
8.2.1. SSL v3.0	248
8.2.2. SSL vs. SHTTP	249
8.2.3. SSL Encryption	249
8.2.4. Ciphers	249
8.3. Certificate Service	251
8.3.1. Certificate Authority	251
8.3.2. Certificate	252
8.3.3. Certificate Issuing Policy	253
8.3.4. Receiving a Server Certificate from Verisign	254
8.4. Using Authentication and SSL	254
8.4.1. Authentication	254
8.4.2. Configuring SSL	257
9. WBAPI	259
9.1. Overview	259
9.2. Concept	259
9.3. Environment Configuration	261
9.4. Service Table Creation	262
9.5. Program Compilation	263
9.6. Starting and Using WBAPI	264
9.7. WBAPI Types	264
9.7.1. INIT/DONE API	264
9.7.2. ALLOC API	265
9.7.3. GET API	265
9.7.4. PUT/SET API	266
9.7.5. SEND API	267

9.7.6. COOKIE API	267
9.7.7. SESSION API	268
9.7.8. ETC API	269
9.8. CGI Conversion Using WBAPI	269
9.8.1. CGI Program	269
9.8.2. WBAPI Program	272
Appendix A: Environment Configuration Example	274
A.1. Basic Configuration	274
A.2. WebtoB and JEUS Integration Configuration	275
A.2.1. JEUS 6 Integration Configuration	276
A.2.2. Embedded Servlet Engine Integration Configuration (JEUS 8)	277
Appendix B: CGI Application Example	280
B.1. Overview	280
B.2. Implementing BBS in C	280
B.2.1. Environment File	280
B.2.2. HTML Page	281
B.2.3. CGI Source Code	282
B.2.4. Configuration and Checking the Results	283
B.3. Implementing BBS in Pearl	284
B.3.1. Environment File	285
B.3.2. Perl Script	285
B.3.3. Configuration and Checking the Results	293
Appendix C: Integration with Tmax	296
C.1. Client Program for Integrating with Tmax	296
C.2. Compiling Client Program	299
C.3. Integrating with Tmax	300

1. Introduction

This chapter describes the basic structure, fundamentals, key features, admin tools, and environment variables of WebtoB.

1.1. Structure and Operation

WebtoB is structurally different from conventional web servers. Conventional web servers, such as Apache, have NCSA's httpd structure. The httpd structure was designed to support only a moderate number of clients. The httpd structure lacks the flexibility to support a large growth in the number of clients.

Handling surge in users has been the top priority in designing WebtoB.

Conventional web servers generate a process for each incoming request from a client. As the number of clients increases, the number of server processes or threads must also increase. The server can quickly overload from the number of clients.

It is normal that surge in users causes server overload. Traditional approaches to address increased users depend on additional memory and CPUs. Rather than purchasing additional hardware to handle this growth, a better and more cost effective solution is required. The server must be flexible enough to support the increase in clients without interrupting its service in order to survive in the ever growing Web environment.

For WebtoB, the first priority in design was providing scalability to support growth in the number of clients. Comparing the structure of conventional web servers with WebtoB, one can understand why WebtoB is the solution for solving scalability problems.

1.1.1. Comparison with Conventional Web Servers

The structure of Apache was inherited from httpd. httpd is the oldest web server and has a Process per Request type structure. This mechanism generates a process whenever a user request is received. As the number of users increases, the number of processes also increases. Once the number of users grows beyond the server's capability, the server overloads. The server overloads because of the limited number of processes that are available in each system and the increase in memory usage.

These shortcomings led to the development of commercial web servers with scalability for supporting the rapidly expanding web environment. The most commonly used technology is Multi-Threading. The Multi-Threading technology uses a thread, which has far less overhead than a process, to service a user request. Threading shows far superior performance than a process. Moreover, multiple threads can be allocated in a process, which leads to less overhead and requires less memory than process-based structures.

Nonetheless, multi-threading has problems of its own. Even though multi-threading is the superior technology, competing threads often result in deadlocks, affecting the stability of service. If a deadlock occurs in a thread structure, a thread cannot continue its service and may crash the entire

system. This is a critical problem for a web server where performance and stability are vital.

The two previously described structures each have their strengths and weaknesses. They are polar opposites in relation to performance and stability. However, both performance and stability is essential to a web server because they directly affect the quality of service provided to the end user.

TmaxSoft has done extensive research in web server structures to satisfy both the performance and stability requirements when developing WebtoB. WebtoB employs the process-based structure for stability and the thread-based structure for performance. To improve performance, each process is designated to a specific task instead of creating independent processes, and task-specific threads are used for some specific processes. Each process is also provided with a secondary process with the focus on stability.

1.1.2. WebtoB Processes and Threads

The following are descriptions of WebtoB's major processes.

- **WSM(WebtoB System Manager)**

WSM is a WebtoB system operation process that manages operational information and all server processes such as HTL and HTH. WSM is the first process loaded onto memory upon startup of WebtoB, and the last process terminated upon shutdown.

- **HTL(HTTP Listener)**

HTL is a listener process that manages connections between a client and WebtoB. When a client first connects to WebtoB, a connection is created between HTL and the client. For a service request, an internal connection with HTH processes the service.

- **HTH(HTTP Handler)**

Also known as a client handler. It is a mediator between the client and the server's task-processing process. HTH communicates with the server process to control data flow. It takes a service request from the client and processes the request. Once the result is ready, HTH retrieves the result and relays it to the client. HTH uses a high performance thread model that minimizes locking. The role of each thread is as follows:

Thread Type	Description
ACCESSLOG	Only one ACCESSLOG thread per HTH. Records accesslog after processing a HTTP(S) request.
WORKER	Processes HTML requests, SSL, Compression, etc.
SENDFILE	Sends a blocking HTML request to the client.

- **PHPS(PHP Server)**

PHP server process for PHP requests.

- **CGIS(CGI Server)**

CGI server process for CGI requests.

- **SSIS(SSSI Server)**

SSI server process for SSI requests.

1.2. Features

The features of WebtoB are as follows.

1.2.1. Caching

Caching is a feature available in many web servers. Most web servers support disk caching, which retrieves data from external machines and saves the data to the local disk. When clients request this data, the data can be accessed from the local disk instead of the external machine. Known as Proxy, other web servers have this feature and it effectively boosts the performance of the web server.

WebtoB uses caching in a more effective manner. When user sends a request, WebtoB selects resources that are frequently accessed and allocates them in the memory.

Currently, many web service providers use high-end servers with multiple CPUs and gigabytes of memory. Since processing bottlenecks generally come from the CPU, memory is not fully utilized. In fact, only 70-80% of memory is utilized depending on the CPU's processing power.

Memory Caching in WebtoB improves performance by utilizing the unused vacant memory to cache frequently used resources. Because the data transfer rate of memory is 40-60% faster than disk, memory caching can result in significant performance enhancement. Memory cache sizes can be adjusted based on available free memory.

In addition, research results show that most user requests target specific sets of resources. Thus, even small memory caching can result in a significant performance boost.

1.2.2. WBAPI

WBAPI is a set of multi-purpose internal functions provided by WebtoB

WBAPI can convert existing CGI applications. Although CGI application can be easily developed, it is architecturally inefficient. Because of performance issues, CGI application cannot service a large number of users. CGI application based sites experiencing a flood of user requests have problems with service continuity. The CGI application can be redeveloped using a different technology or the hardware can be upgraded to deal with continuity issues.

WebtoB provides a revolutionary WBAPI function that can convert a CGI application into a WebtoB service. The conversion improves CGI applications by providing performance levels similar to Servlets and other script languages. WBAPI can also be used inside WebtoB to call the TP-Monitor's Service Routine.

1.2.3. Calling TP-Monitor Tmax Service

When Tmax Service is called by a browser in WebtoB, a request must be in a specific format. This format is the same as a CGI request. However, WebtoB is required in order to configure the request to be recognized as a Tmax service.

For example, a Tmax Service is defined in the "/tmax" directory under WebtoB. Inside that directory, there is a service named service1. service1 can be requested from the browser as follows:

```
http://www.tmax.co.kr/tmax/service1
```

1.2.4. Extension Management

Most web servers define a default set of standardized Mime Types. Other data or applications must also be defined in the web server for proper transmission. If a user wants to define a new data type, a corresponding MIME Type must be also defined. WebtoB provides a manager to help define these data types. Also in the manager, data extensions can be modified and new extensions can be defined.

1.2.5. Logging

Log files can have generic or customized formats.

1.3. Management Tool

WebtoB provides management tools to manage engine and server processes as follows:

Management Tool	Description
wsadmin	Manages the entire WebtoB system. It provides system information and performs administrative tasks.
wscfl	Compiles text-based configuration files to binary format.
wsuncfl	Converts compiled binary format files to text-based configuration files.
wsmkpw	Generates an authentication file consisting of encrypted user and password information.
wsmkppd	Creates a file that contains the passphrase password used to specify "file:<passphrase file path>" in the *SSL.PassPhraseDialog configuration.

1.4. Environment Variables

The following table describes the environment variables provided by WebtoB.

Environment Variable	Description
WEBTOBDIR	Configures the information of the WebtoB installation directory. (Required)
WEBTOB_LICENSE	Configured when license file name is changed. (Default value: license.dat)
WEBTOB_RAC_PORT	Must be configured when the port number used by wsrcd is changed. (Default value: 3333)
WEBTOB_PREFER_IPV6	Configure this to 'Y' or '1' when IPv6 address is used.

"\${WEBTOBDIR}/lib" is the location of the WebtoB shared libraries. The path must be added to the following environment variables using the system library path.

Environment Variable	Description
LD_LIBRARY_PATH	A typical system library path of Linux/UNIX type systems.
LD_LIBRARY_PATH_64	When 64-bit WebtoB is used in Solaris(SunOS), this system library path is applied.
SHLIB_PATH	When 32-bit WebtoB is used in HP-UX, this system library path is applied.

1.5. Directory Structure

The following describes the WebtoB Directory structure.

```

WebtoB
|-- bin
|-- lib
|-- usrinc
|-- ap
|-- config
|-- path
|-- log
|-- docs
|-- license

```

bin

Contains executable files (wsm, wscfl, wsrcd, wsgst, wsboot, wsdown, etc.)

lib

Contains library files.

usrinc

Contains API header files.

ap

Contains application files.

config

Contains WebtoB configuration files.

path

Where the named pipe for process internal communication is created.

log

Contains log files.

docs

Contains the HTML documents registered by WebtoB by default.

license

Contains the WebtoB license file.

1.6. License Specific Functions

WebtoB provides the following functions according to the license type.

License Type	Description
TRIAL	<p>Trial license. Any WebtoB installed using the installer contains a default trial license.</p> <p>Supports a single HTH, and a maximum of 5 users.</p>
BASE	<p>For use as embedded in JEUS.</p> <p>Only supports a single HTH.</p> <p>In a UNIX/Linux environment, JSVPort is not used since only named pipes can be used to connect to JEUS.</p>
STANDARD	<p>One or more HTHs are supported.</p> <p>Can connect to JEUS without limitations and supports most WebtoB functions.</p>
ENTERPRISE	<p>Supports the following functions in addition to all of those provided in the STANDARD version.</p> <ul style="list-style-type: none"> ◦ Supports JSP/Servlet engine of the embedded engine. Only a single JEUS server (DAS) can be used in this case. ◦ Supports Reverse Proxy Group function for multi-configuration when connecting to another web application server through reverse proxy. ◦ Supports HTTP Method(MKCOL, COPY, MOVE, PORPFIND) for WebDAV. ◦ Supports FILTERS process for filter processing. Notably, SiteMinder Filter(wbSmISAPI) support for CA SSO. ◦ Provides WebAdmin.

2. Quick Start

This chapter briefly describes how to use WebtoB.

Assume that the host name where WebtoB is installed is "mynode". If WebtoB is installed with Installer, verify that the environment variables, such as WEBTOBDIR and PATH, are in the WebtoB environment file.



Refer to *WebtoB Installation Guide* for more information on installing WebtoB.

1. Compile the http.m file by using the WebtoB environment file compilation tool **wscfl**. Refer to [Configuration](#) for more information.

```
$ wscfl -i http.m
```

```
<$WEBTOBDIR\config\http.m>
```

```
*DOMAIN
webtob

*NODE
mynode
  WebtoBDir = "$WEBTOBDIR",
  SHMKEY = 54000,
  Docroot="docs/",
  Port = "8080",
  HTH = 1,
  Logging = "access_log",
  ErrorLog = "error_log",
  SysLog = "system_log"

*HTH_THREAD
hworker
  WorkerThreads = 8

*SVRGROUP
cgig
  SvrType = CGI
ssig
  SvrType = SSI

*SERVER
cgis
  SvrName = cgig,
  MinProc = 2,
  MaxProc = 10,
  ASQCount = 100
ssis
  SvrName = ssig,
  MinProc = 2,
  MaxProc = 10,
  ASQCount = 100
```

```
*URI
cgi-bin
  URI = "/cgi-bin/",
  SvrType = CGI

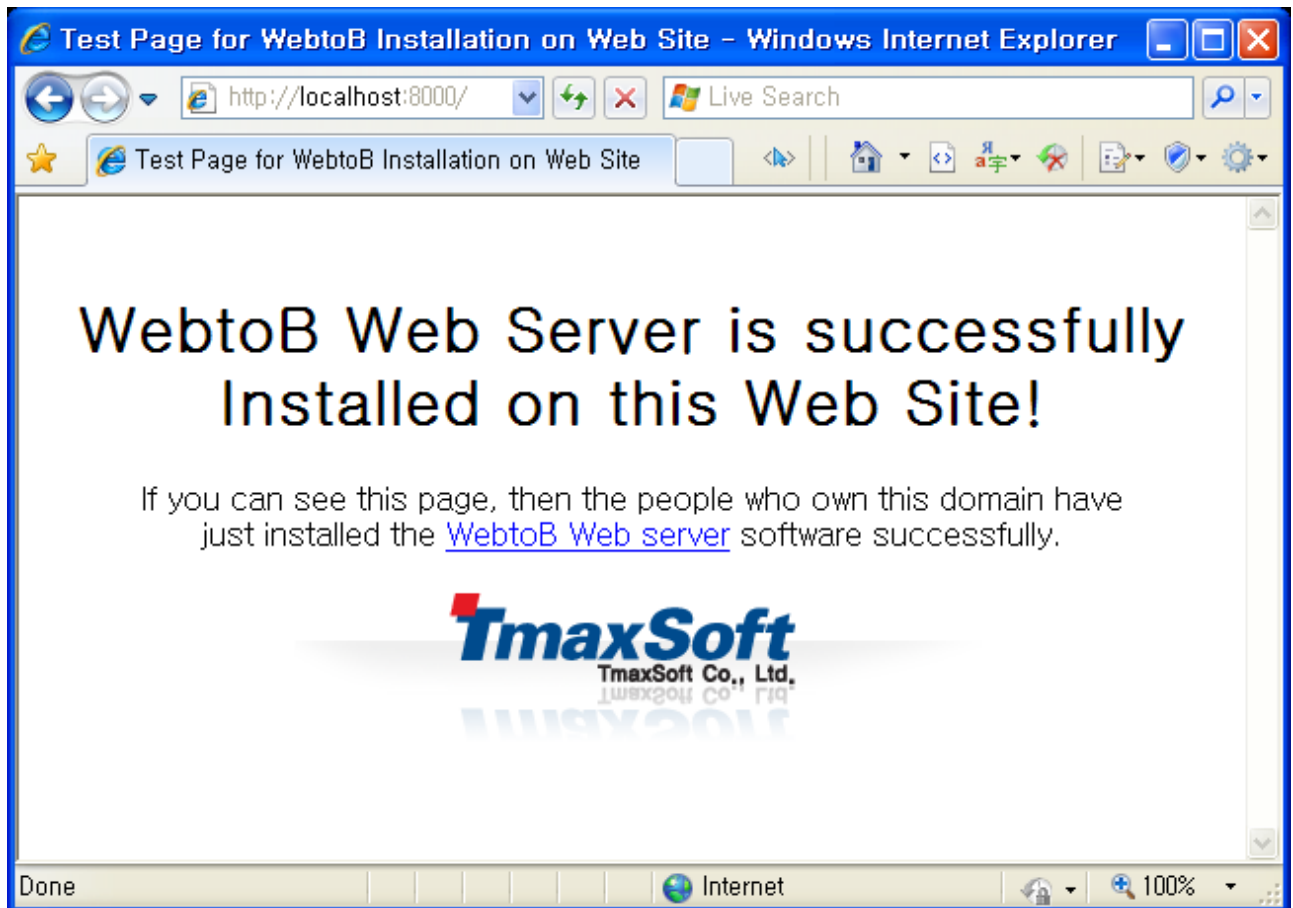
*ALIAS
cgi-bin
  URI = "/cgi-bin/",
  Realpath = "cgi-bin/"

*LOGGING
access_log
  Format = "default",
  Filename = "log/access.log",
  ArchiveFilename = "log/access_%Y%M%D.log",
  Option = "sync"
error_log
  Format = "",
  Filename = "log/error.log",
  ArchiveFilename = "log/error_%Y%M%D.log",
  Option = "sync"
system_log
  Format = "",
  Filename = "log/webtob.log",
  ArchiveFilename = "log/webtob_%Y%M%D.log",
  Option = "sync"
```

2. Verify that **wsconfig**, a binary configuration file, is created under "\${WEBTOBDIR}\config\".
3. To start WebtoB, enter **wsboot** in the command prompt.
4. Open the following URL in a web browser.

```
http://[IP Address]:[8080 or a user-specified port number]/
```

If WebtoB starts without any error, the following WebtoB Test Page appears.



WebtoB Test Page

5. To launch the WebtoB management tool, enter **wsadmin**.

In wsadmin, 'help' or 'h' shows a list of commands that monitors and controls WebtoB. Enter 'quit' to quit the WebtoB management tool.



Refer to [wsadmin Console Management Program](#) for more information on how to use wsadmin.

6. Shut down WebtoB with the **wsdown** command.

3. Configuration

This chapter describes how to configure the WebtoB environment and basic settings.

3.1. Overview

For the WebtoB environment, configure the WebtoB domain, node, and web server operation.

3.1.1. Configuration File Structure

The WebtoB configuration file consists of section definition, name, and items.

```
# Comment
*Section
Name
  # Comment
  Item = Setting value, # Comment
  ...,
  Item = Setting value
```

The start of each section is followed by the section name and then configuration items.

The following describes the sections that must be configured in WebtoB.

Section	Description
DOMAIN	Domain settings.
NODE	Node settings.
VHOST	Virtual host settings.
HTH_THREAD	Threads of the HTH process.
SVRGROUP	Logical groups of server processes for accessing application server processes through WebtoB.
SERVER	Sever process settings for processing requests.
SERVICE	Service settings to execute business logic through WebtoB.
DIRECTORY	Directory settings.
URI	Client request URI (Uniform Resource Identifier) used to determine the service to process the request.
ALIAS	Alias for the physical server path and URI.
DIRINDEX	Indexing method, icon, etc.
LOGGING	Logging format for client requests.
ACCESS	Client access control based on IP, network/netmask, and header information.

Section	Description
AUTHENT	Authentication control at the user and group levels for client access control.
EXT	Maps a client request to a specific process based on the requested file extension.
SSL	SSL to use on WebtoB.
PROXY_SSL	SSL to use when WebtoB is used as a proxy.
ERRORDOCUMENT	Error response actions.
EXPIRES	Expiry date of the server response specified in the server response header.
TCPGW	TCPGW that acts as a proxy that forwards a TCP connection request sent to a specific Port or IP to another server.
REVERSE_PROXY_GROUP	Reverse proxy group settings.
REVERSE_PROXY	HTTP Proxy settings used to forward a request from an external server that has no access to the internal server.
LOGLEVEL	Logging level for system log messages.
HEADERS	Settings for mapping specific HTTP header to a client request or response.
PRECEDING_COMMAND	Commands that perform preparation tasks, such as setting CPU Affinity for each process, at WebtoB startup.
FILTER	Filter module. Set the defined filter in a NODE, VHOST, or SVRGROUP section.
USERLOGFORMAT	Common logging format.
DISK_CACHE	Disk caching settings.
LOG_HANDLER	Settings for access log in a remote server.
IPGROUP	Settings for managing multiple IPs in groups.



The DOMAIN, NODE, and HTH_THREAD sections are required.

A section must conform to the following rules:

- The name of each section starts with an asterisk (*).
- The first line of each section starts with an asterisk (*) and the section name must be all in upper case.
- The starting line of a section must have at least one item configured. Only related items are allowed.
- The items that follow the section name are regarded as part of the section configuration until the end of the file or another section starts.

- The order in which sections are defined is not important.
- A section can be divided into multiple sections.

Section Name

Section name must be defined to identify the section. Section names must conform to the following rules:

- A section name must start from the first column of the first line.
- For section names, alphabetical characters, numbers, hyphens (-), and underlines (_) can be used. However, for SERVICE section names, hyphens (-) cannot be used.
- Name must be unique among the same sections.

Items

After the section name, define items as follows:

```
*Section
Name
  Item 1 = value,
  Item 2 = value,
  ...,
  Item n = value
```

Configure each item according to the following rules:

- Separate multiple items with a comma (,) as a delimiter.



If an item ends with a comma (,), the next line contains additional items to configure. The last item cannot end with a comma (,).

- An item name is not case sensitive.
- An item cannot start from the first column of a line.
- An item must start after one or more blank spaces.
- Multiple items can be defined on the same line.
- An item can be defined multiple times depending on the item.
- An item can be required or optional.

The value of an item can be one of the following types: Numeric, String, Literal, or Boolean. The following describes each value type.

Type	Description
Numeric	Number. If the value begins with '0', e.g., 0700, it is processed as an octal number. If the value starts with '0x', e.g., 0xff, it is processed as a hexadecimal number.
String	Character string.
Literal	Character string. The literal type must be embraced with double quotation marks (" ").
Boolean	Y/N or YES/NO.

Comment

Any text following the hash symbol (#) and up to the EOL character, is considered a comment.

Configuration Example

```
*DOMAIN
webtob

*NODE
mynode
  WebtoBDir = "$WEBTOBDIR",
  SHMKEY = 54000
  # Port = "80" is default

*HTH_THREAD
hworker
  WorkerThreads = 8

*SVRGROUP
cgig
  SvrType = CGI

*SERVER
cgis
  SvgName = cgig
  # MinProc = 1, MaxProc = 1 is default
```

3.1.2. Configuration File Compilation

Before starting WebtoB, the test configuration file must be compiled using the **wscfl** command to generate a binary configuration file that can be used by WebtoB.

In general, the text configuration file is named "http.m" and the binary configuration file is named "wsconfig". During the compilation process, wscfl checks the text file for syntax or system configuration errors.

The following command is an example of compiling the text configuration file "\$WEBTOBDIR/config/http.m".

```
wscfl -i http.m
```



To use a binary configuration file name instead of the default name of "wsconfig", use the environment variable `WEBTOB_CONFIG` or use the option to specify the file name when executing the program, such as `wscfl`, `wsadmin`, `wsboot`, or `wtdown`.

3.1.3. Configuration Item Description Rules

The next section covers the configuration items of each section. You must understand the following rules before reading the item descriptions.

- An item starting with the hash symbol (#) before its name is an optional item.
- For a string or literal variable, specify the buffer length enclosed in square brackets ([]).
For example, the length of a `string[n]` or `literal[n]` is limited to `n - 1`.
- If a default value exists, specify the value after the hash symbol.
- For a numeric variable, use parentheses to display the valid range.
If the maximum limit does not exist, omit the range. The maximum value must be less than or equal to `INT_MAX (2147483647)`.
- "\$ENV" means the environment variable can be referenced.
- "R.PATH" means that a relative path, if specified, is relative to `$WEBTOBDIR`.
- "LIST" means that you can specify multiple values separated by a comma (,) delimiter. If `LIST[n]` is specified, up to `n` values can be set. If `[n]` is omitted, there is no limit as long as the total length is smaller than the buffer size.
- "PM.LIST" means that you can use plus and minus symbols in front of the value to enable (+) or disable (-) the specified element. The plus symbol can be omitted.
- "MULTI" means the same item can be configured multiple times. "MULTI[n]" means that the same value can be specified up to `n` times.
- If "LIST" and "MULTI" are specified together, you can specify multiple values and duplicate values. In this case, the maximum number of values is limited to `n` in `LIST[n]`.

3.2. DOMAIN Section

The DOMAIN section configures the WebtoB domain. This section must be configured only once.

3.2.1. Configuration Items

The following is the configuration format of the DOMAIN section.

```
*DOMAIN
name # String[32]
      #DomainID = Numeric,           # 0 (0-255)
      #MaxSvc = Numeric,             # 512 (1-0xffff)
      #NHthChkTime = Numeric,       # 30 (0-)
      #CloudDasAddress = Literal[256], #"$DASURL", $ENV
      #CloudServiceGroupId = String[32]
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the DOMAIN section configuration items.

DOMAIN Section Name

- Type: String
- Range: Up to 31 characters
- Specifies WebtoB domain name.

DomainID

- Type: Numeric
- Range: 0 to 255
- Specifies the domain ID.
- The specified ID is used to distinguish domains in WebtoB internal communication.
- The values, 254 and 255, have special meanings.

Value	Description
254	Worker thread always processes "304 Not Modified". Processing the 304 response is not recommended due to slow processing time.
255	HTH always responds with "304 Not Modified" to a conditional-get request. The use of this value is not recommended because the browser or proxy may use old resource(s) even after the resources have been modified.

MaxSvc

- Type: Numeric
- Range: 1 to 65535(0xffff)

- Default value: 512
- Changes the maximum number of services that can be configured.
- Services include the URI and EXT sections as well as the SERVICE section.
- The URI and EXT sections are automatically registered in the SERVICE section.

NHthChkTime

- Type: Numeric
- Unit: Second
- Range: 0 to INT_MAX
- Default value: 30
- Alive check time for HTH to check if HTH is being blocked.
- After setting the NHthChkTime, if no response is received for two consecutive requests, an abnormal state is assumed and the HTH is restarted.

CloudDasAddress

- Type: Literal
- Range: Up to 255
- IP address and port number of DAS used by WebtoB and JEUS to detect the other's events for Auto Scaling in a cloud environment.
- Environment variable can be used for convenience, especially when creating a cloud image.

CloudServiceGroupId

- Type: String
- Range: Up to 31 characters
- ID to identify WebtoB per cloud service.

3.2.2. Example

The following is an example configuration of the DOMAIN section.

```
*DOMAIN  
webtob
```


3.3. NODE Section

The NODE section configures each node that composes WebtoB. This section affects all operations related to the node. If a virtual host is configured via the VHOST section, the NODE section configuration operates as a basic host.

In the NODE section, the following can be defined:

- WebtoB system path
- Top-level directory where documents for services are located (document root)
- Key value of shared memory
- Service IP and Port number configuration
- Other system configurations

3.3.1. Configuration Items

The following is the configuration format of the NODE section.

```
*NODE
name # String[128]
      WebtoBDir = Literal[256],           # $ENV
      ShmKey = Numeric,                   # (32768-262143)/(0x8000-0x3FFFF)
      #Nodename = Literal[128],           # "< *NODE.Name>", $ENV
      #User = String[32],
      #Group = String[32],
      #Hostname = Literal[128],           # "< OS hostname>"
      #DocRoot = Literal[256],            # "docs/", $ENV, R.PATH
      #SvrRoot = Literal[256],            # "$WEBTOBDIR", $ENV
      #Method = Literal[256],             # "GET,POST,HEAD,OPTIONS", PM.LIST
      #Hth = Numeric,                     # 1 (1-100)
      #HthQTimeout = Numeric,             # 0 (0-)f
      #Port = Literal[1024],               # "80" or "443" (1-65535), LIST[100]
      #JsvPort = Numeric,                  # 9999 (1-65535)
      #JsvSslPort = Numeric,               # 0 (1-65535)
      #JsvSslFlag = Boolean,               # N
      #JsvSslName = String[32],
      #RacPort = Numeric,                  # 3333 (1-65535)
      #SslFlag = Boolean,                  # N
      #SslName = String[32],
      #Timeout = Numeric,                  # 300 (1-)
      #InitialConnectionTimeout = Numeric, # 10 (0-)
      #RequestHeaderTimeout = Numeric,     # 60 (0-)
      #RequestBodyTimeout = Numeric,       # 0 (0-)
      #CacheKey = String[32],               # HOST_URI (HOST_URI, REAL_PATH)
      #CacheEntry = Numeric,                # 1024 (0-)
      #MaxCacheMemorySize = Numeric,        # 100 (0-)
      #CacheMaxFileSize = Numeric,         # 8192 (0-)
      #CacheMaxCompressSize = Numeric,     # 0 (0-)
      #CacheRefreshImage = Numeric,        # 3600 (0-)
      #CacheRefreshHtml = Numeric,         # 3600 (0-)
      #CacheRefreshJsv = Numeric,          # 3600 (0-)
      #CacheRefreshRproxy = Numeric,       # 3600 (0-)
```

```

#DiskCache = String[32],
#Keepalive = Boolean, # Y
#KeepaliveTimeout = Numeric, # 60 (1-)
#KeepaliveMax = Numeric, # 0 (0-)
#MaxUser = Numeric, # <auto> (1-)
#AppDir = Literal[256], # $ENV, R.PATH
#PathDir = Literal[256], # "path/", $ENV, R.PATH
#UsrLogDir = Literal[256], # "log/usrlog/", $ENV, R.PATH
#IconDir = Literal[256], # $ENV, R.PATH
#UserDir = Literal[256], # MULTI
#IndexName = Literal[256], # "index.html", LIST
#DirIndex = Literal[32],
#Options = Literal[256], # "HTML,CGI,SSI,PHP,JSV", PM.LIST
#ErrorDocument = Literal[256], # LIST[64]
#Logging = Literal[256], # LIST[4]
#ErrorLog = Literal[256], # LIST
#Filter = Literal[256], # LIST[64]
#Listen = Literal[1024], # LIST[100]
#IpcPerm = Numeric, # 0700 (0600-0777)
#ListenBacklog = Numeric, # <auto> (1-)
#SendBufferSize = Numeric, # <auto> (0-)
#RecvBufferSize = Numeric, # <auto> (0-)
#IpcSendBufferSize = Numeric, # <auto> (0-)
#IpcRecvBufferSize = Numeric, # <auto> (0-)
#HthIpcSendBufferSize = Numeric, # <auto> (0-)
#HthIpcRecvBufferSize = Numeric, # <auto> (0-)
#SvrIpcSendBufferSize = Numeric, # <auto> (0-)
#SvrIpcRecvBufferSize = Numeric, # <auto> (0-)
#MimetypesConfig = Literal[256], # "config/mime.types", $ENV, R.PATH
#DefaultMimetype = Literal[128], # "text/html"
#RPAFHeader = Literal[256],
#TimeOutStatus = Numeric, # 500 (511-599)
#Expires = Literal[256], # LIST[64]
#LimitRequestBody = Numeric, # 0 (0-)
#LimitRequestFields = Numeric, # 100 (0-)
#LimitRequestFieldSize = Numeric, # 8190 (0-16382)
#LimitRequestLine = Numeric, # 8190 (0-16382)
#ServerTokens = Literal[256], # "Off"
#ServiceOrder = Literal[256], # "uri,ext"
#IPCBasePort = Numeric, # 6666 (1-65535)
#TcpGW = Literal[1024], # LIST[64]
#DefaultCharset = Literal[32], # "Off"
#JsvAccessName = String[32],
#UseEtag = Boolean, # Y
#TerminateCgiUponClientClose = Boolean, # Y
#URLRewrite = Boolean, # N
#URLRewriteConfig = Literal[256], # R.PATH
#LogPerm = Numeric, # 0600 (0600-0777)
#SysLog = Literal[256], # "default_syslog"
#KeepAliveErrorStatusCode = Literal[256], # (HTTP status code), LIST
#MaxDechunkSize = Numeric, # 10485760(10MB) (0-)
#HtMlSForbidsWEBINF = Boolean, # Y
#HtLHthSendSocketBufferSize = Numeric, # 0 (0-)
#HtLHthRecvSocketBufferSize = Numeric, # 0 (0-)
#TraceAccessLog = Boolean, # N
#CheckURL = Boolean, # N
#CheckURLFrom = Literal[32], # "utf-8"
#CheckURLTo = Literal[32]
#CheckUrlJsvExcept = Boolean, # N

```

```

#SSIMaxDepth = Numeric,          # 16 (1-)
#ForceCacheModificationCheck = Boolean, # N
#DebugHTHMemory = Boolean,      # N
#Headers = Literal[256],        # LIST[15]
#DOSBlock = Boolean,           # N
#DOSBlockTableSize = Numeric,   # 3097 (1-)
#DOSBlockPageCount = Numeric,   # 5 (0-)
#DOSBlockPageInterval = Numeric, # 1 (1-)
#DOSBlockSiteCount = Numeric,   # 50 (0-)
#DOSBlockSiteInterval = Numeric, # 1 (1-)
#DOSBlockPeriod = Numeric,      # 30 (1-)
#DOSBlockWhiteList = Literal[256], # LIST[256], MULTI
#BindIPv6Only = Boolean,        # <OS default>
#JsvListen = Literal[1024],     # LIST[10]
#JsvSslListen = Literal[1024],  # LIST[10]
#UpperDirRestrict = Boolean,    # N
#CheckPingTimeoutStatus = Numeric, # 503 (511-599)
#IgnoreMissingColonErr = Boolean, # N
#HTTP2Fflag = Boolean           # N

```



Refer to [Configuration Item Description Rules](#) for more information on the symbols and details of the NODE section configuration items.

NODE Section Name

- Type: String
- Range: Up to 127 characters
- Specifies the node name.

WebtoBDir (Required)

- Type: Literal
- Range: Up to 255 characters
- Specifies the absolute WebtoB installation path, the home directory.
- Environment variables can be used. Set this to \$WEBTOBDIR for convenience.

ShmKey (Required)

- Type: Numeric
- Range: 32768 to 262143 / 0x8000 ~ 0x3FFFF
- Specifies the shared memory segment. WebtoB internal processes use shared memory to share information.
- Three keys are used as follows:
 - ShmKey

- (ShmKey + 1)
- (ShmKey + 2)

The key values must be different from any other key values used by other programs or applications. If the key values are used elsewhere, conflicts can occur with those programs or applications during WebtoB start up. WebtoB may fail to start due to shared memory conflicts.



Shared memory can be checked with the command "**ipcs**" in UNIX/Linux. Only when shared memory remains and is not returned after forcibly terminating WebtoB, can the shared memory be returned manually through the command "**ipcrm**".

Nodename

- Type: Literal
- Range: Up to 127 characters. Environment variables (\$NODENAME) can be used.
- Default value: NODE section name
- The physical name of the node that defines the name identified by the operating system.
 - UNIX/Linux: "uname -n" or "hostname" command
 - Windows: "hostname.exe" command
- The specified node name must be registered in the file.
 - UNIX/Linux: /etc/hosts
 - Windows: %SystemRoot%\system32\drivers\etc\hosts

User

- Type: String
- Range: Up to 31 characters
- Operates WebtoB with a configured account privilege.



Users can be configured only in UNIX/Linux. It is recommended to use a general account instead of the root account when configuring system security.

Group

- Type: String
- Range: Up to 31 characters

- Operates WebtoB with a configured group privilege.



Group can be configured only in UNIX/Linux. It is recommended to use a general group instead of the root group for system security.

Hostname

- Type: Literal
- Range: Up to 127 characters
- Default value: System host name
- Specifies the system host name for the host name field of HTTP response header.
- Hostname is used as the "SERVER_NAME" variable in CGI/PHP. Use this value instead when processing HTTP/1.0 requests that do not have the "Host" item.

DocRoot

- Type: Literal
- Range: Up to 255 characters
- Default value: "docs/"
- Specifies the absolute root directory of the documents served by WebtoB.
- Environment variables can be used. If DocRoot is a relative path, it is changed to an absolute path using \$WEBTOBDIR.
- Directory configuration pattern can be specified dynamically by using the following directives.

Directive	Description
%p	Replaces the requested port number.
%n	Replaces the Hostname or the Nth element of the IP address. If n is set to 0, the entire string is used. If the value starts with a minus symbol (-), count from the end of Hostname or IP address. If a plus symbol (+) follows, the rest of the Hostname or IP address is used.
%n.m	Replaces the Mth character of the Nth element. Minus (-) and plus (+) symbols can be used as previously described.
%%	Replaces a single percent symbol (%).



If a directive is used, DocRoot must be set as an absolute path. When an environment variable is used, the previously mentioned directives can be used only when the changed path is an absolute path.

SvrRoot

- Type: Literal
- Range: Up to 255 characters
- Default value: "\$WEBTOBDIR"
- If RealPath of the ALIAS section is set as a relative path, this item is specified as the root directory.
- Environment variables can be used. If SvrRoot is a relative path, it is changed to an absolute path based on \$WEBTOBDIR.

Method

- Type: Literal
- Range: Up to 255 characters
- Default value: "GET,POST,HEAD,OPTIONS"
- Specifies the HTTP request method. If the method is not specified in the client request, the request will not be processed.
- Supports GET, POST, HEAD, OPTIONS, CONNECT, TRACE, PROPFIND, PUT, DELETE, MKCOL, COPY, and MOVE.
- If a specific method is not serviced, add a hyphen (-) in front of the method name like "-OPTIONS".

Hth

- Type: Numeric
- Range: 1 to 100
- Default value: 1
- Specifies the number of HTTP Handler (HTH) processes.
- The number of concurrent users that one HTH process can handle is limited. In order to increase the number, set the desired value.
- The number of concurrent users that one HTH process can handle can be checked from the output of the "**wscfl**" command or "**wsboot**" execution.

```
$ wscfl -i http.m
Current configuration:
  Number of client handler(HTH) = 4
  Supported maximum user per node = 32552
  Supported maximum user per handler = 8138
```

HthQTimeout

- Type: Numeric

- Unit: Second
- Range: 0 to INT_MAX
- Default value: 0
- Specifies the timeout for user requests in a server queue.
- When a request cannot be processed due to a large volume of user requests, the request waits in the server queue until a server process becomes available. If a request waits longer than the specified time, remove the request from the queue and respond with "503 Service Unavailable".

Port

- Type: Literal
- Range: Up to 1023 characters, 1 to 65535
- Default value: "80" or "443" (If the SSLFlag is set to Y, "443" is the default value.)
- Specifies the port through which WebtoB listens.
- Up to 100 ports can be specified using a comma (,) as a delimiter.



If both the Port and Listen items are configured, the port configuration is ignored.

JsvPort

- Type: Numeric
- Range: 1 to 65535
- Default value: 9999
- Specifies the port number used to connect WebtoB with a server that executes servlets.

If WebtoB opens the port, JEUS connects to the port using the following configuration in WEBMain.xml.

```
<?xml version="1.0"?>
<web-container xmlns="http://www.tmaxsoft.com/xml/ns/jeus" version="6.0">
  <context-group>
    <group-name>MyGroup</group-name>
    <webserver-connection>
      <webtob-listener>
        <listener-id>webtob1</listener-id>
        <port>9900</port>
        <output-buffer-size>8192</output-buffer-size>
        <thread-pool>
          <min>5</min>
          <max>5</max>
          <step>5</step>
          <max-idle-time>30000</max-idle-time>
        </thread-pool>
      </webtob-listener>
    </webserver-connection>
  </context-group>
</web-container>
```

```

        <webtob-address>__webtob_address__</webtob-address>
        <registration-id>__webtob_server_name__</registration-id>
        <disable-pipe>true</disable-pipe>
    </webtob-listener>
</webserver-connection>
</context-group>
</web-container>

```

JsvSslPort

- Type: Numeric
- Range: 1 to 65535
- Specifies the service port used for SSL communication between WebtoB and JEUS.

When WebtoB opens the port, JEUS uses the following setting in "WEBMain.xml" to connect with WebtoB.

```

<?xml version="1.0"?>
<web-container xmlns="http://www.tmaxsoft.com/xml/ns/jeus" version="6.0">
  <context-group>
    <group-name>MyGroup</group-name>
    <webserver-connection>
      <webtob-listener>
        <listener-id>webtob1</listener-id>
        <port>10443</port> <!-- JsvSslPort -->
        <output-buffer-size>8192</output-buffer-size>
        <thread-pool>
          <min>5</min>
          <max>5</max>
          <step>5</step>
          <max-idle-time>30000</max-idle-time>
        </thread-pool>
        <webtob-address>__webtob_address__</webtob-address>
        <registration-id>__webtob_server_name__</registration-id>
        <disable-pipe>true</disable-pipe>
        <secure>
          <!-- Convert the authentication certificate specified
              in the WebtoB *SSL section into a Java KeyStore
              (JKS) format and use it as the trust store file -->
          <trust-store-file-path>
            __server_certstore.jks__
          </trust-store-file-path>
          <trust-store-file-password>
            __xxxx__
          </trust-store-file-password>
        </secure>
      </webtob-listener>
    </webserver-connection>
  </context-group>
</web-container>

```

JsvSslFlag

- Type: Boolean
- Default value: N
- Specifies whether to use the SSL/TLS protocol to connect to JEUS.
- SSL section item can be set in JsvSslName.

JsvSslName

- Type: String
- Range: 31 characters or less
- Specifies the SSL item when JsvSslFlag is set to 'Y'.

RacPort

- Type: Numeric
- Range: 1 to 65535
- Default value: 3333
- Specifies the port number used for inter-node communication needed for node management in a multi-node configuration. Unlike NodePort, RacPort is used by the **wsracd** daemon, which is a management process.

SSLFlag

- Type: Boolean
- Default value: N
- Specifies the option to use the SSL/TLS protocol for service port.
- The SSL section item that will be used as the SSLName configuration can be specified.

SSLName

- Type: String
- Range: Up to 31 characters
- SSL section name. Valid only when SSLFlag is set to Y.

Timeout

- Type: Numeric
- Unit: Second
- Range: 1 to INT_MAX
- Default value: 300

- Specifies the timeout to read/write data to/from a user socket.
- The Timeout value is applied when the user request is processed. If the user does not write to or read from the socket during the specified time, the socket is closed.

InitialConnectionTimeout

- Type: Numeric
- Unit: Second
- Range: 0 to INT_MAX
- Default value: 0
- Specifies the timeout for receiving the first request from the user.
- Closes the socket if the user does not send any requests after the TCP connection is made. This is not applicable if the user is currently sending a request or has previously sent one or more requests.
- This also applies to SSL connections. The socket will be closed if the user does not complete SSL connection within the timeout period or fails to send the first request after the connection is established.
- If set to 0, no timeout period is applied.

RequestHeaderTimeout

- Type: Numeric
- Unit: Second
- Range: 0 to INT_MAX
- Default value: 60
- Specifies the amount of time to wait for the HTTP Request Header.
- This configuration is applied when a user is sending a request. If the user does not send the requested HTTP Request Header(s) during this time, a "408 Request Timeout" error is sent to the user.
- In the case of a Pipelined Request (or if a new request comes before the response to the previous request is complete), the start time of the next request is immediately after the completion of the response to the prior request.
- If RequestHeaderTimeout is set to 0, there is no limit to the HTTP Request Header send time.

RequestBodyTimeout

- Type: Numeric
- Unit: Second
- Range: 0 to INT_MAX

- Default value: 0
- Specifies the amount of time to wait for the Body when an HTTP Request Body exists.
- This configuration is applied when a user is sending a request. If the user does not send the requested HTTP Request Body during this time, a "408 Request Timeout" error is sent to the user.
- If RequestBodyTimeout is set to 0, there is no limit to the HTTP Request Body send time.

CacheKey

- Type: String
- Range: Up to 31 characters
- Default value: HOST_URI
- Specifies the value to be used when a Key for caching is created.
- The following describes the settings.

Settings	Description
HOST_URI	<p>Uses the Request URI and the HOST value of Request Header to create a Key. Even if REQUEST_URI is the same, VHOST depends on HTTP_HOST. This means that the actual path can be different.</p> <p>Although VHOST is different, the same actual path can cause duplicate caching. This means that a response can be made for the same request with files that are cached differently depending on the host. If DocRoot differs by VHOST, it is better to use this value (HOST_URI). The response is faster than using REAL_PATH because the real path does not need to be calculated.</p> <p>Since responses handled by JEUS or Reverse Proxy cannot be calculated, HOST_URI value is used.</p>
REAL_PATH	<p>Uses the request URI's real path to create a Key. This value is applied when SVRTYPE is HTML. It helps to balance out problems that can occur when HOST_URI is used.</p> <p>Although HTTP_HOST is different, the actual path of the file with the same request can be the same. In this case, because only one file is cached regardless of HTTP_HOST, memory is not wasted. Because the actual path needs to be calculated, the response can be a little slower than using HOST_URI.</p>

CacheEntry

- Type: Numeric
- Range: 0 to INT_MAX
- Default value: 128
- Specifies the HTH cache hash table size.

- If set to 0, the response is not cached.

MaxCacheMemorySize

- Type: Numeric
- Unit: Mbytes
- Range: 0 to INT_MAX
- Default value: 100
- Specifies the maximum amount of memory used by a HTH process for caching.

MaxCacheMemorySize is not the total memory size that all HTH processes use. MaxCacheMemorySize applies to each HTH process.

- If cache items are larger than the maximum memory size, cache items seldom accessed by clients are deleted, and the administrator is informed by syslog.
- If set to 0, the response is not cached.

CacheMaxFileSize

- Type: Numeric
- Unit: bytes
- Range: 0 to INT_MAX
- Default value: 8192
- Specifies the maximum size of a response (Response Header + Response Body) that can be cached.
- If set to 0, the response is not cached.

CacheMaxCompressSize

- Type: Numeric
- Unit: bytes
- Range: 0 to INT_MAX
- Default value: 0
- Specifies the maximum size of a compressed response (Response Header + Response Body) to be cached.
- If set to 0, the compressed response is not cached.

CacheRefreshHtml

- Type: Numeric

- Unit: Second
- Range: 0 to INT_MAX
- Default value: 3600
- Specifies the timeout for deleting a response, which has "Content-Type" of "text/html", from the cache (when SVRTYPE is HTML). This means that a "text/html" response remains valid for the timeout period (sec) after being cached.
- If set to 0, the response remains valid without a timeout.
- A conditional-GET request or a request with the ForceCacheModificationCheck setting enabled checks to see if the cached response has been modified. If it has been modified, the cache is updated with the new version.

CacheRefreshImage

- Type: Numeric
- Unit: Second
- Range: 0 to INT_MAX
- Default value: 3600
- Specifies the timeout for deleting a response, which does not have "Content-Type" of "text/html", from the cache (when SVRTYPE is HTML). This means that a non-text/html response remains valid for the timeout period (sec) after being cached.
- If set to 0, the response remains valid without a timeout.
- A conditional-GET request or a request without response timeout set checks to see if the cached response has been modified. If it has been modified, the cache is updated with the new version.

CacheRefreshJsv

- Type: Numeric
- Unit: Second
- Range: 0 to INT_MAX
- Default value: 3600
- Specifies the value used to calculate the time a cached response is valid for when a response from JEUS is cached.
- For a client's request, WebtoB determines whether the cached content is valid by referring to the specified value (valid time).
 - First, if the Cache-Control:max-age value is in the Response Header when the header is being cached, the header is specified to be valid for the max-age period (second).
 - Second, if the Expires value is in the Response Header when the header is being cached, the header is specified to be valid for the Expires period (hour).

- Third, if the two previously mentioned Headers are not in the Response Header when it is being cached, the header is specified to be valid for the CacheRefreshJsv period (second). This means that if the Cache-Control:max-age and Expires values are not in the Response Header, the cached content is valid for the CacheRefreshJsv value (second) after a response from JEUS is cached.
- When specifying the valid time, priority is as follows.

```
Cache-Control:max-age > Expires > CacheRefreshJsv
```

- If set to 0 (Cache-Control:max-age and Expires values of the Response Header value do not exist), the cached response remains valid without a timeout. This means that if set to 0 (Cache-Control:max-age and Expires values of the Response Header values do not exist), the response is refreshed only when receiving a modified response from JEUS due to a Conditional-GET request.

CacheRefreshRproxy

- Type: Numeric
- Unit: Second
- Range: 0 to INT_MAX
- Default value: 3600
- Specifies the value to calculate the time a cached response is valid when a response handled by RefreshProxy is cached.
- For a client's request, WebtoB determines whether the cached content is valid by referring to the specified value (valid time).
 - First, if the Cache-Control:max-age value is in the Response Header when the header is being cached, the header is specified to be valid for the max-age value (second).
 - Second, if the Expires value is in the Response Header when the header is being cached, the header is specified to be valid until the Expires value (hour).
 - Third, if the two previously mentioned Headers are not in the Response Header when it is being cached, the header is specified to be valid for the CacheRefreshJsv value (second). This means that if the Cache-Control:max-age and Expires values are not in the Response Header, the cached content is valid for the CacheRefreshJsv value (second) after a response handled by ReverseProxy is cached.
- When configuring the valid time, priority is as follows.

```
Cache-Control:max-age > Expires > CacheRefreshRproxy
```

- If set to 0 (the Cache-Control:max-age and Expires values of the Response Header value do not exist), the cached response remains valid without a timeout. This means that if set to 0 (the Cache-Control:max-age and Expires values of the Response Header value do not exist), the response is refreshed only when receiving a modified response handled by ReverseProxy due to

the Conditional-GET request.

DiskCache

- Type: String
- Range: Up to 31 characters
- Specifies the DISK_CACHE section.

Keepalive

- Type: Boolean
- Default value: Y
- Specifies whether to use Keepalive (HTTP persistent connection).
- The following describes the setting values.

Value	Description
Y	The user can reuse a connection to allow a connection to process multiple requests.
N	The user must open a socket when a user request is received.

KeepaliveTimeout

- Type: Numeric
- Unit: Second
- Range: 1 to INT_MAX
- Default value: 60
- Specifies the timeout to wait for the next request.
- After the user request has been processed, the connection is disconnected after the specified time.

KeepaliveMax

- Type: Numeric
- Range: 0 to INT_MAX
- Default value: 0
- Specifies the number of requests that are processed before a connection is closed
- If set to 0, request count is not limited.

MaxUser

- Type: Numeric

- Range: 1 to INT_MAX
- Default value: Differs depending on the license and environment configuration. Generally the value is between 7000 and 8000.
- Specifies the maximum number of concurrent users that can access the node.
- MaxUser can be calculated as follows:

```
MaxUser= MIN(<openfile limit per operating system process>,
            <WebtoB's FD limit, usually 16384> *
            <HTH count> -
            <Number of FD for internal communication,
            SERVER section's total sum of MaxProc +
            HTH count + 2(WSM, HTL)>
```

AppDir

- Type: Literal
- Range: Up to 255 characters
- Specifies the location of the server executable file written in WBAPI.
- Environment variables can be used. If AppDir is a relative path, it is changed to an absolute path based on \$WEBTOBDIR.



If AppDir is set in the SVRGROUP section, use the configuration of the SVRGROUP.

PathDir

- Type: Literal
- Range: Up to 255 characters
- Default value: "path/"
- Specifies the path to the "webtob.pid" file, which records named-pipe for internal communication and each process pid.
- Environment variables can be used. If PathDir is a relative path, it is changed to an absolute path based on \$WEBTOBDIR.

UsrLogDir

- Type: Literal
- Range: Up to 255 characters
- Default value: "log/usrlog/"
- Specifies the path of user logs.

- Environment variables can be used. If `UsrLogDir` is a relative path, it is changed to an absolute path based on `$WEBTODIR`.



`UsrLogDir` uses the first configured value from among the `SERVER`, `SVRGROUP`, and `NODE` sections.

IconDir

- Type: Literal
- Range: Up to 255 characters
- Specifies the path where icon, which is used in the Directory indexing function, is located.
- `IconDir` must be set as a relative path relative to `(/)` and `DocRoot` since it is used in an html tag.

UserDir

- Type: Literal
- Range: Up to 255 characters
- The following three options can be used when configuring a directory for each user.
 - `<per-user-home-directory>` configuration
 - disabled [`<user-list>`]
 - enable [`<user-list>`]
- In order to configure `"~/public_html"` as the user specified directory and allow only specified users (`user1`, `user2`, and `user3`) to access to the directory, see the following example.

```
*NODE
mynode
    ...,
    UserDir = "public_html",
    UserDir = "disabled", # globally disabled
    UserDir = "enable user1 user2 user3",
    ...
```

IndexName

- Type: Literal
- Range: Up to 255 characters
- Default value: `"index.html"`
- Specifies the index page name for a service directory request.

For example, If the request from client is `"/somedir/"`; the path `"/somedir/index.html"` is serviced.

DirIndex

- Type: Literal
- Range: 31 characters
- Specifies the directory index name set in the DIRINDEX section.



In order to use the directory indexing function, "INDEX" must be added to options. The DIRINDEX section configuration can modify the directory indexing function operations.

Options

- Type: Literal
- Range: Up to 255 characters
- Default value: "HTML,CGI,SSI,PHP,JSV,USER"
- Specifies services and other operation methods provided by WebtoB.
- If a certain option is not used, input a hyphen (-) preceding the option name.

For example, "CGI" means WebtoB does not use CGI server and hence any client requests requiring CGI server are denied.

- The following describes the available options.

Option	Description
HTML	Uses the HTML server.
CGI	Uses the CGI server.
PHP	Uses the PHP server.
SSI	Uses the SSI server.
JSV	Uses the JSV (JEUS) server.
INDEX	When a client sends a request to a directory without specifying a file name, this option shows directory content. For example, for a "/docroot/dir/" request, the file name and information in the directory are sent to the client.
ALL	"HTML,CGI,PHP,SSI,JSV,INDEX", which is the same as the default value.

Option	Description
StrictAuthorizationHdr Format	<p>For HTTP requests, the Authorization header item format is checked when authenticating through the AUTHENT section.</p> <p>If the format of the HTTP request Authorization header item does not match the format defined in HTTP 1.1, or the authentication type is not "Basic" or "Digest", an error is sent to the user.</p> <p>If a hyphen (-) is set, the format of the Authorization header item is checked. If there is no authentication type, the type is considered to be "Basic".</p> <p>For example, if a hyphen is set when the Authorization header item value is "YXV0aDIucGFzcp0bWF4MTIzNA==" without an authentication type, the type is considered to be "Basic" and authentication proceeds. If a hyphen is not set, an error is sent to the user without performing the authentication process.</p>
IgnoreExpect100Continue	If "Expect: 100-continue" is included in a request header when requesting HTTP/1.1, it is ignored. Do not respond to the "Expect:100-continue" statement with "100 Continue".
ProcessIncompleteRequest	Sends the client message body to the server before receiving the entire amount of the POST request body content specified by the Content-Length.
ExcludeAllowHeaderOnError	Excludes "Allow" header when sending an error response for a request that calls a method that has been disabled via *NODE.Method setting.

ErrorDocument

- Type: Literal
- Range: Up to 255 characters
- Specifies the use of a user specified page instead of the HTTP error page. Configure the names in the ERRORDOCUMENT section.
- Up to 64 items can be configured by separating each item with a comma (,).

Logging

- Type: Literal
- Range: Up to 255 characters
- Specifies the access log name set in the LOGGING section.
- Up to 4 items can be configured by separating each item with a comma (,).

ErrorLog

- Type: Literal

- Range: Up to 255 characters
- Specifies the error log name set in the LOGGING section.

Filter

- Type: Literal
- Range: Up to 255 characters
- Specifies the FILTER section name to be used.
- Up to 64 items can be configured by separating each item with a comma (,).
- If Filter from the NODE section is set, it is applied to all requests.

Listen

- Type: Literal
- Range: Up to 1023 characters
- Specifies IP address and port number pairs to listen for incoming requests.
- Up to 100 items can be configured by separating each item with a comma (,).
- Listen has a higher priority than Port. If Listen is set, Port is ignored.

IPCPerm

- Type: Numeric
- Range: 0600 to 0700
- Default value: 0700
- Specifies the access permissions for WebtoB internal processes including WSM, HTL, HTH, HTMLS, and CGIS.



Can be used only in UNIX/Linux.

ListenBacklog

- Type: Numeric
- Range: 1 to INT_MAX
- Default value: 511 (UNIX/Linux), <OS default> (Windows)
- Specifies the maximum number of queued pending connections. This limits the number of sockets concurrently trying to access the service port.



May be related to the number of sockets in SYN_RECV state in the netstat

command.

SendBufferSize

- Type: Numeric
- Unit: bytes
- Range: 0 to INT_MAX
- Default value: OS default
- Specifies the TCP transfer buffer size.

RecvBufferSize

- Type: Numeric
- Unit: bytes
- Range: 0 to INT_MAX
- Default value: OS default
- Specifies the TCP receiving buffer size.

IpcSendBufferSize

- Type: Numeric
- Unit: bytes
- Range: 0 to INT_MAX
- Default value: OS default
- Specifies the size of the send buffer for messaging between internal processes.

IpcRecvBufferSize

- Type: Numeric
- Unit: bytes
- Range: 0 to INT_MAX
- Default value: OS default
- Specifies the size of the send buffer for HTH messages.

HthIpcSendBufferSize

- Type: Numeric
- Unit: bytes

- Range: 0 to INT_MAX
- Default value: OS default
- Specifies the size of the send buffer for HTH messages.

HthIpcRecvBufferSize

- Type: Numeric
- Unit: bytes
- Range: 0 to INT_MAX
- Default value: OS default
- Specifies the size of the receive buffer for HTH messages.

SvrIpcSendBufferSize

- Type: Numeric
- Unit: bytes
- Range: 0 to INT_MAX
- Default value: OS default
- Specifies the size of the send buffer for server process messages.

SvrIpcRecvBufferSize

- Type: Numeric
- Unit: bytes
- Range: 0 to INT_MAX
- Default value: OS default
- Specifies the size of the receive buffer for server process messages.

MimetypesConfig

- Type: Literal
- Range: Up to 255 characters
- Default value: "config/mime.types"
- Specifies the path for "MIME-Type" configuration file that maps MIME-Type and extensions.
An absolute or a relative path from "\$WEBTOBDIR" can be specified.
- If set to nothing(""), the function is not used.
- Set the MIME-Type configuration file as follows:

```
# The line starting with '#' is recognized as comment.
# Configure MIME-Type and the extension to map by separating them with blank space.
# Multiple extensions can be set by seperating each extension with a blank space.
# MIME-Type      Extensions
text/html        html htm
image/jpeg       jpeg jpg jpe
```

DefaultMimetype

- Type: Literal
- Range: Up to 127 characters
- Default value: "text/html"
- Specifies the Default Content-Type for documents that cannot decide the MIME-Type.



If the item is configured in the DIRECTORY section, it is used first. The next to be used is the item in the SVRGROUP section. If both are not configured, the configuration in the VHOST or NODE section is used.

RPAFHeader

- Type: Literal
- Range: Up to 255 characters
- Sets the remote IP that was changed by a proxy server to the IP of the host that sent the request.
- Adding RPAFHeader allows users to specify a header.

```
(Example: RPAFHeader = "X-Forwarded-For")
```

- If a header set with RPAFHeader is contained, a remote IP will be changed to the header IP.

TimeoutStatus

- Type: Numeric
- Range: 511 to 599
- Default value : 500
- If a timeout occurs, this value is returned in HTTP Status Code.

Expires

- Type: Literal
- Range: Up to 255 characters

- Specifies the EXPIRES section name.
- Up to 64 items can be configured by separating each item with a comma (,).

LimitRequestBody

- Type: Numeric
- Unit: bytes
- Range: 0 to LONG_MAX
- Default value: 0
- Specifies the client request body size supported by the server through the HTTP protocol.
- Base the value on the "Content-Length" value in the HTTP Request Header. If the body size is larger than the configured value, responds with "413 Request Entity Too Large".
- HTTP Request Body 2G (Long type), It can be processed from JEUS 7 onwards.



In the case of a chunked-request, the MaxDechunkSize configuration is used.

LimitRequestFields

- Type: Numeric
- Range: 0 to INT_MAX
- Default value: 100
- Specifies the maximum number of a client request's HTTP header fields.
- If set to 0, the field count is not limited.

LimitRequestFieldSize

- Type: Numeric
- Unit: bytes
- Range: 0 to 16382
- Default value: 8190
- Specifies the maximum size of a client request's HTTP header field.
- "WJpV2" must be used for over 10000 bytes of request Header.

LimitRequestLine

- Type: Numeric
- Unit: bytes

- Range: 0 to 16382
- Default value: 8190
- Specifies the maximum size of a client request's HTTP line.
- "WJpV2" must be used for over 10000 bytes of request lines.

ServerTokens

- Type: Literal
- Range: Up to 255 characters
- Default value: "Off"
- Configures the server HTTP response header.
- The following are descriptions of configuration values.

Value	Description
Off	Do not use the "Server" item of Response Header.
Prod[uctOnly]	"WebtoB"
Min[imal]	"WebtoB/4.1.3"
OS	(Example: "WebtoB/4.1.3 LINUX-K2.6_x86 libc2.3")
Full	(Example: "WebtoB/4.1.3 LINUX-K2.6_x86 libc2.3")
Custom	user-specified-name

ServiceOrder

- Type: Literal
- Range: Up to 255 characters
- Default value: "uri,ext"
- Specifies the priority of the URI and EXT sections for determining the server and service to process an HTTP request.
- The following are descriptions of configuration values.

Value	Description
uri, ext	Checks the URI section configuration. If the configuration does not exist, checks the EXT section configuration.
ext, uri	Checks the EXT section configuration. If the configuration does not exist, checks the URI section configuration.



If neither option is specified, the default HTML service (worker thread) processes the request.

IPCBasePort

- Type: Numeric
- Range: 1 to 65535
- Default value: 6666
- Specifies the base port used for communication between WebtoB internal processes. Valid only in Windows. (PIPE communication for UNIX)



WSM uses (IPCBasePort + 1). HTL uses (IPCBasePort + 2). HTH uses (IPCBasePort + 4 + <hth-id>).

TcpGW

- Type: Literal
- Range: Up to 1024 characters
- Specifies the TCPGW section name.
- Up to 1024 items can be configured by separating each with a comma (,) or setting multiple times.

DefaultCharset

- Type: Literal
- Range: Up to 32 characters
- Default value: "Off"
- Specifies the charset of the response when charset is not included in the request content-type.

Value	Description
On	Specifies "ISO-8859-1" as the default value.
Off	Charset is not added.
<custom-charset>	Uses the specified <custom-charset> as the default value.



While DefaultCharset can be specified in multiple section, only one definition is used. The DIRECTORY, SVRGROUP, VHOST, and NODE sections are searched sequentially. The first specified DefaultCharset value to be found is used.

JsvAccessName

- Type: String
- Range: Up to 31 characters

- Specifies the ACCESS section name to be used for access control of JsvPort.

UseETag

- Type: Boolean
- Default value: Y
- Specifies the option to use HTTP ETag. If set to Y, ETag, if possible, is added to HTTP response and ETag of HTTP request is used. If set to N, ETag is not added to HTTP response and ETag of HTTP request is ignored.

TerminateCgiUponClientClose

- Type: Boolean
- Default value: Y
- Specifies the option to forcibly terminate a running CGI or PHP process when a client ends a connection while the CGI or PHP request is being processed. If set to Y, the process is forcibly terminated. If set to N, the request is processed and then the process is terminated.

URLRewrite

- Type: Boolean
- Default value: N
- Specifies the option to use the URLRewrite function.



To use this, URLRewriteConfig must be set.

URLRewriteConfig

- Type: Literal
- Range: Up to 255 characters
- Specifies the path to the configuration file for the URLRewrite function. Refer to [URLRewrite](#) for more information.

LogPerm

- Type: Numeric
- Range: 0600 to 0777
- Default value: 0600
- Specifies the permission to access WebtoB log files (system log, access log, error log)
- LogPerm configuration is the same as the file access permissions used in UNIX. This

configuration can be used only in a UNIX/Linux environment.

SysLog

- Type: Literal
- Range: Up to 255 characters
- Default value: "default_syslog"
- Specifies the system log name set in the LOGGING section.
- If the default value (default_syslog) is used, the FileName of the LOGGING section is "log/system_%Y%M%D.log".

KeepAliveErrorStatusCode

- Type: Literal
- Range: Up to 255 characters
- After sending an error response, the user connection is kept alive (persistent connection, keep-alive) and the corresponding HTTP Status Code is configured.
- Multiple status codes can be specified by separating each code with a comma (,).
- Normally, if WebtoB sends an error response with status code of 3xx (excluding "304 Not Modified"), 4xx, or 5xx, the client connection is closed. However, if WebtoB sends a status code specified in this option, the client connection is maintained.
- The following is an example of WebtoB keeping the connection with client alive after responding with either a "302 Found" or "404 Not Found" message.

```
*NODE
mynode
    ...,
    KeepAliveErrorStatusCode = "302,404",
    ...
```

MaxDechunkSize

- Type: Numeric
- Unit: bytes
- Range: 0 to INT_MAX
- Default value: 10485760
- Specifies the maximum body size of a non-chunked request that is converted from a chunked request by HTH.
- Respond with "500 Internal Server Error" to the request whose chunk body is larger than the configured value.

HtmsForbidsWEBINF

- Type: Boolean
- Default value: Y
- Decides the option to send the "403 Forbidden" response if a user request includes /WEB-INF/ (case-insensitive).
- If set to Y, users cannot access configuration files of the application server such as JEUS.

HtlHthSendSocketBufferSize

- Type: Numeric
- Unit: bytes
- Range: 0 to INT_MAX
- Default value: 0 (0 means the default value of the OS where WebtoB is installed.)
- Specifies the buffer size for a socket used between HTL and HTH, WebtoB internal processes.
- This configuration is added due to the sendmsg() bug in the HP-UX environment. HtlHthSendSocketBufferSize configures the buffer size to be used to call sendmsg() through the UNIX domain socket (named-pipe).
- Specify the value to be smaller than the value of HtlHthRecvSocketBufferSize.



In general, the user does not need to configure this.

HtlHthRecvSocketBufferSize

- Type: Numeric
- Unit: bytes
- Range: 0 to INT_MAX
- Default value: 0 (0 means the default value of the OS where WebtoB is installed.)
- Configures the socket buffer size (receiver buffer) that is used for communication between WebtoB internal processes: HTL and HTH.



In general, the user does not need to configure this.

TraceAccessLog

- Type: Boolean
- Default value: N
- If set to Y, client request process procedures are recorded to the access log.



If set to Y, access logs will be two or three times the usual number.

- The following describes types of logs depending on when they are saved.

Log	Description
[ARRIVED]	The point when the request is received in HTH.
[SEND-TO-SERVER <type> <server-index>]	The point when the request is sent to the server.
[QUEUED <type> <server-name>]	The point when the request enters the server queue.
[JSV SUSPEND FAILOVER]	When the request, which was in the queue at the point when JSV server was suspended, is allocated to another JSV server.

CheckURL

- Type: Boolean
- Default value: N
- Specifies the option to convert the charset of HTTP request URL path.
- Charset is a combination of CCS (Coded Character Set) and CES (Character Encoding Scheme). Korean characters generally use the EUC-KR or UTF-8 character set.
- If the charset used by a request's URL path differs from the charset used by server, convert the charset by using CheckURL.
- If set to Y, CheckURLFrom and CheckURLTo must be set.
- If CheckURL and VHOST sections are both configured, VHOST takes precedence over CheckURL.

CheckURLFrom

- Type: Literal
- Range: Up to 31 characters
- Default value: "UTF-8"
- Specifies the charset used by the HTTP request URL path.



Only "EUC-KR" and "UTF-8" are supported.

CheckURLTo

- Type: Literal
- Range: Up to 31 characters

- Specifies the charset used by the server.

CheckUrlJsvExcept

- Type: Boolean
- Default value: N
- Option to disable CheckURL setting when forwarding requests to JEUS.
- If set to Y, request URI is not converted when sent to JEUS.

SSIMaxDepth

- Type: Numeric
- Range: 1 to INT_MAX
- Default value: 16
- Specifies the depth limit of nested-include statements when a SSI (server-side include) request is processed.

SSI can include other SSI resources by using the include directive. Using the include directive on an already included SSI resource may cause a loop to occur. Proper configuration of include depth can prevent wasting server resources from an infinite loop.

ForceCacheModificationCheck

- Type: Boolean
- Default value: N
- Specifies whether to always check if the cached response is valid even when the request is not Conditional-Get (when SVRTYPE is HTML).
- HTH cache verifies the cached response is valid only when a Conditional-Get request comes in.



If set to Y, a stat() system call occurs for every request to verify the cached response is valid, lowering performance.

DebugHTHMemory

- Type: Boolean
- Default value: N
- Set to Y to monitor HTH memory usage.



In order to use the command "wsadmin -C hthmem", DebugHTHMemory must be set to Y.

Headers

- Type: Literal
- Range: Up to 255 characters
- Specifies the HEADERS section name.
- Up to 15 items can be configured by separating each item with a comma (,).

DOSBlock

- Type: Boolean
- Default value: N
- Set to Y to use DoS attack prevention configuration.

DOSBlockTableSize

- Type: Numeric
- Range: 1 to INT_MAX
- Default value: 3097
- Specifies the hash table size used internally in DoS attack prevention functions.

DOSBlockPageCount

- Type: Numeric
- Range: 0 to INT_MAX
- Default value: 5
- Blocks the user IP that requests the same page more than the specified value in DOSBlockPageInterval during the time specified in DOSBlockPeriod.
- If set to 0, DoS attack on the same page is not checked.

DOSBlockPageInterval

- Type: Numeric
- Unit: Second
- Range: 1 to INT_MAX
- Default value: 1
- Specifies the period to check for a DoS attack on the same page.

DOSBlockSiteCount

- Type: Numeric
- Range: 0 to INT_MAX
- Default value: 50
- Blocks the user IP that requests the site more than the specified value in DOSBlockSiteInterval during the time specified in DOSBlockPeriod.
- If set to 0, DoS attack on the site request is not checked.

DOSBlockSiteInterval

- Type: Numeric
- Unit: Second
- Range: 1 to INT_MAX
- Default value: 1
- Specifies the period to check for a DoS attack on the site.

DOSBlockPeriod

- Type: Numeric
- Unit: Second
- Range: 1 to INT_MAX
- Default value: 30
- Specifies the length of time to block the IP suspected of a DoS attack.

DOSBlockWhiteList

- Type: Literal
- Range: Up to 255 characters
- Specifies the IPs to exclude from DoS attack prevention. If a Proxy server is used, specify the server IP.
- Multiple IPs can be specified by separating each IP address with a comma (,). If a large number of IPs are to be specified, define DOSBlockWhiteList multiple times.
- Up to 256 IPs can be configured.

BindIPv6Only

- Type: Boolean
- Default value: <OS default>
- Specifies the range of the address to bind when IPv6 is used. If not set, the OS default

configuration is used.

Value	Description
Y	When IPv6 is used, binds the IPv6 address only.
N	When IPv6 is used, binds both IPv4 and IPv6 so they are used for dual-stack.

JsvListen

- Type: Literal
- Range: Up to 1023 characters
- If the JSV service port is supposed to be open only for a specific IP, set to "<IP>:<Port>".
- Up to 10 items can be specified by using a comma (,) as the delimiter.
- Overrides the JsvPort configuration. If JsvListen is configured, the JsvPort configuration is ignored.

JsvSslListen

- Type: Literal
- Range: Up to 1023 characters
- If the JSVSsl service port is supposed to be open only for a specific IP, set to "<IP>:<Port>".
- Up to 10 items can be specified by using a comma (,) as the delimiter.
- Overrides the JsvSslPort configuration. If JsvSslListen is configured, the JsvSslPort configuration is ignored.

UpperDirRestrict

- Type: Boolean
- Default value: N
- If set to Y, this returns a 403 Forbidden error if the request URI includes '/' to move up to the parent directory.

CheckPingTimeoutStatus

- Type: Numeric
- Range: 511 to 599
- Default value: 503
- HTTP status code to return if a ping request sent to the JSV server times out.

IgnoreMissingColonErr

- Type: Boolean

- Default value: N
- This determines whether to throw an error when the HTTP request header does not comply with the "name:value" format.
- The following describes the setting values.

Value	Description
Y	Ignores the header.
N	Sends a "400 Bad Request" response.

HTTP2Flag

- Type: Boolean
- Default value: N
- If set to Y, HTTP/2 connections and requests are allowed.
- This applies only when SSL/TLS protocols are used. Therefore, SSLFlag must be set to the NODE or VHOST section that will use HTTP/2.

3.3.2. Example

The following is an example configuration of the NODE section.

```
*NODE
mynode
  WebtoBDir = "$WEBTOBDIR", # Refer to environment variable WEBTOBDIR.
  SHMKEY = 0x8000,          # Can configure as hex number by adding 0x at front.
  Docroot="docs/",         # Same meaning with "$WEBTOBDIR/docs/"
  Port = "8080",           # Caution! Not Numeric but Literal type
  JsvPort = 9900,
  HTH = 1,
  Logging = "access_log",
  ErrorLog = "error_log",
  SysLog = "system_log"
```

3.4. VHOST Section

The VHOST section configures environment for using virtual hosting with WebtoB. Virtual hosting allows a single WebtoB server to service different URLs with different document directories so it appears as there are multiple web servers.

The following are defined in VHOST section.

- Node name of a Virtual Host
- Host name of a Virtual Host

- Top-level directory that includes HTML documents for a virtual host

The following configurations defined in NODE section, can be re-defined.

- Paths to various log message directories
- Indexing mechanism and path to the directory icon

Configuration items defined in the NODE section are overwritten when redefined in the VHOST section. Method, Expires, UsrLogDir, EnvFile, ErrorDocument, Logging Option items behave the same as in the NODE section

3.4.1. Configuration Items

The following is the environment configuration format of the VHOST section.

```
*VHOST
name # String[32]
    Hostname = Literal[128],
    #HostAlias = Literal[1024], # LIST
    #DocRoot = Literal[256], # < *NODE's>, $ENV, R.PATH
    #Method = Literal[256], # "GET,POST,HEAD,OPTIONS", PM.LIST
    #Port = Literal[1024], # "80" or "443", 1 ~ 65535, LIST[100]
    #Listen = Literal[1024], # LIST[100]
    #SslFlag = Boolean, # N
    #SslName = String[32],
    #IconDir = Literal[256], # $ENV, R.PATH
    #UserDir = Literal[256], # MULTI
    #EnvFile = String[256], # $ENV, R.PATH
    #IndexName = Literal[256], # "index.html", LIST
    #DirIndex = Literal[32], # LIST
    #Options = Literal[256], # "HTML,CGI,SSI,PHP,JSV", PM.LIST
    #ErrorDocument = Literal[256], # LIST[64]
    #Logging = Literal[256], # LIST[4]
    #ErrorLog = Literal[256], # LIST
    #Filter = Literal[256], # LIST[64]
    #DefaultMimetype = Literal[128],
    #Expires = Literal[256], # LIST[64]
    #ServiceOrder = Literal[256], # < *NODE's>
    #Keepalive = Boolean, # < *NODE's>
    #KeepaliveTimeout = Numeric, # < *NODE's> (1-)
    #KeepaliveMax = Numeric, # < *NODE's> 0 (0-)
    #Timeout = Numeric, # < *NODE's> (1-)
    #DefaultCharset = Literal[32],
    #URLRewrite = Boolean, # N
    #URLRewriteConfig = Literal[256], # R.PATH
    #Headers = Literal[256] # LIST[15]
    #CheckURL = Boolean, # N
    #CheckURLFrom = Literal[32], # "utf-8"
    #CheckURLTo = Literal[32]
    #CheckUrLJsvExcept = Boolean, # N
```



For more information about the symbols or other description rules for the configuration of the section and setting items, refer to [Configuration Item](#)

VHOST section name

- Type: String
- Range: Up to 31 characters
- Specifies the VHOST section name.
- Up to 64 items can be specified in VHOST.

Hostname (Required)

- Type: Literal
- Range: Up to 127 characters
- Specifies the host name used to access the virtual host.
- When using name-based virtual hosting, each VHOST section must have unique host name used as an identifier.

HostAlias

- Type: Literal
- Range: Up to 1023 characters
- When adding a host in addition to Hostname, register the new host with Host Alias.
- Multiple hosts can be configured by separating each host with a comma (,). Asterisks (*) and question marks (?) can also be used. "*" means all hosts.

DocRoot

- Type: Literal
- Range: Up to 255 characters
- Specifies path to the top-level directory that contains HTML documents serviced by a virtual host.
- If not set, DocRoot of the NODE section is used.
- Directory configuration pattern can be configured dynamically by using the following directives.

Directive	Description
%p	Replaces with the requested port number.

Directive	Description
%n	Replaces with the Hostname or the Nth element of the IP address. If n is set to 0, the entire string is used. If the value starts with a minus symbol (-), count from the end of Hostname or IP address. If a plus symbol (+) follows, the rest of the Hostname or IP address is used.
%n.m	Replaces with the Mth character of Nth element. Minus (-) or plus (+) symbols can be used as previously described.
%%	Replaces with a single percent (%) symbol.



If a directive is used, DocRoot must be set as an absolute path. When an environment variable is used, the previously mentioned directives can be used only when the changed path is an absolute path.

Method

- Type: Literal
- Range: Up to 255 characters
- Default value: "GET,POST,HEAD,OPTIONS"
- Specifies the HTTP Request method. If the method is not specified in the client request, the request will not be processed.
- Supports GET, POST, HEAD, OPTIONS, and CONNECT, TRACE, PROPFIND, PUT, DELETE, MKCOL, COPY, and MOVE.
- If a specific method is not serviced, add hyphen (-) in front of the method name like "-OPTIONS".

Port

- Type: Literal
- Range: Up to 1023 characters, 1 ~ 65535
- Default value: "80" or "443" (If the SSLFlag is set to Y, "443" is the default value.)
- Specifies the service port that can be accessed by the user.
- Up to 100 elements can be specified by separating each element with a comma (,).



If both the Port and Listen items are configured, the Port item is ignored.

Listen

- Type: Literal

- Range: Up to 1023 characters
- Set as "<IP>:<Port>" to open the service port only for specific IPs.
- Up to 100 items can be configured by separating each item with a comma (,).
- Listen has a higher priority than Port. If Listen is configured, Port is ignored.

SSLFlag

- Type: Boolean
- Default value: N
- Specifies whether to use the SSL/TLS protocol for service port.
- The SSL section item that will be used as the SSLName configuration can be specified.

SSLName

- Type: String
- Range: Up to 31 characters
- SSLName can be used when SSLFlag is set to Y. Specifies the SSL section item to be used.

IconDir

- Type: Literal
- Range: Up to 255 characters
- Specifies the path where icon, which is used in the directory indexing function, is located.
- IconDir must be set as a relative path relative to (/) and DocRoot since it is used in an html tag.

UserDir

- Type: Literal
- Range: Up to 255 characters
- Used when configuring per user directories.
- Three options for configuring per user directories are:
 - *<per-user-home-directory>*
 - disabled [*<user-list>*]
 - enable [*<user-list>*]

EnvFile

- Type: String

- Range: Up to 255 characters
- When transferring environment variables before starting the server process, specify the path to the file where environment variables are configured.
- If EnvFile item is configured in the SVRGROUP section, the VHOST section configuration is ignored.
- Environment variables can be used. If EnvFile is a relative path, it is changed to an absolute path based on \$WEBTOBDIR.



The EnvFile item configuration in the NODE section affects the entire WebtoB system. However, the VHOST and SVRGROUP section configurations only affect server processes.

IndexName

- Type: Literal
- Range: Up to 255 characters
- Default value: "index.html"
- Specifies the default document name to be processed when a directory request comes in.

DirIndex

- Type: Literal
- Range: Up to 31 characters
- Specifies the DIRINDEX section name.



In order to use directory indexing function, "INDEX" must be added to Options item. The DIRINDEX section configuration can modify the directory indexing function operations.

Options

- Type: Literal
- Range: Up to 255 characters
- Default value: "HTML,CGI,SSI,PHP,JSV,USER"
- Specifies services and other operating methods provided by VHOST.
- If an option is not used, input a hyphen (-) preceding the option name.
- The following are descriptions of configurable options.

Option	Description
HTML	HTML services.
CGI	Uses the CGI server.
PHP	Uses the PHP server.
SSI	Uses the SSI server.
JSV	Uses the JSV (JEUS) server.
INDEX	When a client sends a request to a directory without specifying a file name, this option shows directory content. For example, for a "/docroot/dir/" request, the file name and information in the directory are sent to the client.
ALL	"HTML,CGI,PHP,SSI,JSV,INDEX", which is the same as the default value.

ErrorDocument

- Type: Literal
- Range: Up to 255 characters
- Specifies the use of a user specified page instead of the HTTP error page. Configure the names in the ERRORDOCUMENT section.
- Up to 64 items can be configured by separating each item with a comma (,).

Logging

- Type: Literal
- Range: Up to 255 characters
- Default value: <NODE section configuration>
- Specifies the LOGGING section name corresponding to the access log.
- Up to 4 items can be configured by separating each item with a comma (,).

ErrorLog

- Type: Literal
- Range: Up to 255 characters
- Default value: <NODE section configuration>
- Specifies the LOGGING section name corresponding to the error log.

Filter

- Type: Literal

- Range: Up to 255 characters
- Specifies the FILTER section name to be used.
- Up to 64 items can be configured by separating each item with a comma (,).

DefaultMimetype

- Type: Literal
- Range: Up to 127 characters
- Default value: <NODE section configuration>
- Specifies the Default Content-Type for documents that cannot decide the MIME-Type.



If DefaultMimetype is configured both in the DIRECTORY and SVRGROUP sections, the DIRECTORY's setting is primarily used, and then the SVRGROUP's. If neither is configured, the setting in the VHOST or NODE section is used.

Expires

- Type: Literal
- Range: Up to 255 characters
- Specifies the EXPIRES section name.
- Up to 64 items can be configured by separating each item with a comma (,).

ServiceOrder

- Type: Literal
- Range: Up to 255 characters
- Default value: <NODE section configuration>
- Specifies the priority of the URI and EXT sections when selecting the server and services to process the user request.
- The following are descriptions of configuration values.

Value	Description
uri,ext	Checks the URI section configuration first. If the configuration does not exist, check the EXT section configuration.
ext,uri	Checks the EXT section configuration first. If the configuration does not exist, check the URI section configuration.



If neither option is specified, the default HTML service processes the request.

Keepalive

- Type: Boolean
- Default value: <NODE section configuration>
- Specifies whether to use Keepalive (HTTP persistent connection).
- The following are descriptions of configuration values.

Value	Description
Y	User can reuse a connection so that a single connection can process multiple incoming requests.
N	User needs to reconnect to the socket for every single new request.

KeepaliveTimeout

- Type: Numeric
- Unit: Second
- Range: 1 to INT_MAX
- Default value: <NODE section configuration>
- Specifies the time period during which Keepalive remains active.
- After the user request has been processed, the connection is disconnected after the specified time.

KeepaliveMax

- Type: Numeric
- Range: 0 ~ INT_MAX
- Default value: <NODE section configuration>
- Specifies when to limit the Keepalive request count.
- If set to 0, request count is not limited.

Timeout

- Type: Numeric
- Unit: Second
- Range: 1 to INT_MAX
- Default value: <NODE section configuration>

- Specifies the timeout value when reading or writing data from the user connected socket.
- The configuration is applied when the user request is processed. If the user does not write to or read from the socket during the specified time, the socket is closed.

DefaultCharset

- Type: Literal
- Range: Up to 32 characters
- Specifies the default value of "charset=<value>" to add to the "Content-Type" item in HTTP Response Header.

Value	Description
On	"ISO-8859-1" is used as the default value.
Off	Charset is not added.
<custom-charset>	The specified "<custom-charset>" is used as the default value.



If DefaultCharset is configured both in the DIRECTORY and SVRGROUP sections, the DIRECTORY's setting is primarily used, and then the SVRGROUP's. If neither is configured, the setting in the VHOST or NODE section is used.

URLRewrite

- Type: Boolean
- Default value: N
- Specifies whether to use the URLRewrite function.



To use this function, specify the configuration file path in URLRewriteConfig.

URLRewriteConfig

- Type: Literal
- Range: Up to 255 characters
- Specifies the file path to use the URLRewrite function. Refer to [URLRewrite](#) for more information.

Headers

- Type: Literal
- Range: Up to 255 characters

- Specifies the HEADERS section name.
- Up to 15 items can be configured by separating each item with a comma (,).

CheckURL

- Type: Boolean
- Default value: N
- Indicates whether to convert the charset in the HTTP request URL path.
- The charset is a combination of CCS (Coded Character Set) and CES (Character Encoding Scheme). For Korean, EUC-KR or UTF-8 is generally used.
- If the charset used in the request URL path is different from that of the server, CheckURL can be used to convert the charset.
- If set to Y, CheckURLFrom and CheckURLTo need to be specified.
- If this option is also configured in the NODE section, the VHOST's configuration is primarily used.



Only "EUC-KR" and "UTF-8" are supported.

CheckURLFrom

- Type: Literal
- Range: Up to 31 characters
- Default value: "utf-8"
- Specifies the charset to be used in the HTTP request URL path.



Only "EUC-KR" and "UTF-8" are supported.

CheckURLTo

- Type: Literal
- Range: Up to 31 characters
- Specifies the charset to be used by the server.

CheckUrlJsvExcept

- Type: Boolean
- Default value: N
- Indicates whether to exempt requests forwarded to JEUS from the CheckURL execution.
- When set to Y, the URIs of requests sent to JEUS will remain unconverted.

3.4.2. Example

The following is an example configuration of the VHOST section.

```
*VHOST
webtob
  Docroot = "docs/vhost_docs",
  Hostname = "webtob.tmax.co.kr",
  Port="80",
  IndexName = "welcome.html",
  Logging = "webtob_access",
  ErrorLog = "webtob_error"
```

3.5. HTH_THREAD Section

The HTH_THREAD section configures threads of the HTH process. This section must be configured only once.

3.5.1. Configuration Items

The following is the environment configuration format of the HTH_THREAD section.

```
*HTH_THREAD
name # String[32]
  WorkerThreads = Numeric # 8 (1-100)
  #ReadBufSize = Numeric, # 1048576 (65536-)
  #SendfileThreads = Numeric, # 4 (0-100)
  #SendfileThreshold = Numeric, # 0 (0-)
  #AccessLogThread = Boolean, # Y
  #HtmlsCompression = Literal[256], # LIST[32], MULTI
  #HtmlsCompressionMinSize = Numeric, # 1 (1-),
  #Schedule = Literal[256] # "FA"
```



For more information about the symbols or other description rules for the configuration of the section and setting items, refer to [Configuration Item Description Rules](#).

HTH_THREAD Section Name

- Type: String
- Range: Up to 31 characters
- Specifies the HTH_THREAD name.

WorkerThreads

- Type: Numeric
- Range: 1 to 100
- Default value: 8
- Specifies the number of worker threads.
- Worker threads process static files, SSL/TLS (handshake, encryption, and decryption), compression, and HTTP authentication.

ReadBufSize

- Type: Numeric
- Unit: bytes
- Range: 65536 to INT_MAX
- Default value: 1048576
- Specifies the read buffer size used by worker threads to read static files.

SendfileThreads

- Type: Numeric
- Range: 0 to 100
- Default value: 4
- Specifies the number of sendfile threads.
- Sendfile threads are for the async sendfile function (in blocking mode), an OS-specific function, to process static files.



Valid only in UNIX and Linux.

SendFileThreshold

- Type: Numeric
- Unit: bytes
- Range: 0 to INT_MAX
- Default value: 0
- Specifies the file size threshold. If a requested file size is greater than the specified value, the sendfile function is used.
- If set to 0, the sendfile function is not used.



Valid only in UNIX and Linux.

AccessLogThread

- Type: Boolean
- Default value: Y
- Specifies the option to use a thread to record access log after HTH completes user request processing.
- If set to N, WSM records the access log.



Valid only in UNIX and Linux.

HtmlsCompression

- Type: Literal
- Range: Up to 255 characters
- Specifies the MIME-types (content-types) of files to compress in the response to the static files processed by a worker thread.
- The worker thread uses GZIP to compress the response before sending it to the client.
- Up to 32 MIME types delimited by commas can be set.
- Compression can reduce the network traffic but also the server performance. Files with low compression ratio, such as zip or jpeg, should not be compressed if possible to avoid server overhead.



Compression can only be used for requests with Accept-Encoding set to GZIP or deflate in the HTTP request header.

HtmlsCompressionMinSize

- Type: Numeric
- Range: 1 to INT_MAX
- Default value: 1
- Specifies the minimum size of the response to compress.
- If the requested file size is greater than the specified size, the response is compressed.

Schedule

- Type: Literal

- Range: RR | FA
- Default value: FA
- Specifies the mode to select a thread that will handle incoming requests when there are multiple worker threads.
- The following describes the setting values.

Value	Description
RR	Round Robin mode. In this mode, incoming requests are allocated to all worker threads sequentially.
FA	First Assign mode. In this mode, incoming requests are primarily allocated to the least loaded worker thread.

3.5.2. Example

The following is an example configuration of the HTH_THREAD section.

```
*HTH_THREAD
hworker
  WorkerThreads = 8,
  SendfileThreads = 4,
  SendfileThreshold = 32768,
  AccessLogThread = Y
```

3.6. SVRGROUP Section

When accessing an application server process through WebtoB, it is required to manage the server processes as a group according to their logical relationships. The SVRGROUP section configures the group.

SVRGROUP section defines the following:

- Name of the node that a server group belongs to.
- Type of service provided by a server group.

In addition, items defined in the NODE and VHOST sections can be defined again per server group. If the database is being used, database access information can be defined as well.

3.6.1. Configuration Items

The following is the environment configuration format of the SVRGROUP section.

```
*SVRGROUP
name # String[32]
  SvrType = String[32],
```

```

#VhostName = String[1024],          # LIST[64], MULTI
#AppDir = Literal[256],            # <*NODE/VHOST's>, $ENV, R.PATH
#UsrLogDir = Literal[256],         # $ENV, R.PATH
#EnvFile = String[256],           # $ENV, R.PATH
#AuthentName = String[32],
#Logging = Literal[256],          # LIST[4]
#ScriptLoc = Literal[256],
#ScriptArgs = Literal[256],
#Filter = Literal[256],           # LIST[64]
#DefaultMimetype = Literal[128],
#Expires = Literal[256],          # LIST[64]
#DefaultCharset = Literal[32],
#LBServers = Literal[256],         # LIST[32]
#LBType = String[16],             # Dynamic, Dynamic | Static
#LBBackup = Literal[32],
#Headers = Literal[256],          # LIST[15]
#UserAgentRegExp = String[512]

```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the SVRGROUP section configuration items.

SVRGROUP section name

- Type: String
- Range: Up to 31 characters
- Specifies the server group name.
- Each logical server group name must be unique in the SVRGROUP section.

SvrType (Required)

- Type: String
- Range: Up to 31 characters
- Specifies the server type supported by the server group.
- The following are descriptions of server type values.

Value	Description
HTML	Server type that services HTML requests.
CGI	Server type that services CGI requests.
PHP	Server type that services PHP requests.
SSI	Server type that services SSI requests.
JSV	Server type that integrates with JEUS and services JSP and servlet requests.
WEBSTD	Server type that services server requests written in WBAPI.

VhostName

- Type: Literal
- Range: Up to 1023 characters
- Specifies the VHOST section name.
- Up to 64 items can be configured by separating each with a comma (,).

AppDir

- Type: Literal
- Range: Up to 255 characters
- Default value: <Corresponding NODE or VHOST section AppDir value>
- Specifies the WBAPI application directory.
- Environment variables can be used, and if a relative path is used, it will be changed to an absolute path based on \$WEBTOBDIR.

UsrLogDir

- Type: Literal
- Range: Up to 255 characters
- Specifies the directory path of user logs.
- Related to the "-e <stderr-log-file> -o <stdout-log-file>" options of the Clopt configuration, the log is set as the name of the configuration path.



UsrLogDir uses the first configured value from among the SERVER, SVRGROUP, and NODE sections.

EnvFile

- Type: String
- Range: Up to 255 characters
- Specifies the file path of the environment variables used by WebtoB, when transferring environment variables before starting up the server process.
- Environment variables can be used, and if a relative path is used, it will be changed to an absolute path based on \$WEBTOBDIR.



The EnvFile item configuration of the NODE section affects the entire WebtoB system, but VHOST and SVRGROUP section configurations affect only server processes.

AuthentName

- Type: String
- Range: Up to 31 characters
- Specifies the AUTHENT section name.



The AuthentName configuration can configure the DIRECTORY, URI, EXT, and SVRGROUP sections. The configured values are applied in sequential order.

Logging

- Type: Literal
- Range: Up to 255 characters
- Specifies the LOGGING section name that corresponds to the access log.
- Up to 4 items can be configured by separating each with a comma (,).

ScriptLoc

- Type: Literal
- Range: Up to 255 characters
- Specifies the path for PHP execution module used to configure PHP-related server groups.
- Specifies a relative path based on \$WEBTOBDIR.



Specifies the path to "php-cgi" among PHP execution modules. In a UNIX/Linux environment, use a symbolic link to use the path under \$WEBTOBDIR.

ScriptArgs

- Type: Literal
- Range: Up to 255 characters
- Specifies the PHP execution module's parameters used to configure PHP-related server groups.
- For path options, use an absolute path as the set value is used directly as the parameter value.

Filter

- Type: Literal
- Range: Up to 255 characters
- Specifies the FILTER section name.

- Up to 64 items can be configured by separating each with a comma (,).

DefaultMimetype

- Type: Literal
- Range: Up to 127 characters
- Specifies the default content-type for documents whose MIME type cannot be decided.



If DefaultMimetype is configured both in the DIRECTORY and SVRGROUP sections, the DIRECTORY's setting is primarily used, and then the SVRGROUP's. If neither is configured, the setting in the VHOST or NODE section is used.

Expires

- Type: Literal
- Range: Up to 255 characters
- Specifies the EXPIRES section name.
- Up to 64 items can be configured by separating each with a comma (,).

DefaultCharset

- Type: Literal
- Range: Up to 32 characters
- Specifies the charset of the response when charset is not included in the request content-type.

Value	Description
On	"ISO-8859-1" is used as the default value.
Off	Charset is not added.
<custom-charset>	The specified "<custom-charset>" is used as the default value.



If DefaultCharset is configured both in the DIRECTORY and SVRGROUP sections, the DIRECTORY's setting is primarily used, and then the SVRGROUP's. If neither is configured, the setting in the VHOST or NODE section is used.

LBServers

- Type: Literal
- Range: Up to 255 characters

- Specifies the list of servers that will participate in load balancing.
- Up to 32 servers can be configured by separating each with a comma (,).

LBType

- Type: String
- Range: Up to 15 characters
- Default value: Dynamic
- Specifies the load balancing type for servers set in LBServers.

Value	Description
Dynamic	Distributes the load in round-robin style except when the number of requests waiting in a server queue is greater than the current server processes in the applicable server. If all server queues contain a request, distribute any new requests using the round-robin method.
Static	Distributes the load so that a server can handle requests in proportion to LBFactor of the SERVER section.

LBBackup

- Type: Literal
- Range: Up to 31 characters
- Specifies the backup server for the servers set in LBServers.

Headers

- Type: Literal
- Range: Up to 255 characters
- Specifies the HEADERS section name.
- Up to 15 names can be specified by using a comma(,) as the delimiter.

UserAgentRegExp

- Type: String
- Range: Up to 511
- JEUS server to process requests received via USER-AGENT.
- Requests received via this agent is processed by a server in the server group.

3.6.2. Example

The following is an example configuration of the SVRGROUP section.

```
*SVRGROUP
htmlg
  SvrType = HTML
phpg
  SvrType = PHP,
  ScriptLoc = "bin/php-cgi"
  ScriptArgs = "-c /php_conf/php.ini"
cgig
  SvrType = CGI
ssig
  SvrType = SSI
jsvg
  SvrType = JSV,
  LBServers = "jsv1, jsv2, jsv3",
  LBType = Static,
  LBBackup = "jsv4"
wbapg
  SvrType = WEBSTD
```

3.7. SERVER Section

The SERVER section configures the services provided by WebtoB. When a new server application is added, it must be registered in SERVER section of the environment file.

Most of the services can be registered in the SERVER section. Some services whose business logic is called directly by WebtoB must be registered in the SERVICE section. Servers are categorized in the service types (CGI, JSV, etc.) defined in the SERVER section. Additionally, the SERVER section defines a server group name and the maximum number of available processes.

Since all HTML services are processed by HTH worker threads, servers for HTML do not need to be configured.

3.7.1. Configuration Items

The following is the environment configuration format of the SERVER section.

```
*SERVER
name # String[32]
  SvgName = String[32],
  #CLOPT = Literal[256],
  #MinProc = Numeric, # 1 (1-)
  #MaxProc = Numeric, # <MinProc> (1-)
  #WSProc = Numeric, # 0 (0-)
  #UsrLogDir = Literal[256], # $ENV, R.PATH
  #MaxQCount = Numeric, # 0 (0-)
  #MaxQUrl = Literal[256],
  #MaxQUrlRedirectStatus = String[8],
```

```

#ASQCount = Numeric,           # 0 (0-)
#FlowControl = Numeric,        # 50 (1-)
#MaxRestart = Numeric,         # 20 (0-)
#SvrCPC = Numeric,             # 1 (1-)
#SvrType = String[32],
#HttpOutBufSize = Numeric,     # 8192 (0-)
#HttpInBufSize = Numeric,      # 8192 (0-)
#MaxRequests = Numeric,        # 0 (0-)
#SvrChkTime = Numeric,         # 0 (0-)
#Schedule = String[256],       # "RR"
#Options = Literal[256],       # PM.LIST
#SessionIdCookieKey = Literal[256],
#SessionIdUrlKey = Literal[256],
#FlexibleStickySessionRouting = Boolean, # N
#Compression = Literal[256],   # LIST[32], MULTI
#CompressionMinSize = Numeric, # 1 (1-)
#VhostName = String[1024],     # LIST[64], MULTI
#FcgiInitEnv = Literal[256],   # MULTI[32]
#FcgiKillTimeout = Numeric,    # 0 (0-)
#FcgiKillMaxRequest = Numeric, # 0 (0-)
#Headers = Literal[256],       # LIST[15]
#LBFactor = Numeric,           # 1 (1-1000)
#MaxJengineCount = Numeric,    # 32 (0-)
#RequestLevelPing = Boolean,   # N
#RequestLevelPingTimeout = Numeric, # 3 (0-)
#RequestLevelPingRetryCount = Numeric # 0 (0-)

```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the SERVER section configuration items.

SERVER section name

- Type: String
- Range: Up to 31 characters
- Specifies the unique server executable file name. A server name can be defined in the SERVER section only once. If duplicate names are defined, an error occurs during the environment file compilation.



If the server type is JSV, the server name can be up to 15 characters long. If the 15 character limit is exceeded, communication with JEUS is impossible.

SvgName (Required)

- Type: String
- Range: Up to 31 characters
- Specifies the server group name of the server.

- The specified server group must be also defined in the SVRGROUP section.
- This reference between a server and the SVRGROUP section allows WebtoB to determine the server's node location, type of resource manager (database) to be used, and parameters required when accessing the resource manager.

Clopt

- Type: Literal
- Range: Up to 255 characters
- Specifies the command options passed to the server process at startup.
- Server options are specified before '--', and user options are specified after '--'.

MinProc

- Type: Numeric
- Range: 1 to INT_MAX
- Default value: 1
- Specifies the number of server processes that starts at WebtoB startup.
- In the older server-client model, a single server process is created for each client. However, WebtoB has a more efficient design that fixes the number of server processes so that each server process can handle multiple client requests.
- MinProc specifies the number of server processes. It is recommended that the user sets the MinProc value based on previous experience. MinProc sets the minimum number of server processes in WebtoB, which is equivalent to the number of server processes WebtoB starts with.

MaxProc

- Type: Numeric
- Range: 1 to INT_MAX
- Default value: <MinProc>
- Specifies the maximum number of server processes that can be started.
- When WebtoB starts, the number of active server processes is equal to MinProc. As system load increases, the number of server processes automatically increases to MaxProc.
- Related to the "-e <stderr-log-file> -o <stdout-log-file>" options of the Clopt configuration, the log is set as the name of the configuration path.

WSProc

- Type: Numeric
- Range: 0 to INT_MAX

- Default value: 0
- Reverse connection setting between WebtoB and JEUS for a WebSocket request.
- Only for JSV servers.
- If this is not set, reverse connection cannot be used for WebSocket requests (reverse proxy can be used).
- This can be set per JEUS Managed Server that uses WebSocket.

UsrLogDir

- Type: Literal
- Range: Up to 255 characters
- Specifies the directory path of user logs.
- Related to the "-e <stderr-log-file> -o <stdout-log-file>" options of the Clopt configuration, the log is set as the name of the configuration path.



UsrLogDir uses the first configured value from among the SERVER, SVRGROUP, and NODE sections.

MaxQCount

- Type: Numeric
- Range: 0 to INT_MAX
- Default value: 0
- When a surge in client requests overloads a server and the server is no longer able to respond to new requests, it is necessary to ignore them. Once the number of client requests in the queue reaches a certain limit, any further incoming requests will not be queued. Instead, the server will respond to the client with an error.
- MaxQCount defines the limit for the number of requests that can be queued. This setting allows client requests to queue only up to the specified value. If the specified value is sufficient to accommodate all incoming requests, they can be handled without any issues. However, it's important to note that a larger value will result in slower response times.

MaxQUrl

- Type: Literal
- Range: Up to 255 characters
- Specifies the URL of the page that says that a server queue is full. If the server queue is full, a new request is not enqueued and a redirection response is returned. If the server that sends the response and the server that receives it are the same, redirection can occur infinitely.

MaxUrlRedirectStatus

- Type: String
- Range: Up to 7 characters
- Specifies the Redirect Status Code when MaxUrl is specified.
- The following values can be used.

Value	Alias	Description
301	permanent	Responds as "301 Moved Permanently".
302	found	Responds as "302 Found".
303	seeother	Responds as "303 See Other".
305	useproxy	Responds as "305 Use Proxy".
307	temp	Responds as "307 Temporary Redirect".
410	gone	Responds as "410 Gone".

ASQCount

- Type: Numeric
- Range: 0 to INT_MAX
- Default value: 0
- When the number of enqueued requests exceeds this threshold, a new server process is added automatically (if configured). The number of server processes increases from MinProc up to MaxProc.
- If set to 0, the number of server processes remains at MinProc.

FlowControl

- Type: Numeric
- Range: 1 to INT_MAX
- Default value: 50
- Specifies the unit used for flow control when configuring the buffer size for HTH to read a response from the server.

Flow control buffer size = FlowControl * HttpOutBufSize

- If set to 0, the default value is used.

MaxRestart

- Type: Numeric

- Range: 0 to INT_MAX
- Default value: 20
- Specifies the maximum number of times a server process can restart.

SvrCPC

- Type: Numeric
- Range: 1 to INT_MAX
- Default value: 1
- Specifies the number of channels for parallel communication between server process and HTH process.
- The parallel communication is used when there is too much load for the server process to handle with one channel.

HttpOutBufSize

- Type: Numeric
- Unit: bytes
- Range: 0 to INT_MAX
- Default value: 8192
- Specifies the buffer size used when the server responds to a user request.
- If the buffer size is set to 0 or greater than 16777216 (16 MB), the buffer is set to 16777216 (16 MB) for server processes. HTH uses the default size of 8192 needed to create the flow control buffer.

HttpInBufSize

- Type: Numeric
- Unit: bytes
- Range: 0 to INT_MAX
- Default value: 8192
- Specifies the buffer size used when the server receives user requests.
- If value is set to 0 or greater than 16777216 (16 MB), the value 16777216 (16 MB) is used.

MaxRequests

- Type: Numeric
- Range: 0 to INT_MAX

- Default value: 0 (0 means unlimited.)
- Specifies the maximum number of requests that can be processed by each server process. Each server process reboots automatically after processing this number of requests.
- Useful when AP has a memory related bug on a WEBSTD server written in WBAPI.
- If there are a large number of services running, a number of requests greater than MaxRequest may be processed before rebooting to provide seamless service.

SvrChkTime

- Type: Numeric
- Unit: Second
- Range: 0 ~ INT_MAX
- Default value: 0 (for JSV, 60)
- Specifies the interval at which a server connection is checked. SvrChkTime is often used to check the connection with JEUS if a firewall exists in between WebtoB and JEUS.
- After SvrChkTime has been set, the connection is in Ready state, and there are no service requests. If the target host does not respond to the keepAlive request for two consecutive SvrChkTime periods, a connection problem is declared. When the connection problem is found, the connection is disconnected and removed from the service distribution list.

Schedule

- Type: String
- Range: RR | FA
- Default value: RR
- Specifies the mode to select a server process that will handle client requests.
- The following describes the setting values.

Value	Description
RR	Round robin mode. In this mode, requests are allocated to idle processes in a round-robin manner.
FA	First Assign mode. In this mode, requests are allocated to an idle process with a high index.

Options

- Type: Literal
- Range: Up to 255 characters
- Specifies the options applied to the server:

Option	Description
{+ -}Cache	Option to cache the content. Valid only for HTML and JSV servers.
BlockListen	<p>Blocks the port to prevent client access when the server cannot provide services. Once the server is restored, the ports begin accepting clients, and processes new client requests.</p> <p>In a multi-node system using L4 equipment, blocking the ports of problematic nodes enables L4 equipment to distribute requests to available nodes.</p> <p>When multiple virtual hosts use the same service port, the virtual hosts are split among the available service ports. Blocking a service port prevents it and related virtual hosts from providing services.</p>
BlockListenCloseClients	<p>BlockListenCloseClients should be specified when clients that are linked to ports are supposed to be closed while closing service ports with the BlockListen option.</p> <p>Disconnects a client whose request, which is being processed, completes. A client that maintains a connection with keepalive is immediately disconnected.</p>
NotifyClientClose	Specifies whether to notify the server when a user disconnects while a request is processing. JEUS disconnects the connection that was processing the request and notifies the server.
SecurityHolePassAuthorization	<p>Passes authorization or proxy-authorization header to the CGI/PHP environment variable.</p> <p>Caution: User information can be exposed to CGI/PHP.</p>
503ResponseOnSuspend	<p>Responds with the "503 Service Temporarily Unavailable" message to requests if the server cannot provide the service due to the suspend command.</p> <p>If this option is not used in the case mentioned above, requests that the server must process are queued.</p>
PassOriginalUriAfterFilters	<p>Sends the original client request when forwarding the request to JSV server (or another server process) after the FILTERS process handles the request.</p> <p>This option should be used by considering that the request URL can be modified during FILTERS processing due to URL encoding, etc.</p>
AllServers	Option to use FILTERS process to process Filter module for an HTMLS request.

SessionIdCookieKey

- Type: Literal

- Range: Up to 255 characters
- Default value: "JSESSIONID"
- Specifies the HTTP cookie key name used for session routing when the server type is JSV (JEUS).

SessionIdUrlKey

- Type: Literal
- Range: Up to 255 characters
- Specifies the HTTP URL key name used for session routing when the server type is JSV (JEUS).
- If SessionIdCookieKey and SessionIdUrlKey are both configured, the SessionIdCookieKey setting takes precedence.

FlexibleStickySessionRouting

- Type: Boolean
- Default value: N
- When the server type is JSV, session routing occurs, and this specifies whether to use flexible routing.
- The following table describes the setting values.

Value	Description
Y	When server processes with the same jengineid are in the RUN state, the sessions are not queued. The sessions are routed to another process with a different jengineid within the same server.
N (default value)	Sticky Session routing is used (the same as the previous version). Sessions are routed to the server processes with the same jengineid. If the server processes are in the RUN state, the sessions are queued.



When using flexible routing, multiple clients with the same JSESSIONID can be each routed to different servers. It is recommended to use the default value.

Compression

- Type: Literal
- Range: Up to 255 characters
- Specifies the response content-type to compress.
- Specifies the MIME-Type to be compressed. Content-Type of the response is used. The target response is compressed with GZIP then sent to the client.
- Up to 32 MIME-Types can be configured by separating each with a comma (,).

- Compression allows network traffic to be reduced, but server performance may be affected.

Avoid compressing files with low compression rates, such as .zip files or .jpeg images, as this places unnecessary load on the server without providing any benefits.



The compression function is applied only to the requests where Accept-Encoding is specified as GZIP or deflate in the HTTP Request Header.

CompressionMinSize

- Type: Numeric
- Range: 1 to INT_MAX
- Default value: 1
- Specifies the minimum response size when the compression option is set.
- If the Content-Length of the response header is greater than the set size, the response is compressed. This does not apply to a chunked response whose size is difficult to determine.

VhostName

- Type: Literal
- Range: Up to 1,023 characters
- Specifies the virtual host name when using the server as a virtual host.
- A virtual host name can also be set for a server group. If the name is set for both the server and server group, they must be identical.
- Up to 64 items can be configured by separating each with a comma (,).

FcgiInitEnv

- Type: Literal
- Range: Up to 255 characters
- Adds the environment variables required to execute FastCGI applications. Configuration format is NAME=VALUE.

```
FcgiInitEnv = "LOGFILE=/wb-413/log/myapp.log"
```

```
# myapp.log example  
FAST_CGI_ENV_TEST="sample fast cgi environment variable"
```

- Multiple environment variables can be configured by using one or more FcgiInitEnv statements.

FcgiKillTimeout

- Type: Numeric
- Unit: Second
- Range: 0 to INT_MAX
- Default value: 0 (0 means unlimited.)
- Specifies whether to kill the application process after a FastCGI request is processed.
- If the time since the application process began is greater than the time specified in FcgiKillTimeOut, the server forcibly terminates the application process after the request is processed.

FcgiKillMaxRequest

- Type: Numeric
- Range: 0 to INT_MAX
- Default value: 0 (0 means unlimited.)
- Specifies whether to terminate the application process after a FastCGI request is processed.
- If the number of requests processed since the start of the application process is greater than the value specified in FcgiKillMaxRequest, after the server processes the request, the server forcibly terminates the application process.

Headers

- Type: Literal
- Range: Up to 255 characters
- Specifies the HEADERS section name.
- Up to 15 items can be configured by separating each with a comma (,).

LBFactor

- Type: Numeric
- Range: 1 to 1000
- Default value: 1
- Specifies the ratio the request is to be processed with when LBType is static for servers that are set to LBServers in the SVRGROUP section.
- The value can be dynamically changed using the `set -v <server_name> lbf <value>` command in wsadmin.

MaxJengineCount

- Type: Numeric
- Range: 0 to INT_MAX
- Default value: 32
- The default value is 32 for JSV servers and 0 for all others.
- Specifies the maximum allowed number of Jengines per server.

RequestLevelPing

- Type: Boolean
- Default value: N
- Specifies whether to send a ping before forwarding a request when the target server type is JSV.
- If set to Y, a request is only forwarded to JSV server after sending a ping and receiving a response.
- This is useful when connected to multiple JSV servers and a server may fail to respond due to reasons like OOM (Out Of Memory).



This may cause performance issues as a ping is sent and a response is received for all requests. It is recommended to use the default value.

RequestLevelPingTimeout

- Type: Numeric
- Range: 0 to INT_MAX
- Default value: 3
- Waiting time for receiving a response for a ping sent to a scheduled JSV server. If no response is received during the waiting time, the connection is terminated and another JSV server is scheduled and pinged.

RequestLevelPingRetryCount

- Type: Numeric
- Range: 0 to INT_MAX
- Default value: 0
- Specifies the retry count for sending another round of pings to all Jengines in the JSV server.

3.7.2. Example

The following is an example configuration of the SERVER section.

```

*SERVER
cgi
  SvgName = cgig,
  MinProc = 1,
  MaxProc = 5
ssi
  SvgName = ssig,
  MinProc = 1,
  MaxProc = 2
jsv1
  SvgName = jsvg,
  MinProc = 1,
  MAXProc = 10,
  LBFactor = 10
jsv2
  SvgName = jsvg,
  MinProc = 1,
  MAXProc = 10
  LBFactor = 5
jsv3
  SvgName = jsvg,
  MinProc = 1,
  MAXProc = 10
  LBFactor = 1
jsv4
  SvgName = jsvg,
  MinProc = 1,
  MAXProc = 10
wbaps
  SvgName = wbapg,
  MinProc = 2,
  MAXProc = 5,
  Schedule = "FA"

```



SvgName is jsvg. jsv1, jsv2, and jsv3 set to LBServers in the *SVRGROUP section run with the load balancing server and distribute requests in a ratio of 10, 5, and 1(LBFactor is meaningful only when LBType is static). jsv4 (LBBackup) processes requests when jsv1, jsv2, and jsv3 are not available.

3.8. SERVICE Section

The SERVICE section needs to be configured only when WebtoB directly executes business logic. In this situation, the server type of the service is often set to WEBSTD and standard CGI functions are substituted with equivalent WebtoB functions. Finally, transformed business logic is executable on WebtoB as a new service.

The following items are configurable in the SERVICE section:

- Server processes that provides a service.
- Service priority order.

- Service processing time limit.

3.8.1. Configuration Items

The following is the environment configuration format of the SERVICE section.

```
*SERVICE
name # String[16]
    SvrName = String[32],
    #Priority = Numeric,           # 50 (0-100)
    #SvcTime = Numeric           # 0 (0-)
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the SERVICE section configuration items.

SERVICE section name

- Type: String
- Range: Up to 15 characters
- Specifies the service routine name of the server program that implements the business logic executed by WebtoB.
- The name must be unique in the SERVICE section, and can be up to 15 characters.

SvrName (Required)

- Type: String
- Range: Up to 31 characters
- Specifies the name of the server process that provides the service. Specifies the executable file name of the server program that has the service routine.
- Server process must be defined in the SERVER section.

Priority

- Type: Numeric
- Range: 0 to 100
- Default value: 50
- Specifies the client request priority between 1 (the lowest) and 100 (the highest).

SvcTime

- Type: Numeric
- Unit: Second
- Range: 0 to INT_MAX
- Default value: 0 (0 means unlimited.)
- Specifies the maximum period of time to process a service. Once a service starts, it must finish within the specified time. If the service cannot finish its processing within the time limit, the server process stops the service and sends an error response to the client.

3.8.2. Example

The following is an example configuration of the SERVICE section.

```
*SERVICE
example
    SvrName = webaps,
write_board
    SvrName = webaps
```

3.9. DIRECTORY Section

The DIRECTORY section configures a directory of a node. Configurable items are AuthentName, which grants access to a directory; ForceMimetype, which sets file extension to the files in a directory; and DefaultMimetype. Moreover, directory access history logs can be configured.

3.9.1. Configuration Items

The following is the environment configuration format of the DIRECTORY section.

```
*DIRECTORY
name # String[32]
    Directory = Literal[256], # $ENV, R.PATH
    #DefaultMimetype = Literal[128],
    #ForceMimetype = Literal[128],
    #VhostName = String[1024], # LIST[64], MULTI
    #AccessName = String[32],
    #AuthentName = String[32],
    #Options = Literal[256], # PM.LIST
    #DefaultCharset = Literal[32],
    #ErrorDocument = Literal[256] # LIST[64]
```



Refer to [Configuration Item Description Rules](#) for more information on symbols

and details of the DIRECTORY section configuration items.

DIRECTORY section name

- Type: String
- Range: Up to 31 characters
- Specifies the directory name.

Directory (Required)

- Type: Literal
- Range: Up to 255 characters
- Specifies the directory path. The path can be either an absolute path or a relative path from \$WEBTOBDIR.

DefaultMimetype

- Type: Literal
- Range: Up to 127 characters
- Specifies the default content-type for documents whose MIME type cannot be determined.
- The following is the order of priority applied to various sections.
 1. DefaultMimetype configured in the DIRECTORY section.
 2. DefaultMimetype configured in the SVRGROUP section.
 3. DefaultMimetype configured in the NODE or VHOST section.

ForceMimetype

- Type: Literal
- Range: Up to 127 characters
- Specifies the default MIME type for all resources in the specified directory.

For example, if ForceMimetype is set to CGI, all resources in the directory process client requests as CGI, which means that the server in charge of CGI handles the services.

VhostName

- Type: Literal
- Range: Up to 1023 characters
- Specifies the virtual host name. Configurations in the DIRECTORY section are applied only to the

specified virtual host.

- Up to 64 elements can be configured by separating each element with a comma (,).
- If the VhostName item is not set, this item can be applied to all Virtual Hosts.

AccessName

- Type: String
- Range: Up to 31 characters
- Specifies the ACCESS section name. Accesses from certain IPs can be restricted according to this setting.

AuthentName

- Type: String
- Range: Up to 31 characters
- Specifies the name of AUTHENT section to apply to the directory. If the server type is JSV, authentication can be applied only to the SVRGROUP section.

Options

- Type: Literal
- Range: Up to 255 characters
- The following are descriptions of configurable options.

Option	Description
{+ -}Cache	Enables or disables the cache (+ -) of the selected content
SSLRequireSSL	Processes SSL requests only. Responds to non-SSL requests with the 403 Forbidden message.
SSLDenySSL	Processes non-SSL requests only. Responds to SSL requests with the 403 Forbidden message.

DefaultCharset

- Type: Literal
- Range: Up to 31 characters
- Specifies the charset of the response when charset is not included in the request content-type.

Value	Description
On	"ISO-8859-1" is specified as the default value.
Off	Charset is not added.

Value	Description
charset	User specified _charset_ is used.

- The following is the order of priority applied to various sections.
 1. DefaultCharset configured in the DIRECTORY section.
 2. DefaultCharset configured in the SVRGROUP section.
 3. DefaultCharset configured in the NODE or VHOST section.

ErrorDocument

- Type: Literal
- Range: Up to 255 characters
- Specifies the use of a user specified page instead of the HTTP error page. Configures the names in the ERRORDOCUMENT section.
- Up to 64 items can be configured by separating each item with a comma (,).

3.9.2. Example

The following is an example configuration of the DIRECTORY section.

```
*DIRECTORY
dir_test
    DIRECTORY = "/usr/local/webtob/docs/vhost_docs",
    ForceMimetype = "text/plain"
```

3.10. URI Section

The URI section is used to provide an appropriate service according to the Uniform Resource Identifier (URI) of a client request. This section is often used for CGI services.

For example, if the address <http://www.tmax.co.kr/cgi-bin/test.cgi> is requested, the "/cgi-bin/" URI can be configured and used as a CGI service.

In the following example, the URI section must be configured for <first> or a URI that starts with <first> must be used.

```
http://<hostname>:<port>/<first>/<second>/index.html
```


3.10.1. Configuration Items

The following is the environment configuration format of the URI section.

```
*URI
name # String[32]
    URI = Literal[256],
    SvrType = String[32],
    #SvrName = String[32],
    #SvcName = String[16],
    #Redirect = Literal[256],
    #RedirectStatus = String[32], # 302
    #VhostName = String[1024], # LIST[64], MULTI
    #AccessName = String[32],
    #AuthentName = String[32],
    #Options = Literal[256], # PM.LIST
    #SCGI = Boolean, # N
    #SCGIServer = Literal[256],
    #Match = Literal[256], # "prefix"
    #Priority = Numeric, # 50 (0-100)
    #FCGI = Boolean, # N
    #RedirectNoSub = Boolean, # N
    #Ext = String[256], # LIST, MULTI
    #GotoEXT = Boolean, # N
    #StopIfNoEXT = Boolean, # Y
    #LBSvgName = Literal[256]
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of URI section configuration items.

URI section name

- Type: String
- Range: Up to 31 characters
- The URI section name string can hold any character.

Uri (Required)

- Type: Literal
- Range: Up to 255 characters
- Specifies the pattern that matches the HTTP request path. Once matched, the URI section configuration is applied to the request.
- WebtoB supports a variety of pattern types. Pattern type is configured in the Match item.

SvrType (Required)

- Type: String
- Range: Up to 31 characters
- Specifies the server type that processes the request that includes the specified URI. For example, it specifies the JSV server type for a server that processes a request containing the URI `"/jsv/"`.

SvrName

- Type: String
- Range: Up to 31 characters
- Specifies the server name of the service.
- In WebtoB, service objects that have the same server type are called a server group. Each server in a server group has a minimum and maximum number of processes allocated for service.
- The URI section allows micro management service control, which can assign a server for a service.

SvcName

- Type: String
- Range: Up to 15 characters
- Specifies the service name that processes the URI.

Redirect

- Type: String
- Range: Up to 255 characters
- Maps a request for the specified URI to another URI. Depending on the value specified in `RedirectStatus`, the value specified in `Redirect` is specified in the Location Header item of the HTTP response to be sent to user.
- If the value of `RedirectStatus` is omitted and only `Redirect` is used, `RedirectStatus` is set to "302 Found".

RedirectStatus

- Type: String
- Range: Up to 31 characters
- Default value: 302
- Specifies the HTTP status generated when using redirection.
- The following values can be used.

Value	Alias	Description
301	permanent	Responds with "301 Moved Permanently".
302	found	Responds with "302 Found".
303	seeother	Responds with "303 See Other".
305	useproxy	Responds with "305 Use Proxy".
307	temp	Responds with "307 Temporary Redirect".
410	gone	Responds with "410 Gone".

VhostName

- Type: Literal
- Range: Up to 1,023 characters
- Specifies the virtual host name for a URI or each server types that share the same URI.
- Up to 64 items can be set to the same URI by using a comma as a delimiter.

AccessName

- Type: String
- Range: Up to 31 characters
- Specifies in the ACCESS section whether to allow or deny a request coming from a specific IP address.

AuthentName

- Type: String
- Range: Up to 31 characters
- Enables authentication of the URI section. Specifies the name of authentication applied to the URI section. However, if the server type is JSV, authentication can be applied in the SVRGROUP section.

Options

- Type: Literal
- Range: Up to 255 characters
- The following are descriptions of configurable options.

Option	Description
{+ -}Cache	Enables or disables the cache (+ -) of the selected content
SSLRequireSSL	Processes SSL requests only. Responds to non-SSL requests with the "403 Forbidden" message.

Option	Description
SSLDenySSL	Processes non-SSL requests only. Responds to SSL requests with the "403 Forbidden" message.

SCGI

- Type: Boolean
- Default value: N
- Specifies whether the request is SCGI when SvrType is set to CGI.
- If set to Y, the SCGI server address is also required.

SCGIServer

- Type: Literal
- Range: Up to 255 characters
- Specifies the SCGI server address in the "server_name[IP_address]:port_number" format.

```
SCGIServer = "172.16.1.100:9001"
```

Match

- Type: Literal
- Range: Up to 255 characters
- Default value: "prefix"
- Specifies the matching pattern type. A pattern set in the Uri item is used as the prefix for the request URL by default.

Value	Description
prefix	The pattern specified in the URI is the HTTP Request URL's prefix. For example, pattern "/uri/" is matched with the request path "/uri/a/", "/uri/a/b/", "/uri/a/b/c", etc.
exact	The pattern specified in the URI matches the HTTP Request URL. For example, pattern "/uri/" only matches when the Request path is "/uri/". Other request paths do not match.
fn	The pattern specified in the URI is the FNMATCH pattern used in a UNIX shell. Wild card such as an asterisk (*) and a question mark (?) can be used. For example, pattern "/uri/*/*/" is matched with the request path "/uri/a/b/", "/uri/c/d/", etc.

Value	Description
regex	The pattern specified in the URI is the regular expression used in Perl. For example, pattern <code>"/uri.*"</code> is matched with the request path <code>"/uri"</code> , <code>"/uri/"</code> , <code>"/uri1"</code> , etc.

- Depending on the pattern type, the matching method with the HTTP Request path differs.

Priority

- Type: Numeric
- Range: 0 to 100
- Default value: 50
- Prioritizes queued client requests. Priority is between 1 (lowest) and 100 (highest).

FCGI

- Type: Boolean
- Default value: N
- Specifies whether the request is FastCGI when `SvrType` is set to CGI.
- If set to Y, the request is processed as Fast CGI.

RedirectNoSub

- Type: Boolean
- Default value: N
- When configuring redirect, if prefix is used to match, specifies whether to attach the rest of the request URL to the specified URL and then redirect.

Ext

- Type: Literal
- Range: Up to 255 characters
- When matching a URI, match to the additionally configured extension.
- Use this item when applying the URI section to certain extensions under a certain path.
- Multiple extensions can be configured by using commas (,) or configuring multiple times.

GotoExt

- Type: Boolean

- Default value: N
- Decides whether to match the EXT section additionally when performing the URI matching first. Note that if an EXT match is found and selected the previously matched URI setting is ignored.

StopIfNoExt

- Type: Boolean
- Default value: Y
- Applied when 'GotoExt = Y'.
- The following describes the values.

Value	Description
Y	Stops matching when the matched EXT section configuration item does not exist.
N	Searches the next URI section configuration.

LBSvgName

- Type: Literal
- Range: Up to 255 characters
- Specifies the name of a SVRGROUP for load balancing between servers that will handle incoming requests with a pre-specified URI.
- The specified SVRGROUP must contain the LBServers or LBBackup option.

3.10.2. Example

The following is an example configuration of the URI section.

```
*URI
uri1
  Uri = "/cgi-bin/",
  SvrType = CGI
uri2
  Uri = "/cgi/",
  SvrType = CGI
uri3
  Uri = "/test/",
  SvrType = CGI
uri4
  Uri = "/jsv/",
  SvrType = JSV
```

3.11. ALIAS Section

The ALIAS section maps a URI to a physical directory path on the server regardless of the document root. Aliasing can map a directory regardless of the Document Root, which is convenient from a management perspective.

3.11.1. Configuration Items

The following is the environment configuration format of the ALIAS section.

```
*ALIAS
name      # String[32]
          URI = Literal[256],
          RealPath = Literal[256],          # $ENV, R.PATH
          #VhostName = String[1024]        # LIST[64], MULTI
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the ALIAS section configuration items.

ALIAS section name

- Type: String
- Range: Up to 31 characters
- Specifies the ALIAS section name.

URI (Required)

- Type: Literal
- Range: Up to 255 characters
- Specifies the URI alias.

Realpath (Required)

- Type: Literal
- Range: Up to 255 characters
- Specifies the physical directory path on the server.
- When using a large number of virtual hosts, the directory configuration pattern can be configured dynamically by using the following directives.

Directive	Description
%p	Replaces with the requested port number.
%n	Replaces with the Hostname or the Nth element of the IP address. If n is set to 0, the entire string is used. If the value starts with a minus symbol (-), count from the end of Hostname or IP address. If a plus symbol (+) follows, the rest of the Hostname or IP address is used.
%n.m	Replaces with the Mth character of the Nth element. Minus (-) or plus (+) symbols can be used as previously described.
%%	Replaces with a single percent symbol (%).

VhostName

- Type: Literal
- Range: Up to 1023 characters
- Specifies the maximum number of virtual hosts that apply to the ALIAS section.
- Up to 64 items can be configured by separating each with a comma (,).

3.11.2. Example

The following is an example configuration of the ALIAS section.

```
*ALIAS
alias1
    URI = "/cgi-bin/",
    RealPath = "/usr/local/webtob/cgi-bin/"
alias2
    URI = "/tpsvc/",
    RealPath = "/usr/local/webtob/ap/"
```

3.12. DIRINDEX Section

In general, when a client request URL specifies a directory, the directory's index.html file is returned. However, if the index.html file is missing, a directory listing page can be shown. This section can enable or disable the directory listing feature and configures the indexing mechanism and icons used in the directory listing.

3.12.1. Configuration Items

The following is the environment configuration format of the DIRINDEX section.


```
*DIRINDEX
name # String[32]
  Options = Literal[256], # "Fancy,EncodeURL", PM.LIST
  #Ignore = Literal[256], # LIST
  #DefaultIcon = String[32],
  #Description = Literal[256],
  #HeaderFile = String[32],
  #TailFile = String[32],
  #IconExt = Literal[256]
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the DIRINDEX section configuration items.

To use the DIRINDEX section, some items must be first configured in the DIRINDEX section. The required previously configured items are as follows:

Item	Description
Options	"+Index" displays directory information.
DirIndex	Specifies the DIRINDEX name.

DIRINDEX section name

- Type: String
- Range: Up to 31 characters
- Specifies the DIRINDEX section name.

Options (Required)

- Type: Literal
- Options can be used to specify the indexing method.
- The following describes the options.

Option	Description
{+ -}Fancy	Option to use Fancy Indexing. Fancy Indexing, provided by WebtoB, displays links sorted by the file name, last modified time, or file size.
{+ -}EncodeURL	Option to use URL encoding for non-ASCII characters used for URLs on the page. If set to "-EncodeURL", Korean file links are displayed in Korean. The file may not be found depending on browser settings.

Ignore

- Type: Literal
- Range: Up to 255 characters
- Specifies the files to exclude from the index list. Filenames can be explicitly specified or an asterisk(*) can be used to filter files.

DefaultIcon

- Type: String
- Range: Up to 31 characters
- Specifies the icon for unknown file types.

Description

- Type: Literal
- Range: Up to 255 characters
- Configures the description file path. This configuration shows the description of each file when the directory structure is displayed.

For example, when a.html, b.html, c.html, etc. exist in the directory, write the contents of the des.txt file as follows to display the description. Configure as Description = "des.txt".

```
a.html Description of a.html  
b.html Description of b.html  
c.html Description of c.html  
...
```

HeaderFile

- Type: String
- Range: Up to 31 characters
- Specifies the header file name that will be inserted at the top of the listing. Filename uses a relative path from the indexing directory.

TailFile

- Type: String
- Range: Up to 31 characters
- Specifies the tail file name that will be inserted at the bottom of the listing. Filename uses a relative path from the indexing directory.

IconExt

- Type: Literal
- Range: Up to 255 characters
- Specifies the file extension to map to icon file URL.
- The MIME-Type and the URL of the icon file are mapped.

3.12.2. Example

The following is an example configuration of the DIRINDEX section.

```
*DIRINDEX
dindex
  Options = "Fancy"
```

3.13. LOGGING Section

The LOGGING section configures client request log. Access log and error log are recorded separately, and their format can be configured. To use the LOGGING section, the Logging and ErrorLog items must be set in the NODE section.

3.13.1. Configuration Items

The following is the environment configuration format of the LOGGING section.

```
*LOGGING
name # String[32]
  FileName = Literal[256], # $ENV, R.PATH
  Format = Literal[256],
#Option = Literal[256], # PM.LIST
#RotateBySeconds = Numeric, # 0 (0-)
#ExcludeByExt = Literal[1024], # LIST
#ArchiveFileName = Literal[256], # $ENV
#ValidHours = Numeric, # 0 (0-23)
#LogHandler = Literal[256] # LIST[5]
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the LOGGING configuration items.

The following are the NODE section items that are related to the LOGGING section configuration. The Logging and Errorlog items (not Syslog) can be specified in both the VHOST and NODE sections.

Item	Description
SysLog	Specifies system log name.
Logging	Specifies access log name.
Errorlog	Specifies error log name.

LOGGING section name

- Type: String
- Range: Up to 31 characters
- Specifies the LOGGING section name.

FileName (Required)

- Type: Literal
- Range: Up to 255 characters
- Specifies the log file path and name.
- A relative path that starts without a slash (/) is automatically replaced by "WEBTOBDIR/relative_path".
- The substitution string specified in FileName is replaced with the actual value when the file is created. For example, the value configured in FileName is replaced with "\$WEBTOBDIR/log/access_20091105.log".

```
FileName = "log/access_%Y%M%D%.log"
```

- The following strings in a file name are replaced by actual values when the file is created:

String	Actual Value
%Y%	4-digit year. (Example: 2009)
%M%	2-digit month. (Example: 11)
%D%	2-digit day. (Example: 05)
%h%	2-digit hour. (Example: 10)
%m%	2-digit minute. (Example: 30)
%s%	2-digit second. (Example: 45)

Format (Required)

- Type: Literal
- Range: Up to 255 characters

- Specifies the log format
- This value can be set to the name of the log format set in the *USERLOGFORMAT section.
- The following format string is applied only to the access log. System log and error log can have any value.

Format	Description
DEFAULT	Default log file format. (Format string: "%h %t \"%r\" %s %b %D")
COMMON	Common log file format. (Format string: "%h %l %u %t \"%r\" %s %b")
COMBINED	Combined log file format. (Format string: "%h %l %u %t \"%r\" %s %b \"%{Referer}i\" \"%{User-Agent}i\"")
COMBINEDIO	CombinedIO log file format. (Format string: "%h %l %u %t \"%r\" %s %b \"%{Referer}i\" \"%{User-Agent}i\" %I %O ")
%a	Displays the IP address of the machine that sent the request. Same as %h.
%b	Displays the byte of the response object excluding the header.
%c	Displays the location where a WebtoB response is generated. <ul style="list-style-type: none"> • Set to "hc" if a response is generated in the internal cache. • Set to "dc" if a response is generated in the disk cache. • Set to "sf" if a response is generated in a sendfile. • Set to "sf/dc" if a response is generated in a sendfile/disk cache. • Set to "hm" if a response is generated in a remote host.
%{_attr_name_}C	Displays the value corresponding to '_attr_name_' among cookie Header values of an HTTP request.
%d	Displays the time the response was sent.
%D	Displays the time spent processing the request. (unit: millisecond)
%{ENV_NAME}e	Displays the environment variable ENV_NAME.
%g	Displays the request identifier used internally by WebtoB.
%h	Displays the IP address of the machine that sent the request. Same as %a.
%H	Displays the HTTP version used.
%{HEADER_FIELD}i	Displays the HEADER_FIELD Header value of the HTTP request.

Format	Description
%{_id_name_}j	Displays the request id used internally when a request is forwarded to JEUS for processing. <ul style="list-style-type: none"> • If _id_name is JSVCid, then this is the Client ID. • If _id_name is JSVReqSeq, then this is the Request Sequence.
%l	Displays the name for remote login.
%m	Displays the HTTP request method.
%p	Displays the server port number that received the request.
%q	Displays the query value of the HTTP request.
%r	Displays the entire request line of the HTTP request.
%R	Displays the entire request line of the HTTP request. Request lines that were changed by the CheckURL or URLRewrite function are displayed.
%s	Displays the HTTP Status Code used in the response.
%t	Displays the time when the request processing ended.
%T	Displays the time spent in processing the request. (unit: second)
%u	Displays the user name used in the HTTP authentication.
%U	Displays the HTTP request URI.
%v	Displays the Host Header field value.
%z	If the response is compressed, displays the response size and compression rate before and after the compression.
%S	Separates HTTP and HTTPS.
%A	Displays server IP address.
%I	Displays the byte of the request.
%O	Displays the byte of the response.

Option

- Type: Literal
- Range: Up to 255 characters
- Specifies the logging options.

Option	Description
Sync	Does not buffer logging content and records directly onto the disk. Services that require immediate logs, such as finance service and banking service, should use this feature for immediate error identification.
NoErrorClientAddress	Excludes client IP addresses from error log messages. This ensures security by avoiding exposure of user information.

RotateBySeconds

- Type: Numeric
- Unit: Second
- Range: 0 to INT_MAX
- Default value: 0 (The default value disables this function.)
- Specifies the maximum period of time to create a new log file. An existing log file is older than this period of time, a new log file is created to record a new log.

ExcludeByExt

- Type: Literal
- Range: Up to 1,023 characters
- Specifies the file extension to exclude from logging.
- The following is an example configuration.

```
ExcludeByExt = "jpg,png"
```

ArchiveFileName

- Type: Literal
- Range: Up to 255 characters
- Specifies the log file name format for an archived file. New log files will be named in the format (file name + date) specified with the FileName setting. Archived files have the same name format as log files.

ValidHours

- Type: Numeric
- Unit: hours
- Range: 0 to 23
- Default value: 0 (0 disables this function.)
- Specifies the creation of a new log file at the specified interval.

LogHandler

- Type: Literal
- Range: Up to 255
- Specifies the name of a LOG_HANDLER section to use for access logs in a remote server.

- Up to five items can be set by separating each with a comma (,).

3.13.2. Example

The following is an example configuration of the LOGGING section.

```
*LOGGING
access_log
    Format = "default", # "%h %l %u %t \"%r\" %s %b" is default format
    Filename = "log/archives/access_%Y%M%D.log"
    Option = "sync"
error_log
    Format = "%r Host=%{HOST}i",
    Filename = "log/archives/error_%Y%M%D.log"
    Option = "sync"
system_log
    Format = "",
    Filename = "log/archives/system_%Y%M%D.log"
    Option = "sync"
```

3.14. ACCESS Section

The ACCESS section configures client accesses based on IP address and netmask information and the order of applying configurations. This section can be applied to the URI, EXT, DIRECTORY, and TCPGW sections. The defined resources from each of these sections are accepted or denied.

3.14.1. Configuration Items

The following is the configuration format of the ACCESS section.

```
*ACCESS
name # String[32]
    #Method = Literal[256], # "<all-methods>", PM.LIST
    #MethodExcept = Literal[256], # PM.LIST
    #Order = Literal[256], # "deny,allow"
    #Allow = Literal[256*35], # LIST[256]
    #Deny = Literal[256*35], # LIST[256]
    #AllowIf = Literal[256], # MULTI[32]
    #DenyIf = Literal[256] # MULTI[32]
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the ACCESS section configuration items.

ACCESS section name

- Type: String
- Range: Up to 31 characters
- Specifies the ACCESS section name.

Method

- Type: Literal
- Range: Up to 255 characters
- Default value: <all-HTTP-methods>
- Specifies the HTTP method to use.

MethodExcept

- Type: Literal
- Range: Up to 255 characters
- Specifies the methods to be excluded from the HTTP methods list (GET, POST, HEAD, OPTIONS, PUT, DELETE, TRACE, PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, UNLOCK) that can be configured in WebtoB. I.e., only the methods not specified by MethodExcept can be applied.
- For example, MethodExcept = "POST" is same as Method = "GET, HEAD, ..." (all HTTP methods except POST).

Order

- Type: Literal
- Default value: "Deny,Allow"
- Specifies the order of the following access restriction directives: Method, Allow, AllowIf, Deny, and DenyIf.
- The following are descriptions of the options.

Option	Description
Deny, Allow	<p>Executes matching in the following order: Method > Deny > DenyIf > Allow > AllowIf</p> <ul style="list-style-type: none">• When a request method is not included in the Method item: Request is allowed.• Matched with Deny, DenyIf, but not matched with Allow, AllowIf: Request is denied.• Other cases: Request is allowed.

Option	Description
Allow, Deny	<p>Executes matching in the following order: Method > Allow > AllowIf > Deny > DenyIf</p> <ul style="list-style-type: none"> • When a request method is not included in the Method item: Request is denied. • Matched with Allow, AllowIf, but not matched with Deny, DenyIf: Request is allowed. • Other cases: Request is denied.

Allow

- Type: Literal
- Range: Up to 8,959 characters
- Specifies the IP address and netmask that are allowed to send requests.
- Specifies IP address and netmask as follows.

```
Allow = "192.168.1.43/255.255.255.0"
```

- Allow = "all" is a special value and allows all IP addresses.
- Up to 256 items can be configured by separating each with a comma (,).

Deny

- Type: Literal
- Specifies the IP address and netmask that are not allowed to send requests. This item can be configured in the same way as the Allow item.

AllowIf

- Type: Literal
- Range: Up to 255 characters
- Specified as "<header item name> <regular expression>".
- Allows access if the <header item name> Header value included in the request matches the specified <regular expression> pattern. The <regular expression> are Perl compatible regular expressions.
- One or more AllowIf values can be specified.

```
AllowIf="Referer http://10.0.0.2/"
```

DenyIf

- Type: Literal
- Range: Up to 255 characters
- DenyIf can be configured similarly to AllowIf.

3.14.2. Example

The following is an example configuration of the ACCESS section.

```
*ACCESS
access1
    Order = "allow, deny",
    Allow = "all"

access2
    Order = "allow, deny",
    Allow = "211.1.1.10, 211.1.1.20"

access3
    Order = "allow, deny",
    Allow = "211.1.1.0/255.255.255.0"

access4
    Order = "deny, allow",
    Deny = "211.1.1.30"

access5
    Order = "allow, deny",
    Allow = "all", Deny = "211.1.1.30"
```

3.15. AUTHENT Section

The AUTHENT section configures user and group authentication to restrict client accesses. This section can be configured in the SVRGROUP, URI, EXT, and DIRECTORY sections.

3.15.1. Configuration Items

The following is the configuration format of the AUTHENT section.

```
*AUTHENT
name    # String[32]
        Type = String[32],
        UserFile = Literal[256],          # $ENV, R.PATH
        #AccessName = String[32]
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the AUTHENT section configuration items.

AUTHENT section name

- Type: String
- Range: Up to 31 characters
- Specifies the AUTHENT section name.

Type (Required)

- Type: String
- Range: Up to 31 characters
- Specifies the authorization control method. Either Basic (HTTP basic authentication) or Digest (HTTP digest authentication). Refer to [Authentication](#) for more information.

UserFile (Required)

- Type: Literal
- Range: Up to 255 characters
- Specifies the file in which user names, passwords, and realms are saved.
- WebtoB provides the '**wsmkpw**' tool, which manages UserFiles. The **wsmkpw** tool can be used to record user names and encrypted codenames to the UserFile.

AccessName

- Type: String
- Range: Up to 31 characters
- Access section name to use for access control on requests for specific IP bandwidth.

3.15.2. Example

The following is an example configuration of the AUTHENT section.

```
*AUTHENT
authent1
    Type = Basic,
    UserFile = "/usr/local/webtob/bin/pwfile"
```

3.16. EXT Section

The EXT section maps a process to a request according to the file extension. WebtoB is already configured with mappings between basic MIME types and processes. This section can be used to configure additional mappings.

3.16.1. Configuration Items

The following is the configuration format of the EXT section.

```
*EXT
name      # String[32]
          #Mimetype = Literal[128],
          #SvrType = String[32],           # HTML
          #SvrName = String[32],
          #SvcName = String[16],
          #VhostName = String[1024],      # LIST[64], MULTI
          #AccessName = String[32],
          #AuthentName = String[32],
          #Options = Literal[256],       # PM.LIST
          #Charset = Literal[32],
          #SCGI = Boolean,                # N
          #ScgiServer = Literal[256],
          #Match = Literal[256],         # "exact"
          #RegExp = Literal[512],
          #Priority = Numeric,           # 50, 0 ~ 100
          #FCGI = Boolean,                # N
          #Extension = Literal[32]
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the EXT section configuration items.

EXT section name

- Type: String
- Range: Up to 31 characters
- Specifies the pattern to match with the HTTP request path extension. Once matched, the EXT section configuration is applied to the request.
- WebtoB supports various pattern types. Configure the pattern types in the Match item.

Mimetype

- Type: Literal
- Range: Up to 127 characters

- Specifies the MIME type corresponding to an extension.

```
MimeType = "text/html"
```

SvrType

- Type: String
- Default value: HTML
- Specifies the server type to process extensions.

SvrName

- Type: String
- Range: Up to 31 characters
- Specifies the server name to process the request.

SvcName

- Type: String
- Range: Up to 15 characters
- Specifies the service name to process the request.

VhostName

- Type: String
- Range: Up to 1023 characters
- When EXT is used in a specific virtual host, the virtual host name defined in the VHOST section is specified.
- Up to 64 items can be configured by separating each item with a comma (,).

AccessName

- Type: String
- Range: Up to 31 characters
- Specifies in the EXT section whether to allow or deny a request coming from a specific IP address.

AuthentName

- Type: String
- Range: Up to 31 characters

- Specifies the AUTHENT section name for authentication. If the server type is JSV, authentication can be applied to only SVRGROUP.

Options

- Type: Literal
- The following are descriptions of configurable options.

Option	Description
{+ -}Cache	Enables or disables the cache (+ -) of the selected content.
SSLRequireSSL	Processes SSL requests only. Responds to Non-SSL requests with the "403 Forbidden" message.
SSLDenySSL	Processes Non-SSL requests only. Responds to SSL requests with the "403 Forbidden" message.
UnSet	Unsets default MIME-Types such as CGI and JSP. Note that HTML cannot be unset since it is one of the most foundational features for web servers.

Charset

- Type: Literal
- Specifies the charset to add to the Content-Type Header.

Value	Description
on	"iso-8859-1" is used.
off	Does not add charset to the Content-Type Header.
<custom>	Uses the user specified value as the charset.

SCGI

- Type: Boolean
- Default value: N
- Specifies whether the request is SCGI when SvrType is set to CGI.
- If set to Y, the SCGI server address is also required.

ScgiServer

- Type: Literal
- Range: Up to 255 characters
- Specifies the SCGI server address in the "server_name[IP_address];port_number" format.

```
SCGIServer = "172.16.1.100:9001"
```

Match

- Type: Literal
- Default value: "exact"
- Specifies the matching pattern type. The matching method used for HTTP Request path extensions depends on the pattern type.
- The following are descriptions of configurable pattern types.

Pattern Type	Description
prefix	The specified pattern is prefixed of the extension. For example, a pattern "abc" is matched with the following extensions: "abc", "abc1", and "abc12".
exact	The specified pattern and extension match exactly. For example, a pattern "abc" is matched only when the extension is "abc".
regexp	The specified pattern is a regular expression used in the Perl script language. To use regexp, the pattern must be configured in RegExp. In this case, the pattern of EXT name is ignored. (For example, pattern "jsp[a-z]" is matched with extensions such as "jspa", "jspb", "jspx", etc.)

RegExp

- Type: Literal
- Range: Up to 511 characters
- Specifies the regular expression pattern.
- Only applied if Match = "regexp".

Priority

- Type: Numeric
- Range: 0 to 100
- Default value: 50
- Prioritizes client requests. Priority is between 1 (lowest) and 100 (highest).

FCGI

- Type: Boolean

- Default value: N
- Specifies whether the request is FastCGI when SvrType is set to CGI.
- If set to Y, the request is processed as Fast CGI.

Extension

- Type: Literal
- Range: Up to 31 characters
- Specifies when to use another value as an extension instead of the EXT section name.

3.16.2. MIME-Type

General MIME-Types are provided by default.

The items listed in the following table do not need to be configured in the EXT section, and they are all processed by a worker thread that processes HTML services. Hence, if the extension is processed by a different server, the server must be specified in the SvrType item of the EXT section.

Extension	MIME-Type	Extension	MIME-Type
avi	"video/video-x-msvideo"	asc	"text/plain"
au	"audio/basic"	ai	"application/postscript"
aif	"audio/x-aiff"	aiff	"audio/x-aiff"
af	"audio/x-aiff"	aifc	"audio/x-aiff"
bmp	"image/bmp"	bin	"application/octet-stream"
bcpio	"application/x-bcpio"	cpio	"application/x-cpio"
cs	"application/x-csh"	cpt	"application/mac-compactpro"
class	"application/octet-stream"	cdf	"application/x-netcdf"
css	"text/css"	doc	"application/msword"
dvi	"application/x-dvi"	dms	"application/octet-stream"
dcr	"application/x-director"	dir	"application/x-director"
dxr	"application/x-director"	eps	"application/postscript"
exe	"application/octet-stream"	ez	"application/andrew-inset"
etx	"text/x-setext"	gif	"image/gif"
gtar	"application/x-gtar"	html	"text/html"
htm	"text/html"	hqx	"application/mac-binhex40"
hdf	"application/x-hdf"	hwp	"application/x-hwp"
hif	"application/x-hif"	hpt	"application/x-hpt"
hst	"application/x-hst"	ief	"image/ief"

Extension	MIME-Type	Extension	MIME-Type
igs	"model/iges"	iges	"model/iges"
ice	"x-conference/x-cooltalk"	jpg	"image/jpeg"
js	"application/x-javascript"	jpeg	"image/jpeg"
jpe	"image/jpeg"	kar	"audio/midi"
latex	"application/x-latex"	lha	"application/octet-stream"
lzh	"application/octet-stream"	mpg	"video/mpeg"
mpeg	"video/mpeg"	mpe	"video/mpeg"
mid	"audio/midi"	midi	"audio/midi"
mov	"video/quicktime"	mif	"application/vnd.mif"
man	"application/x-troff-man"	me	"application/x-troff-me"
ms	"application/x-troff-ms"	mpga	"audio/mpeg"
mp2	"audio/mpeg"	mp3	"audio/mpeg"
msh	"model/mesh"	mesh	"model/mesh"
movie	"video/video-x-sgi-movie"	nc	"application/x-netcdf"
oda	"application/oda"	pdf	"application/pdf"
ps	"application/postscript"	ppt	"application/vnd.ms-powerpoint"
pgn	"application/x-chess-pgn"	pdb	"chemical/x-pdb"
png	"image/png"	pbm	"image/x-portable-bitmap"
pbm	"image/x-portable-bitmap"	pgm	"image/x-portable-graymap"
ppm	"image/x-portable-pixmap"	qt	"video/quicktime"
rtf	"application/rtf"	ra	"audio/x-realaudio"
rgb	"image/x-rgb"	roff	"application/x-troff"
rmm	"audio/x-pn-realaudio"	ram	"audio/x-pn-realaudio"
rm	"application/vnd.rn-realmedia"	rpm	"application/x-rpm"
ras	"image/x-cmu-raster"	rtx	"text/richtext"
rt	"text/vnd.rn-realtext"	rv	"video/vnd.rn-realvideo"
rf	"image/vnd.rn-realflash"	rp	"image/vnd.rn-realpix"
sgm	"text/sgml"	src	"application/x-wais-source"
snd	"audio/basic"	smi	"application/smil"
smil	"application/smil"	spl	"application/x-futuresplash"
skp	"application/x-koan"	skd	"application/x-koan"
skt	"application/x-koan"	skm	"application/x-koan"
sh	"application/x-sh"	shar	"application/x-shar"

Extension	MIME-Type	Extension	MIME-Type
swf	"application/x-shockwave-flash"	sit	"application/x-stuffit"
sv4cpio	"application/x-sv4cpio"	sv4crc	"application/x-sv4crc"
silo	"model/mesh"	sgml	"text/sgml"
sdp	"application/sdp"	txt	"text/plain"
tar	"application/x-tar"	tcl	"application/x-tcl"
tex	"application/x-tex"	texinfo	"application/x-texinfo"
texi	"application/x-texinfo"	t	"application/x-troff"
tr	"application/x-troff"	tiff	"image/tiff"
tif	"image/tiff"	tsv	"text/tab-separated-values"
ustar	"application/x-ustar"	vrml	"model/vrml"
vcd	"application/x-cdlink"	wav	"audio/x-wav"
wrl	"model/vrml"	xls	"application/vnd.ms-excel"
xyz	"chemical/x-pdb"	xbm	"image/x-xbitmap"
xpm	"image/x-pixmap"	xwd	"image/x-windowdump"
xml	"text/xml"	zip	"application/zip"

3.16.3. Example

The following is an example configuration of the EXT section.

```
*EXT
htm
  Mimetype = "text/html",
  SvrType = HTML
php4
  Mimetype = "text/html",
  SvrType = PHP
do
  Mimetype = "text/html",
  SvrType = JSV
```

3.17. SSL Section

The SSL section configures SSL functions used in WebtoB. SSL service operates according to these settings.

3.17.1. Configuration Items

The following is the configuration format of the SSL section.

```
*SSL
name # String[32]
  CertificateFile = Literal[256], # $ENV, R.PATH
  CertificateKeyFile = Literal[256], # $ENV, R.PATH
  #CACertificatePath = Literal[256], # $ENV, R.PATH
  #CACertificateFile = Literal[256], # $ENV
  #CertificateChainFile = Literal[256], # $ENV, R.PATH
  #VerifyDepth = Numeric, # 0 (0-)
  #VerifyClient = Numeric, # 0 (0-3)
  #FakeBasicAuth = Boolean, # N
  #Protocols = Literal[256], # LIST
  #RequiredCiphers = Literal[1024], # LIST
  #RandomFile = Literal[256], # $ENV, R.PATH
  #RandomFilePerConnection = Literal[256], # $ENV, R.PATH
  #PassPhraseDialog = Literal[256], # $ENV, R.PATH
  #CryptoDevice = Literal[256], # "builtin"
  #RenegotiationLevel = String[256], # secure (secure, insecure, disable)
  #DHParameter = Literal[256], # $ENV, R.PATH
  #Options = Literal[256] # PM.LIST
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the SSL section configuration items.

SSL section name

- Type: String
- Range: Up to 31 characters
- Specifies the section name.

CertificateFile (Required)

- Type: Literal
- Range: Up to 255 characters
- The certificate in PEM format, which is Base64 encoded DER (Distinguished Encoding Rules), is used like ASCII code for web transmissions. If the certificate is encrypted, a passphrase is requested.
- Up to 4 certificates can be set using a comma (,) as a delimiter.



The number of certificates must match the number of keys specified with CertificateKeyFile.

CertificateKeyFile (Required)

- Type: Literal
- Range: Up to 255 characters
- Specifies the private key for the server certificate encoded in PEM format.
- If the key is not included with the certificate, use this directive to specify the location of the key. Generally, this file is placed in the WebtoB SSL directory.
- Up to 4 keys can be set using a comma (,) as a delimiter.



The number of keys must match the number of certificates specified with CertificateFile.

CACertificatePath

- Type: Literal
- Range: Up to 255 characters
- Specifies the directory of the certificate.
- The certificate includes data required to certify user certificates. This file is generally encoded in PEM.

CACertificateFile

- Type: Literal
- Range: Up to 255 characters
- Specifies the certificate file when using a single CA to authenticate users.

CertificateChainFile

- Type: Literal
- Range: Up to 255 characters
- Specifies the path to CA certificates used to create a server certificate chain. For client authentication, the CACertificateFile or CACertificatePath item must be set.

VerifyDepth

- Type: Numeric
- Range: 0 to INT_MAX
- Default value: 0
- Specifies the level to trace CA for authentication. Different CAs exist at different levels in a certificate. This item tracks and validates CAs at different levels. If verification from a single CA is

sufficient, set the item value to 1.

VerifyClient

- Type: Numeric
- Default value: 0
- Specifies the authentication level requested to the user.
- The following describes each verification level.

Level	Description
0	The user does not request a certificate.
1	The user can submit a valid certificate to the server.
2	The user must submit a valid certificate to the server.
3	The user must submit a valid certificate to the server. The server does not perform verification without having the certificate.

FakeBasicAuth

- Type: Boolean
- Default value: N
- Specifies the option to translate subject DN (distinguished name) of the certificate into a HTTP basic authorization username. If this item is used with the VerifyClient item, the authentication result can be seen in the log file. A predefined password is used as the code.

Protocols

- Type: Literal
- Range: Up to 255 characters
- Default value: "TLSv1, TLSv1.1, TLSv1.2, TLSv1.3"
- Specifies the protocols that can be used by the server. This can determine whether to support a specific TLS version. Note that "SSLv2" and "SSLv3" are no longer supported.
- To disable a specific protocol, add a hyphen (-) before the protocol name.
- When configuring multiple hosts using the same port, note that only the *SSL.Protocols setting for the first specified host can be applied.

RequiredCiphers

- Type: Literal
- Range: Up to 1,023 characters
- Default value: "HIGH:MEDIUM:!SSLv2:!PSK:!SRP:!ADH:!AECDH:!EXP:!RC4:!IDEA:!3DES"

- Specifies the ciphers that can be used by the server. Specific cipher, SSL or TLS versions can be specified. Since WebtoB uses OpenSSL, refer to the OpenSSL guide for cipher names.

RandomFile

- Type: Literal
- Range: Up to 255 characters
- Specifies the random seed file and its size for SSL. If a file is selected, WebtoB extracts a random value from this file to create a random seed when creating a password for SSL.

RandomFilePerConnection

- Type: Literal
- Range: Up to 255 characters
- Specifies the random seed file for SSL and its size used to establish each connection.

PassPhraseDialog

- Type: Literal
- Default value: "builtin"
- Specifies the passphrase value to use for encrypted private key files when SSL is used.
- The following are descriptions of configuration values.

Value	Description
builtin	Asks for a passphrase when starting WebtoB.
exec:<program path>	When starting WebtoB, a program is executed and its output is used as a passphrase. A file that is executed with "exec" can be a compiled executable or a shell script.
file:<passphrase file path>	Uses the passphrase in the passphrase file that was generated by the wsmkppd tool when WebtoB is started.

CryptoDevice

- Type: Literal
- Range: Up to 255 characters
- Default value: "builtin" (The default value specifies the use of the method implemented in OpenSSL.)
- Specifies the SSL external crypto device type.

RenegotiationLevel

- Type: String
- Range: Up to 255 characters
- Default value: secure
- Specifies the renegotiation level when SSL is used.
- The following are descriptions of configuration values.

Value	Description
secure	Proceeds to renegotiate when the client and web server are safe. Example: RFC5746
insecure	Proceeds to renegotiate although the client and web server are not safe. Example: CVE-2009-3555
disable	Does not renegotiate in any situation.



Note that if renegotiation proceeds when safety is not guaranteed, it becomes vulnerable to MITM (Man in the Middle) Attack or DoS (Denial of Service) Attack.

DHParameter

- Type: Literal
- Range: Up to 255
- Specifies the Diffie-Hellman group created to enhance security of encrypted communication.

You can create a Diffie-Hellman group (2048bit) as follows:

```
wbssl dhparam -out dhparams.pem 2048
```

SSLServerCipherPref

- Type: Boolean
- Default value: N
- Specifies to use the order of ciphers set in RequiredCiphers when using SSL algorithm.

Options

- Type: Literal
- Range: Up to 255 characters
- The following are descriptions of the configurable options.

Option	Description
StdEnvVars	<p data-bbox="438 152 1458 275">Creates SSL environmental variables or the HTTP Request Header value so that the basic information used in the SSL Handshake process can be used in CGI/PHP, JEUS, and internal Reverse Proxy servers.</p> <ul data-bbox="464 309 1442 573" style="list-style-type: none"> <li data-bbox="464 309 1442 427">• Environmental variables available in CGI/PHP servers: SSL_SESSION_ID (SSL session id), SSL_CIPHER (used cipher), and SSL_CIPHER_USEKEYSIZE (used cipher bit) <li data-bbox="464 454 1386 573">• HTTP Request Headers available in JEUS or internal Reverse Proxy servers: WJP-SSL-SESSION-ID (SSL session id), WJP-SSL-CIPHER (used cipher), and WJP-SSL-CIPHER-USEKEYSIZE (used cipher bit) <p data-bbox="438 600 1046 638">The following are examples in each language.</p> <ul data-bbox="464 667 544 701" style="list-style-type: none"> <li data-bbox="464 667 544 701">• CGI <div data-bbox="491 730 1458 813" style="border: 1px solid #ccc; border-radius: 5px; padding: 5px; background-color: #f9f9f9;"> <pre data-bbox="512 757 1150 790">c - getenv("SSL_CIPHER") perl - \$ENV{'SSL_CIPHER'}</pre> </div> <ul data-bbox="464 869 544 902" style="list-style-type: none"> <li data-bbox="464 869 544 902">• PHP <div data-bbox="491 931 1458 1014" style="border: 1px solid #ccc; border-radius: 5px; padding: 5px; background-color: #f9f9f9;"> <pre data-bbox="512 958 794 992">\$_SERVER['SSL_CIPHER']</pre> </div> <ul data-bbox="464 1070 528 1104" style="list-style-type: none"> <li data-bbox="464 1070 528 1104">• JSP <div data-bbox="491 1133 1458 1216" style="border: 1px solid #ccc; border-radius: 5px; padding: 5px; background-color: #f9f9f9;"> <pre data-bbox="512 1160 959 1193">request.getHeader("WJP-SSL-CIPHER")</pre> </div>

Option	Description
ExportClientCert	<p data-bbox="438 150 1458 398">Creates SSL environmental variables or the HTTP Request Header value so that the client certificate can be used in CGI/PHP, JEUS, and internal Reverse Proxy servers when client authentication is implemented (VerifyClient = 2). To process the client certificate's extension items (Extension Items - Key Usage, Certificate Policies, CRL Distribution Points, etc.), obtain and parse the client certificate information from the configuration to use it.</p> <ul data-bbox="464 427 1437 651" style="list-style-type: none"> <li data-bbox="464 427 1437 506">• Environmental variables available in CGI/PHP servers: SSL_CLIENT_CERT (client certificate - PEM encoded string value). <li data-bbox="464 535 1437 651">• HTTP Request Headers available in JEUS or internal Reverse Proxy servers: WJP-SSL-CLIENT-CERT (client certificate - PEM encoded string value). <p data-bbox="438 680 1046 719">The following are examples in each language.</p> <ul data-bbox="464 748 544 786" style="list-style-type: none"> <li data-bbox="464 748 544 786">• CGI <div data-bbox="491 813 1458 898" style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;"> <pre data-bbox="512 837 1278 869">c - getenv("SSL_CLIENT_CERT") perl - \$ENV{'SSL_CLIENT_CERT'}</pre> </div> <ul data-bbox="464 949 544 987" style="list-style-type: none"> <li data-bbox="464 949 544 987">• PHP <div data-bbox="491 1014 1458 1099" style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;"> <pre data-bbox="512 1039 858 1070">\$_SERVER['SSL_CLIENT_CERT']</pre> </div> <ul data-bbox="464 1151 528 1189" style="list-style-type: none"> <li data-bbox="464 1151 528 1189">• JSP <div data-bbox="491 1216 1458 1301" style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;"> <pre data-bbox="512 1240 1023 1272">request.getHeader("WJP-SSL-CLIENT-CERT")</pre> </div>

3.17.2. CA Command

This section describes CA commands used in UNIX.

CertificateFile or CertificateKeyFile specifies the path to the key and certificate files for user authentication. While the key is generally supplied by external security experts, it is possible for a server administrator to create this file and use it to secure connections with clients. The key and certificate is generated using a tool called CA.

The CA program is in the executable directory (/bin) of WebtoB.

- Certificate creation

This command creates a certificate. Using this command, a server creates a certificate and uses the certificate to authenticate a user. Successful execution of this command also generates a private key. When generating a certificate, CA asks for a password. **Do not forget this password.** If a self-generated certificate is used, the password is required when WebtoB starts up. WebtoB will fail to boot without the correct password.

```
$ CA -newcert
```

- Certificate check

Verifies the input file.

```
$ CA -newca
```

- Certificate Request Form creation

Creates a certificate request form. This form is required by the Certificate Authority, such as Verisign, when creating a certificate.

```
$ CA -newreq
```

- Sign input

Enter a sign to the created key.

```
$ CA -sign
```



For more information on SSL, refer to other references or sites.

3.17.3. Example

The following is an example configuration of the SSL section.

```
*SSL
ssl1
CertificateFile = "/user/webtob/ssl/newcert.pem",
CertificateKeyFile = "/user/webtob/ssl/newcert.pem",
PassPhraseDialog = "exec: /user/webtob/ssl/pass.sh"
```

3.18. PROXY_SSL Section

The PROXY_SSL section configures SSL functions used in WebtoB when it is used as a proxy. SSL service operates according to these settings.

3.18.1. Configuration Items

The following is the configuration format of the PROXY_SSL section.

```
*PROXY_SSL
name      # String[32]
  Verify = Numeric,           # 0 (0-3)
  #VerifyDepth = Numeric,    # 0 (0-)
  #CACertificatePath = Literal[256], # $ENV, R.PATH
  #CACertificateFile = Literal[256], # $ENV
  #CheckPeerValidPeriod = Boolean,   # N
  #Protocols = Literal[256],        # LIST
  #RequiredCiphers = Literal[1024],  # LIST
  #CertificateFile = Literal[256],   # $ENV, R.PATH
  #CertificateKeyFile = Literal[256], # $ENV, R.PATH
  #CertificateChainFile = Literal[256], # $ENV, R.PATH
  #PassPhraseDialog = Literal[256]   # $ENV, R.PATH
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the PROXY_SSL section configuration items.

PROXY_SSL section name

- Type: String
- Range: Up to 31 characters
- Specifies the section name.

Verify

- Type: Numeric
- Default value: 0
- Specifies the verification level of the internal server certificate.
- The following describes each authentication level.

Level	Description
0	Internal server certificate does not get verified.
1	Internal server can submit a valid certificate. If submitted, WebtoB verifies the certificate.
2	Internal server must submit a valid certificate and WebtoB verifies the certificate received from the internal server.
3	Internal server must submit a valid certificate. WebtoB does not does not perform verification without having the certificate.

VerifyDepth

- Type: Numeric

- Range: 0 to INT_MAX
- Default value: 0
- This is used when different CAs intervene sequentially in the verification process. Specifies to what level depth the linked CAs are tracked for authentication. If verification from a single CA is sufficient, set this to 1.

CACertificatePath

- Type: Literal
- Range: Up to 255 characters
- Specifies the directory where the certificate will be saved.
- The certificate includes data required to verify the certificate of the server to connect to. This file is generally encoded in PEM.

CACertificateFile

- Type: Literal
- Range: Up to 255 characters
- Use this directive instead of CACertificatePath to verify the server from a single CA (Certificate Authority). The certificate file must be encoded in PEM.

CheckPeerValidPeriod

- Type: Literal
- Default value: N
- Specifies whether to verify the certificate expiry date when verifying the internal server certificate.

Protocols

- Type: Literal
- Range: Up to 255 characters
- Default value: "TLSv1, TLSv1.1, TLSv1.2, TLSv1.3"
- Specifies the protocols that can be used by the server. Specific SSL or TLS versions can be specified. "TLSv1.1, TLSv1.2" is supported for WBSSL 1.0.1 or later. "SSLv2 and "SSLv3" are no longer supported.
- To disable a specific protocol, add a hyphen (-) before the protocol name.

RequiredCiphers

- Type: Literal

- Range: Up to 1023 characters
- Default value: "HIGH:MEDIUM:!SSLv2:!PSK:!SRP:!ADH:!AECDH:!EXP:!RC4:!IDEA:!3DES"
- Specifies the ciphers that can be used by the server. Specific cipher, SSL or TLS versions can be specified. Since WebtoB uses OpenSSL, refer to the OpenSSL guide for cipher names.

CertificateFile

- Type: Literal
- Range: Up to 255 characters
- Specifies the client certificate file encoded in PEM. This is required if the internal server requires client verification. The certificate is encoded in DER and is used like ASCII for web transmission. If the server certificate is encoded, then it will prompt for a passphrase.

CertificateKeyFile

- Type: Literal
- Range: Up to 255 characters
- Specifies the private key of a PEM encoded certificate used on the client-side. This is required if the internal server requires client verification.
- If the key is not combined with the certificate, use this directive to specify the location of the key. Generally, this file is placed in the WebtoB SSL directory.

CertificateChainFile

- Type: Literal
- Range: Up to 255 characters
- Specifies the paths to the certificates of CAs (root certificate authority) that are used to create the certificate chain of client certificates.

PassPhraseDialog

- Type: Literal
- Default value: "builtin"
- Specifies how to retrieve the passphrase for an encrypted private key file when using client authentication (CertificateFile, CertificateKeyFile) in PROXY_SSL.
- The following are descriptions of configuration values.

Value	Description
builtin	Asks for a passphrase when starting WebtoB.

Value	Description
exec:<program path>	When starting WebtoB, a program is executed and its output is used as a passphrase. A file that is executed with “exec” can be a compiled executable or a shell script.
file:<passphrase file path>	Uses the passphrase in the passphrase file that was generated by the wsmkppd tool when WebtoB is started.

3.18.2. Example

The following is an example configuration of the PROXY_SSL section.

```
*PROXY_SSL
proxy_ssl1
  Verify = 2,      # Verify the server certificate
  VerifyDepth = 1,
  CACertificateFile="/user/webtob/ssl/server.crt",
  CheckPeerValidPeriod = Y
```

3.19. ERRORDOCUMENT Section

There are four ways to troubleshoot an error in WebtoB:

1. Output the error message defined in the source code.
2. Output the error message defined by the user.
3. Retransmit it to a local URL.
4. Retransmit it to an external URL.

For the second, third, and fourth methods, the ERRORDOCUMENT section configures an HTTP response status code, except 401, with the URL to redirect to.

3.19.1. Configuration Items

The following is the configuration format of the ERRORDOCUMENT section.

```
*ERRORDOCUMENT
name      # String[32]
  Status = Numeric,           # (HTTP status code)
  #MultiStatus = Literal[256], # (HTTP status code)
  Url = Literal[256]
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the ERRORDOCUMENT section configuration items.

ERRORDOCUMENT section name

- Type: String
- Range: Up to 31 characters
- Specifies the section name.

Status (Required)

- Type: Numeric
- Specifies the HTTP status code.

MultiStatus

- Type: Literal
- Range: Up to 255 characters
- Specifies multiple HTTP status codes for the same URL.
- Either Status or MultiStatus must be configured. If both are set, the first specified one will take precedence and be applied first.

Url (Required)

- Type: Literal
- Range: Up to 255 characters
- Specifies the relative path from DocRoot or absolute path that a client can interpret.

3.19.2. Example

The following is an example configuration of the ERRORDOCUMENT section.

```
*ERRORDOCUMENT
forbidden403
    Status = 403,
    Url = "err/403.html"
notfound404
    Status = 404,
    Url = "http://www.tmax.co.kr/404.html"
```

3.20. EXPIRES Section

The EXPIRES section configures server response header information.

When transmitting a specific MIME type document, the document's expiration date can be set in the

server response header. When a client's web browser reconnects to a site, only updated documents and documents that do not exist in the browser cache are retrieved from the site.

The client browser sends a specific MIME-Type and MIME expiration date to the web server. The browser requests the server to retransmit only the documents that have expired. The web server analyzes the MIME-Type and the expiration data in the client request header and retransmits when the MIME-Type document has expired.

The web server refers to the configuration file for the expiration date sent from the client to the web server. The web server then transmits the expiration date to client. The EXPIRES section defines the configuration related to the expiration date.

3.20.1. Configuration Items

The following is the configuration format of the EXPIRES section.

```
*EXPIRES
name # String[32]
    ExpiresTime = Literal[256],
    #Mimetype = Literal[128],
    #Url = Literal[256]
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the EXPIRES section configuration items.

EXPIRES section name

- Type: String
- Range: Up to 31 characters
- Specifies the section name.

ExpiresTime (Required)

- Type: Literal
- Range: Up to 255 characters
- Specifies the expiration date in either of the following two formats. If only ExpiresTime is specified, the expiration date in the Response Header for all requests is specified.
 - `<code><seconds>`

Item	Description
<code><code></code>	<ul style="list-style-type: none">◦ 'M': expiration date = file Modification date + <code><seconds></code>.◦ 'A': expiration date = Access time + <code><seconds></code>.

Item	Description
<second>	Specifies the time cycle in seconds. <ul style="list-style-type: none"> ◦ 1 hour(3600) ◦ 1day(86400) ◦ 1week(604800) ◦ 1moth(241920)

- <base> [plus] {<num> <type>}*

Item	Description
<base>	<ul style="list-style-type: none"> ◦ 'access' or 'now' : access time ◦ 'modification' : file modification date
[plus]	optional
<num>	integer value
<type>	<ul style="list-style-type: none"> ◦ 'years' ◦ 'months' ◦ 'weeks' ◦ 'days' ◦ 'hours' ◦ 'minutes' ◦ 'seconds' ◦ 'year' ◦ 'month' ◦ 'week' ◦ 'day' ◦ 'hour' ◦ 'minute' or 'second'

Mimetype

- Type: Literal
- Range: Up to 127 characters
- Specifies the response MIME type for the expiry date.

Url

- Type: Literal
- Range: Up to 255 characters
- Specifies the request URL for the expiry date.

3.20.2. Example

The following is an example configuration of the EXPIRES section.

```
*EXPIRES
exp1
  MimeType = "text/html",
  ExpiresTime = "A604800"
exp2
  MimeType = "image/gif",
  ExpiresTime = "A2419200"
exp3
  ExpiresTime = "A86400"
exp4
  Url = "/news/",
  ExpiresTime = "modification 1 days"
exp5
  MimeType = "text/html",
  ExpiresTime = "access plus 1 weeks"
exp6
  ExpiresTime = "M86400"
exp7
  Url = "/image/",
  MimeType = "image/gif",
  ExpiresTime = "access plus 1 months"
exp8
  MimeType = "text/html",
  ExpiresTime = "M86400"
```

3.21. TCPGW Section

TCPGW acts as a proxy that transfers a TCP connection from specific port or IP address to another server. TCPGW never interprets transferred data. TCPGW leaves a sysLog instead of an AccessLog when a client opens or terminates access.

3.21.1. Configuration Items

The following is the environment configuration format of the TCPGW section.

```
*TCPGW
name # String[32]
  ServerAddress = Literal[1024], # LIST[100]
```

```

#Port = Literal[1024],          # (1-65535), LIST[100]
#Listen = Literal[1024],       # LIST[100]
#SSLFlag = Boolean,           # N
#SSLName = String[32],
#Timeout = Numeric,           # 300 (0-)
#CheckAliveTime = Numeric,    # 0 (0-)
#ConnectTimeout = Numeric,    # 5 (0-)
#ConnectDirection = String[256], # "CS"
#Schedule = String[256],      # "RR"
#AccessName = String[32],
#ClientConnectionLog = Boolean # Y

```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the TCPGW section configuration items.

TCPGW section name

- Type: String
- Range: Up to 31 characters
- Specifies the section name.

ServerAddress (Required)

- Type: Literal
- Range: Up to 1,023 characters
- Specifies the pairs of an IP address and a port number of servers that will process client requests.
- Up to 100 pairs can be specified. The requests are distributed to servers by using a round robin method.

Port

- Type: Literal
- Range: Up to 1,023 characters
- Specifies the port used to listen to client requests.
- Up to 100 ports can be specified. Multiple ports can listen concurrently through the port configuration.



If both the Port and Listen items are configured, the Port item is ignored.

Listen

- Type: Literal

- Range: Up to 1023 characters
- Specifies the pairs of IP address and port number used to listen to client requests.
- Up to 100 pairs can be specified. If a server has multiple IP addresses, this item can be used to accept only the requests from a specific IP address.

SSLFlag

- Type: Boolean
- Default value: N
- Specifies when to apply SSL.

SSLName

- Type: String
- Range: Up to 31 characters
- Specifies the SSL section name.

Timeout

- Type: Numeric
- Unit: Second
- Range: 0 to INT_MAX
- Default value: 300
- Specifies the maximum period of time to wait for a server response. After this timeout expires, the connection is ended.
- If set to 0, there is no time limit.

CheckAliveTime

- Type: Numeric
- Unit: Second
- Range: 0 to INT_MAX
- Default value: 0
- Specifies the time interval at which a server status check occurs. Failed servers are excluded from load balancing.
- Failed servers are regularly checked at specified intervals to assess their readiness for inclusion in load balancing for subsequent incoming requests.
- If set to 0, a server status check occurs for each individual incoming request.

ConnectTimeout

- Type: Numeric
- Unit: Second
- Range: 0 to INT_MAX
- Default value: 5
- Specifies the maximum period of time for WebtoB TCP gateway to try to access a set server in order to send a client request. After this timeout expires, the gateway tries to connect to another server.
- If set to 0, there is no time limit.

ConnectDirection

- Type: String
- Range: CS | SC
- Default value: CS
- Specifies the connection direction: client to server (CS) or server to client (SC).

Schedule

- Type: String
- Range: RR | FA
- Default value: RR
- Specifies the request dispatch method when multiple external servers are configured.

AccessName

- Type: String
- Range: Up to 31 characters
- Specifies the ACCESS section name used to determine whether to allow user accesses.

ClientConnectionLog

- Type: Boolean
- Default value: Y
- Specifies whether to record logs in SysLog when the client is connecting or disconnecting to/from TCP gateway port.

3.21.2. Example

The following is an example configuration of the TCGW section.

```
*TCGW
tcpgw1
  Port = "5000",
  ServerAddress = "192.168.1.20:5000, 92.168.1.21:5000"
tcpgw2
  Listen = "192.168.1.10:5050",
  ServerAddress = "192.168.1.20:5050, 92.168.1.21:5050",
  TimeOut = 0,
  AccessName = access1
tcpgw3
  Listen = "192.168.1.10:5060",
  ServerAddress = "192.168.1.20:5060,92.168.1.21:5060",
  ConnectTimeOut = 20,
  AccessName = access2

*ACCESS
access1
  Order = "allow, deny",
  Allow = "all"
access2
  Order = "allow, deny",
  Allow = "211.1.1.10, 211.1.1.20"
```

3.22. REVERSE_PROXY_GROUP Section

The REVERSE_PROXY_GROUP section is used to manage multiple REVERSE_PROXY sections as a group for load balancing and sticky session routing.

3.22.1. Configuration Items

The following is the configuration format of the REVERSE_PROXY_GROUP section.

```
*REVERSE_PROXY_GROUP
name # String[32]
  PathPrefix = Literal[256],
  ServerPathPrefix = Literal[256],
  #RegExp = Literal[512],
  #SetHostHeader = Literal[256],
  #VhostName = String[1024], # LIST[64], MULTI
  #RewriteRedirect = Literal[256], # MULTI[16]
  #RewriteCookieDomain = Literal[256],
  #RewriteCookiePath = Literal[256],
  #RewriteHtmlMaxSize = Numeric, # 10240 (1-)
  #RewriteHtmlUrl = Literal[256], # MULTI[32]
  #HtmlUrl = Literal[256], # MULTI[64]
  #WBRoutingCookieKey = Literal[256],
  #SessionIdCookieKey = Literal[256],
  #FlexibleStickySessionRouting = Boolean, # N
```

```
#LBBackup = Literal[256],
#FailbackCheckTime = Numeric,          # 60 (0-)
#AccessName = String[32],
#Headers = Literal[256],                #LIST[15]
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the REVERSE_PROXY section configuration items.

REVERSE_PROXY_GROUP section name

- Type: String
- Range: Up to 31 characters
- Specifies the reverse proxy group name.

PathPrefix (Required)

- Type: Literal
- Range: Up to 255 characters
- Specifies the HTTP request URL prefix. If the request URL is prefixed with the specified value, the request is sent to a reverse proxy.

```
PathPrefix = "/internal/"
```

ServerPathPrefix (Required)

- Type: Literal
- Range: Up to 255 characters
- Specifies the prefix to replace the HTTP request URL prefix. The modified request is sent to an internal server set in the ServerAddress item.
- Using the following setting, the HTTP Request URL is changed from "/internal/abc.html" to "/docs/abc.html" and then sent to the internal server.

```
ServerPathPrefix = "/docs/"
```

RegExp

- Type: Literal
- Range: Up to 511 characters
- Specifies the regular expression to match with the HTTP request URL. If they match, the request

is sent to a reverse proxy.

- '!' means to exclude.

```
RegExp = "\.(do|jsp)$"
```

SetHostHeader

- Type: Literal
- Range: Up to 255 characters
- Specifies the Host header when forwarding the request to the internal server using reverse proxy. If not set, the value specified in ServerAddress is used.

If set to "\$BypassHostHeader", the client Host header is used without modification.

```
SetHostHeader = "reverseproxy.server.com"
```

VhostName

- Type: String
- Range: Up to 1023 characters
- Specifies the reverse proxy's virtual host name.
- Up to 64 items can be configured by separating each with a comma (,).

RewriteRedirect

- Type: Literal
- Range: Up to 255 characters
- Specifies the two strings delimited by a blank space. When the status code of an internal server response is 301, 302, 303, or 307 (related to redirection), the Location and Content-Location headers for the response are compared to the first string. If they match, the domain name and port are replaced by those of the request, and the path is replaced by the second string.
- The following replaces "http://internal.server.com:80/docs/abc.html", the location of original response, by "http://webtob:8100/internal/abc.html".

```
RewriteRedirect = "http://internal.server.com:80/docs/ /internal/"  
RewriteRedirect = "http://internal.server.com:80/docs_kr/ /internal_kr/"  
RewriteRedirect = "http://internal.server.com:80/docs_ch/ /internal_ch/"
```

- Up to 16 items can be specified.

RewriteCookieDomain

- Type: Literal
- Range: Up to 255 characters
- Used as a condition to change the "domain=" value of the Cookie Header item in the internal server response. If the string specified in "domain=" matches the string specified in RewriteCookieDomain, the "domain=" value is replaced with the domain of the user request.
- Using the following configuration, if the WebtoB domain is webtob, the cookie of the original response "jsessionid=abc, domain=internal.server.com" is modified to "jsessionid=abc, domain=webtob".

```
RewriteCookieDomain = "internal.server.com"
```

RewriteCookiePath

- Type: Literal
- Range: Up to 255 characters
- Used as a condition to change the "path=" value of the Cookie header item in the internal server response. If the string specified in "path=" is prefixed with the first string of the RewriteCookiePath, the "path=" string is replaced with the second string.
- Using the following configuration, the cookie of the original response "jsessionid=abc, path=/jeus/application" is modified to "jsessionid=abc, path=/jeus_proxy/application".

```
RewriteCookiePath = "/jeus /jeus_proxy"
```

RewriteHtmlMaxSize

- Type: Numeric
- Unit: bytes
- Range: 1 ~ INT_MAX
- Default value: 10240
- If the response is an HTML page (Content-Type: text/html), specific tag values inside the page can be modified. This value sets the maximum internal buffer size that can be used.
- If the response size is greater than the specified value, the original response is sent to the client.

RewriteHtmlUrl

- Type: Literal
- Range: Up to 255 characters
- Specifies the two strings used to replace the URL of an HTML document.

- If the URL is prefixed with the first string, the part is replaced by the second string. The domain name and port in the URL are replaced by the WebtoB server address.
- A tag and its attributes that specify a URL can be defined in the HtmlUrl item.
- It is assumed that a meta tag always includes a URL.

```
<meta ... http-equiv="refresh" ... content="http://internal.server.com:80/docs/link3.html" ...>
```

```
RewriteHtmlUrl = "http://test2:80/ /proxy/",
RewriteHtmlUrl = "/ /proxy/"
```

- Up to 32 items can be configured for RewriteRedirect.

HtmlUrl

- Type: Literal
- Range: Up to 255 characters
- If the ReverseProxyGroupName is configured, this can be omitted and the setting in the REVERSE_PROXY_GROUP section can be used instead.
- Specifies the names of the tags and attributes that include the URL in the HTML document.
The first string is the tag name. Starting from the second line, the attribute names are specified.
- Up to 64 tags can be configured by specifying HtmlUrl multiple times.

```
HtmlUrl = "a href",
HtmlUrl = "img src longdesc usemap"
```

WBRoutingCookieKey

- Type: Literal
- Range: Up to 255 characters
- Specifies the key in the HTTP cookie used for routing between reverse proxy servers in a group.
- This item overrides the SessionIdCookieKey item. If set, StickySessionRoutingId set in the REVERSE_PROXY section is ignored.
- Routing id is set to the Base64 encoded values of Reverse Proxy Group Name and Reverse Proxy Name.
- Using the following configuration, the response Set-Cookie Header is set to a form of W2BRID=RPG1.rproxy1 and is used for routing by checking the request Cookie Header.

```
WBRoutingCookieKey = "W2BRID"
```

- WBRoutingCookieKey value can be set to the SessionIdCookieKey value. In this case, if the key, e.g., JSESSIONID, set in the response (Set-Cookie) header received from the internal Reverse Proxy server exists, add the routingid to the key value and send it to the client to use it as the routingid for any subsequent requests.

SessionIdCookieKey

- Type: Literal
- Range: Up to 255 characters
- Default value: "JSESSIONID"
- Specifies the key in the HTTP cookie used for session routing.

FlexibleStickySessionRouting

- Type: Boolean
- Default value: N
- Specifies the option to use flexible session routing.
- The following are descriptions of the configurable options.

Value	Description
Y	If connections up to MaxPersistentServerConnections of the internal server are in the RUN state when using the StickySessionRoutingId for session routing, requests are routed to another internal server connection with a different StickySessionRoutingId instead of being queued.
N (default value)	Uses the basic sticky session routing. Requests are routed only to internal server connections with the same StickySessionRoutingId, and requests are queued if connections up to MaxPersistentServerConnections are all in the RUN state.



When using flexible routing, multiple clients with the same JSESSIONID can be each routed to different servers. It is recommended to use the default value.

LBBBackup

- Type: Literal
- Range: Up to 31 characters
- Specifies the backup server for reverse proxy servers.

FailbackCheckTime

- Type: Literal
- Range: 0 to INT_MAX
- Default value: 60 (If set to 0, Failback checks are not performed.)
- Specifies the time interval at which a backup server tries to fail back. If set to 0, failback is not used.

AccessName

- Type: String
- Range: Up to 31 characters
- Specifies the ACCESS section name to use to accept or reject a reverse proxy request from a specific IP address.

Headers

- Type: Literal
- Range: Up to 255 characters
- Specifies the name of a HEADERS section.
- Up to 15 items can be specified by separating each with a comma (,).

3.22.2. Example

The following is an example configuration of the REVERSE_PROXY_GROUP section.

```
*REVERSE_PROXY_GROUP
rpg_internal
  VhostName = "vhost1",
  PathPrefix = "/internal/",
  RegExp = "\.(do|jsp)$",
  ServerPathPrefix = "/docs/",
  RewriteRedirect = "http://internal.server.com:80/docs/ /internal/",
  RewriteCookieDomain = "internal.server.com",
  RewriteCookiePath = "/jeus /jeus_proxy",
  RewriteHtmlUrl = "http://internal.server.com:80/ /internal/",
  RewriteHtmlUrl = "/ /internal/",
  HtmlUrl = "a href",
  HtmlUrl = "img src longdesc usemap",
  RewriteHtmlMaxSize = 4194304,
  SessionIdCookieKey = "JSESSIONID",
  FlexibleStickySessionRouting = N
```

3.23. REVERSE_PROXY Section

The REVERSE_PROXY section configures a reverse proxy. A reverse proxy modifies a client request and then passes it to an internal server. It also modifies headers of the response and sends it to the client.

3.23.1. Configuration Items

The following is the environment configuration format of the REVERSE_PROXY section.

```
*REVERSE_PROXY
name      # String[32]
  PathPrefix = Literal[256],
  ServerPathPrefix = Literal[256],
  ServerAddress = Literal[256],
  #ReverseProxyGroupName = Literal[32],
  #RegExp = Literal[512],
  #SetHostHeader = Literal[256],
  #VhostName = String[1024],          # LIST[64], MULTI
  #HttpInbufSize = Numeric,         # 16384 (0-)
  #RewriteRedirect = Literal[256],  # MULTI[16]
  #RewriteCookieDomain = Literal[256],
  #RewriteCookiePath = Literal[256],
  #RewriteHtmlMaxSize = Numeric,    # 10240 (1-)
  #RewriteHtmlUrl = Literal[256],   # MULTI[32]
  #HtmlUrl = Literal[256],          # MULTI[64]
  #StickySessionRoutingId = Literal[256],
  #MinPersistentServerConnections = Numeric, # 0 (0-)
  #MaxPersistentServerConnections = Numeric, # 0 (0-)
  #PersistentServerCheckTime = Numeric,     # 30 (0-)
  #PersistentServerTimeout = Numeric,       # 300 (0-)
  #PersistentServerCheckUrl = Literal[256],
  #ProxySslFlag = Boolean,                  # N
  #ProxySslName = String[32],
  #MaxWebsocketConnections = Numeric,      # 0 (0-)
  #WebsocketSessionTimeout = Numeric,      # 0 (0-)
  #Options = Literal[256],                  # PM.LIST
  #ConnectRetryCount = Numeric,            # 3 (0-)
  #ConnectTimeout = Numeric,               # 5 (0-)
  #AccessName = String[32],
  #Headers = Literal[256],                 # LIST[15]
  #Compression = Literal[256],             # LIST[32], MULTI
  #CompressionMinSize = Numeric            # 1 (1-)
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the REVERSE_PROXY section configuration items.

REVERSE_PROXY section name

- Type: String

- Range: Up to 31 characters
- Specifies the reverse proxy name.

PathPrefix (Required)

- Type: Literal
- Range: Up to 255 characters
- Specifies the request URL prefix. If the request URL is prefixed with the specified value, the request is sent to the reverse proxy.
- If an HTTP Request URL is prefixed with the specified value, Reverse Proxy processes the request.

```
PathPrefix = "/internal/"
```

ServerPathPrefix (Required)

- Type: Literal
- Range: Up to 255 characters
- If the ReverseProxyGroupName is configured, this can be omitted and the setting in the REVERSE_PROXY_GROUP section can be used instead. If the ReverseProxyGroupName is used, this value must be set to the value set in the REVERSE_PROXY_GROUP section.
- Specifies the prefix to replace the request URL prefix. The modified request is sent to an internal server set in the ServerAddress item.
- Using the following setting, the HTTP Request URL is changed from "/internal/abc.html" to "/docs/abc.html" and then sent to the internal server.

```
ServerPathPrefix = "/docs/"
```

ServerAddress (Required)

- Type: Literal
- Range: Up to 255 characters
- Specifies the internal server address to which the request is sent.

```
ServerAddress = "internal.server.com:80"
```

ReverseProxyGroupName

- Type: Literal
- Range: Up to 31 characters

- Specifies the name defined in the REVERSE_PROXY_GROUP.
- Use this to configure a multi server setup by grouping multiple reverse proxies.

```
ReverseProxyGroupName = "rproxyGroup1"
```

RegExp

- Type: Literal
- Range: Up to 511 characters
- This setting is not necessary when ReverseProxyGroupName is used. In this case, the setting values specified in the REVERSE_PROXY_GROUP section will be used. When using the ReverseProxyGroupName setting, RegExp must contain the same setting values as the REVERSE_PROXY_GROUP section.
- When an HTTP request URL matches the regular expression, the request will be handled by a reverse proxy.
- '!' means to exclude.

```
RegExp = "\.(do|jsp)$"
```

SetHostHeader

- Type: Literal
- Range: Up to 255 characters
- If the ReverseProxyGroupName is configured, this can be omitted and the setting in the REVERSE_PROXY_GROUP section can be used instead.
- Specifies the Host Header value when a request is forwarded to the internal server using Reverse Proxy. If not set, the Host Header value is set to the value specified in ServerAddress.

If set to "\$BypassHostHeader", the client-sent Host Header value is used.

```
SetHostHeader = "reverseproxy.server.com"
```

VhostName

- Type: String
- Range: Up to 1,023 characters
- If the ReverseProxyGroupName is configured, this can be omitted and the setting in the REVERSE_PROXY_GROUP section can be used instead. If the ReverseProxyGroupName is used, this value must be set to the value set in the REVERSE_PROXY_GROUP section.
- Specifies the reverse proxy's virtual host name.

- Up to 64 items can be configured by separating each with a comma (,).

HttpInBufSize

- Type: Numeric
- Unit: bytes
- Range: 0 to INT_MAX
- Default value: 16384
- Specifies the buffer size used when a server receives user requests.
- If value is set to 0 or greater than 16777216 (16 MB), the value 16777216 (16 MB) is used.

RewriteRedirect

- If the ReverseProxyGroupName is configured, this can be omitted and the setting in the REVERSE_PROXY_GROUP section can be used instead.
- Specifies the two strings delimited by a blank space. When the status code of an internal server response is 301, 302, 303, or 307 (related to redirection), the Location and Content-Location headers for the response are compared to the first string. If they match, the domain name and port are replaced by those of the request, and the path is replaced by the second string.
- The following replaces "http://internal.server.com:80/docs/abc.html", the location of original response, by "http://webtob:8100/internal/abc.html".

```
RewriteRedirect = "http://internal.server.com:80/docs/ /internal/"  
RewriteRedirect = "http://internal.server.com:80/docs_kr/ /internal_kr/"  
RewriteRedirect = "http://internal.server.com:80/docs_ch/ /internal_ch/"
```

- Up to 16 items can be specified.

RewriteCookieDomain

- If the ReverseProxyGroupName is configured, this can be omitted and the setting in the REVERSE_PROXY_GROUP section can be used instead.
- Specifies the domain value to replace. If the domain in the response cookie header is identical to the specified value, it is replaced by the domain name of the request.
- Using the following configuration, if the WebtoB domain is webtob, the cookie of the original response "jsessionId=abc, domain=internal.server.com" is modified to "jsessionId=abc, domain=webtob".

```
RewriteCookieDomain = "internal.server.com"
```

RewriteCookiePath

- Type: Literal
- Range: Up to 255 characters
- If the ReverseProxyGroupName is configured, this can be omitted and the setting in the REVERSE_PROXY_GROUP section can be used instead.
- Specifies the two strings delimited by a blank space. If the path in the cookie header for the response is prefixed with the first string, the part is replaced by the second string.
- Using the following configuration, the cookie of the original response "jsessionid=abc, path=/jeus/application" is modified to "jsessionid=abc, path=/jeus_proxy/application".

```
RewriteCookiePath = "/jeus /jeus_proxy"
```

RewriteHtmlMaxSize

- Type: Numeric
- Unit: bytes
- Range: 1 to INT_MAX
- Default value: 10240
- If the ReverseProxyGroupName is configured, this can be omitted and the setting in the REVERSE_PROXY_GROUP section can be used instead.
- Specifies the maximum buffer size used to store the HTML response in order to modify its URL. If the response size is bigger than the specified size, the URL is not modified.

RewriteHtmlUrl

- Type: Literal
- Range: Up to 255 characters
- If the ReverseProxyGroupName is configured, this can be omitted and the setting in the REVERSE_PROXY_GROUP section can be used instead.
- Specifies the two strings used to replace the URL of an HTML document (Content-Type: text/html), which is an internal server response. If the URL is prefixed with the first string, the part is replaced by the second string. The domain name and port in the URL are replaced by the WebtoB server address of the request.
- A tag and its attributes that specify a URL can be defined in the HtmlUrl item.
- It is assumed that a meta tag always includes a URL.

```
<meta ... http-equiv="refresh" ... content="http://internal.server.com:80/docs/link3.html" ...>
```

```
RewriteHtmlUrl = "http://test2:80/ /proxy/",  
RewriteHtmlUrl = "/ /proxy/"
```

- Up to 32 items can be specified.

HtmlUrl

- Type: Literal
- Range: Up to 255 characters
- If the ReverseProxyGroupName is configured, this can be omitted and the setting in the REVERSE_PROXY_GROUP section can be used instead.
- Specifies the tag and attribute(s) names that include the URL of an HTML document. The first string is the tag name followed by attribute names.
- Up to 64 tags can be configured by specifying HtmlUrl multiple times.

```
HtmlUrl = "a href",  
HtmlUrl = "img src longdesc usemap"
```

StickySessionRoutingId

- Type: Literal
- Range: Up to 255 characters
- Used for sticky session routing when connecting to multiple web application servers in a reverse proxy group that is specified with ReverseProxyGroupName.
- Use the name of the engine that corresponds to the Sticky Session id(JSESSIONID) value set in the Set-Cookie response header by the internal server (WAS). For example, if the "JSESSIONID" value in the Set-Cookie response header is "Pl1xfBkEVbUu2cj20CUNIHJoWlMlU.xxx_servlet_engine1", use the value following the delimiter (period) which is "xxx_servlet_engine1".

```
StickySessionRoutingId = "xxx_servlet_engine1"
```

MinPersistentServerConnections

- Type: Numeric
- Range: 0 to INT_MAX
- Default value: 0
- Specifies the minimum number of connections required to maintain connection with an internal server after request processing is complete.
- A new connection with an internal server is created when there is a new request, and as long as the internal server doesn't terminate the connection it is maintained and reused for subsequent

requests.

- The `PersistentServerCheckUrl` setting is required to be able to send a ping, which must be sent to maintain the connection with an internal server.

MaxPersistentServerConnections

- Type: Numeric
- Range: 0 to `INT_MAX`
- Default value: 0
- Specifies the maximum number of connections required to maintain connection with an internal server after request processing is complete.
- When there is a new request, a new connection is created if all connections are currently processing other requests and the request is queued if the number of connections has reached the specified maximum limit.
- The `PersistentServerCheckUrl` setting is required to be able to send a ping, which must be sent to maintain the connection with an internal server.

PersistentServerCheckTime

- Type: Numeric
- Unit: Second
- Range: 0 ~ `INT_MAX`
- Default value: 30
- Specifies the interval for checking connection with an internal server in order to manage internal server connections.
- To maintain internal server connections, this must be set to a value less than the `KeepAliveTimeout` of the internal server. A single ping is sent to connections in the Ready state, and the connection is disconnected if no reply is received within the `PersistentServerCheckTime` period.
- If set to 0, no ping is sent and connections are maintained for the `KeepAliveTimeout` period of the external server.

PersistentServerTimeout

- Type: Numeric
- Unit: Second
- Range: 0 to `INT_MAX`
- Default value: 300
- Specifies the timeout for terminating a connection in the Ready state when the number of

internal server connections has reached the MinPersistentServerConnections.

- If set to 0, timeout is not applied to idle connections.

PersistentServerCheckUrl

- Type: Literal
- Range: Up to 255 characters
- Specifies to internally use the HTTP HEAD request as the ping message for maintaining internal server connections.
- Connections are maintained by using a ping message to check for the connection state after sending an HTTP request to the internal server and receiving an HTTP response. If the response message is not 200, the connection is terminated. An application must be implemented on the internal server to respond to the ping.

```
PersistentServerCheckUrl = "/ping.html"
```

ProxySslFlag

- Type: Boolean
- Default value: N
- Specifies whether to use SSL/TLS to connect to the internal server.
- If set, the PROXY_SSL section item can be set in ProxySslName.

ProxySslName

- Type: String
- Range: Up to 31 characters
- Specifies the PROXY_SSL section item when ProxySslFlag is set to 'Y'.

MaxWebSocketConnections

- Type: Numeric
- Range: 0 to INT_MAX
- Default value: 0
- Specifies the maximum number of connections when connection is upgraded from HTTP to WebSocket protocol.
- If set to 0, no limit is set on the number of WebSocket connections.

WebSocketSessionTimeout

- Type: Numeric
- Unit: Second
- Range: 0 to INT_MAX
- Default value: 0
- Specifies the timeout for WebSocket connection. **It is recommended to set this value based on the timeout setting of the WebSocket session on the internal server.**
- If set to 0, timeout check is not performed.

Options

- Type: Literal
- Range: Up to 255 characters
- Specifies the Cache option that is applied to the server.

Option	Description
{+ -}Cache	Determines the presence (+ -) of the content's cache.
DynamicServerAddress	Retries DNS resolution when the connection with the internal server fails.

ConnectRetryCount

- Type: Numeric
- Range: 0 to INT_MAX
- Default value: 3
- Specifies the number of DNS resolution retries to execute when TCP connection with the internal server fails.

ConnectTimeout

- Type: Numeric
- Range: 0 to INT_MAX
- Default value: 5
- Specifies the retry interval for TCP connection with the internal server.
- Retries DNS resolution and TCP connection attempts when this time is exceeded.

AccessName

- Type: String

- Range: Up to 31 characters
- Specifies the ACCESS section name to use to accept or reject a reverse proxy request from a specific IP address.

Headers

- Type: Literal
- Range: Up to 255 characters
- Specifies the name of a HEADERS section.
- Up to 15 items can be specified by separating each with a comma (,).

Compression

- Type: Literal
- Range: Up to 255 characters
- Specifies an item for compression before being sent in the response.
- Sets a MIME-type to compress, using the content-type of the response. The response is compressed using gzip before being sent to the client.
- Up to 32 MIME-types can be specified by separating each with a comma (,).
- Compression reduces network traffic but might impact server performance.

Avoid compressing files with low compression rates, such as .zip files or .jpeg images, as this places unnecessary load on the server without providing any benefits.



The compression function is applied only to the requests where Accept-Encoding is specified as GZIP or deflate in the HTTP Request Header.

CompressionMinSize

- Type: Numeric
- Range: 1 to INT_MAX
- Default value: 1
- Specifies the minimum size of a response that can be compressed.
- When the Content-Length response header value exceeds the specified threshold, the response can be compressed. Yet, this setting does not apply to chunked responses, as the response size cannot be measured for this type.

3.23.2. Example

The following is an example configuration of the REVERSE_PROXY section.

```
*REVERSE_PROXY
rproxy1
  ReverseProxyGroupName = "rpg_internal",
  VhostName = "vhost1",
  PathPrefix = "/internal/",
  RegExp = "\.(do|jsp)$",
  ServerAddress = "internal.server.com:80",
  ServerPathPrefix = "/docs/",
  RewriteRedirect = "http://internal.server.com:80/docs/ /internal/",
  RewriteCookieDomain = "internal.server.com",
  RewriteCookiePath = "/jeus /jeus_proxy",
  RewriteHtmlUrl = "http://internal.server.com:80/ /internal/",
  RewriteHtmlUrl = "/ /internal/",
  HtmlUrl = "a href",
  HtmlUrl = "img src longdesc usemap",
  RewriteHtmlMaxSize = 4194304,
  MinPersistentServerConnections = 10,
  MaxPersistentServerConnections = 20,
  PersistentServerCheckTime = 50,
  PersistentServerTimeout = 300,
  PersistentServerCheckUrl = "/ping.html",
  ProxySslFlag = Y,
  ProxySslName = proxy_ssl1,
  MaxWebSocketConnections = 20,
  WebSocketSessionTimeout = 300,
  Options = "-cache"
```

3.24. LOGLEVEL Section

The LOGLEVEL section configures the syslog level.

3.24.1. Configuration Items

The following is the configuration format of the LOGLEVEL section.

```
*LOGLEVEL
name # String[256]
  Level = String[256] # "INFO"
#Options = String[256], # PM.LIST
#RotateBySeconds = Numeric # 300 (0-)
#RotateByFileSize = Numeric # 0 (0-)
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the LOGLEVEL section configuration items.

LOGLEVEL section name

- Type: String
- Specifies the logger name. Logger is a WebtoB component.

Logger Name	Description
. ROOT	All WebtoB logger.
.hth HTH	For HTH logs.
.htl HTL	For HTL logs.
.wsm WSM	For WSM logs.
.server SERVER	For all server logs.
.server.cgis CGIS	For CGIS logs.
.server.phps PHPS	For PHPS logs.
.server.ssis SSIS	For SSIS logs.
.server.filters FILTERS	For FILTERS logs.

Level (Required)

- Type: String
- Default value: "INFO"
- Specifies the log level.
- Messages with a log level higher than the specified value are recorded. Set to one of: "TRACE" (lowest), "DEBUG", "INFO", "WARN", and "FATAL". The default level of all loggers is "INFO".
- TRACE level logs are recorded in a trace file set in the **WEBTOB_TRACE** environment variable, not the syslog file.

If the environment variable is not set, the messages are recorded in `$(WEBTOBDIR)/log/trace/processname-pid-date.trace`.

WEBTOB_TRACE can be set with one of the following format strings:

Format String	Description
%N%	Process name.
%P%	Process ID.
%D%	Date.

For example, the following means that HTH and CGIS messages are recorded in HTH.trace and CGIS.trace, respectively.

```
WEBTOB_TRACE = %N%.trace
```

Options

- Type: String
- Specifies the options for HTL log (.hth). Specify multiple options delimited with a comma
- The following describes the options used in ".hth" logger.

Option	Description
dcR	Records data sent by the client in the trace file.
dcW	Records data sent to the client in the trace file.
dsR	Records data sent by the server in the trace file.
dsW	Records data sent to the server in the trace file.

- The following is an example.

```
Options = "dcR,dcW,dsR,dsW"
```

RotateBySeconds

- Type: Numeric
- Unit: Second
- Range: 0 to INT_MAX
- Default value: 300 (If set to 0, this function is disabled.)
- Specifies the time (sec) since the creation of the current log file after which a new log file is created for a new log message.

RotateByFileSize

- Type: Numeric
- Unit: bytes
- Range: 0 to INT_MAX
- Default value: 0 (If set to 0, this function is disabled.)
- Specifies the file size threshold for log messages. If an existing log file surpasses this specified size, a new log file will be created.

3.24.2. Example

The following is an example configuration of the LOGLEVEL section.

```
*LOGLEVEL  
.hth
```

```
Level = "DEBUG"
.server.cgis
Level = "INFO"
```

3.25. HEADERS Section

The HEADERS section is used when a specific HTTP Header is added to a user request or a response. Headers items can be configured in the NODE, VHOST, and SERVER sections.

3.25.1. Configuration Items

The following is the environment configuration format of the HEADERS section.

```
*HEADERS
name # String[256]
    Action = String[256],
    #FieldName = String[256],
    #FieldValue = String[1024],
    #RegExp = Literal[512],
    #StatusCode = String[256] # (HTTP status code)
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the HEADERS section configuration items.

HEADERS section name

- Type: String
- Range: Up to 255 characters
- Specifies the section name.

Action (Required)

- Type: String
- Range: Up to 255 characters
- The following describes each Action type. Certain Action types have required FieldValue.

Action	Description
AddRequest	Adds a specified HTTP header in the request. Caution is advised as duplicates may occur if the specified header already exists in the request.

Action	Description
AddResponse	Adds a specified HTTP header in the response. Caution is advised as duplicates may occur if the specified header already exists in the response.
AddIfAbsentRequest	Adds a specified HTTP header when the request contains no HTTP headers.
AddIfAbsentResponse	Adds a specified HTTP header when the response contains no HTTP headers.
AppendResponse	Appends a specified FieldValue at the end of an existing header value. If there is no existing header, no action occurs.
EchoResponse	Adds the same header and value to the response as the request, if the the request contains a specified header. This action has no FieldValue.
SetResponse	Adds a specified HTTP header when the response contains no HTTP headers. If the header already exists, the header value is replaced by the specified FieldValue.
UnsetResponse	Deletes a specified header if it already exists in the response. This action has no FieldValue.

FieldName

- Type: String
- Range: Up to 255 characters
- Specifies the name of an HTTP Header to add.

FieldValue

- Type: String
- Range: Up to 1,023 characters
- Specifies the value of an HTTP Header to add.
- Use a string or symbols. For allowed symbols and information about the feature, refer to ["*LOGGING.Format Fields"](#).

RegExp

- Type: Literal
- Range: Up to 511 characters
- If the HTTP Request URL matches the specified Regular expression, add the specified Header.

StatusCode

- Type: String

- Range: Up to 255 characters
- Specifies when to add the header for a specific HTTP Status Code.

3.25.2. Example

The following configures a HEADERS section.

```
* HEADERS
header1      Action="AddIfAbsentRequest",
              FileName="Test_Header",
              FieldValue="test"

header2      Action="AddIfAbsentResponse",
              FileName="Test_Header",
              FieldValue="test"

header3      Action="AppendResponse",
              FileName="Test_Header",
              FieldValue=", AppendResponse"

header4      Action="EchoResponse",
              FileName="Test_Header"

header5      Action = "AddIfAbsentResponse",
              FileName = "Access-Control-Allow-Origin",
              FieldValue = "%{HEADER_FIELD}i"
```

3.26. PRECEDING_COMMAND Section

This section configures preceding commands such as specifying CPU Affinity by each process of WebtoB when starting WebtoB.

3.26.1. Configuration Items

The following is the configuration format of the PRECEDING_COMMAND section.

```
*PRECEDING_COMMAND
name      # String[256]
          Command = Literal[256]          # MULTI[100]
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the PRECEDING_COMMAND section configuration items.

PRECEDING_COMMAND section name

- Type: String
- Specifies the WebtoB process names.

Process name	Description
.hth HTH	HTH process
.htl HTL	HTL process
.wsm WSM	WSM process
.server SERVER	Server process

Command

- Type: Literal
- Range: Up to 255 characters
- Enters the commands to perform before process execution.
- The commands vary depending on the OS. The full path of the command located in the OS must be entered.

OS	Command
Linux	taskset
AIX	execrset
HP-UX	psrset
SunOS	psrset

- For HTH, up to 100 commands can be set. Except for HTH processes, only the first command is applied even if commands are configured multiple times.
- If multiple HTH processes are configured in the NODE section and the number of commands is greater than the number of HTH processes, commands are executed only up to the number of HTH processes. On the other hand, if the number of HTH processes is greater than the number of commands, commands are executed sequentially from top to bottom in a round-robin style.

If two commands are configured in the .hth file (as follows) and HTH is set to 5 in the NODE section, the first command is executed for hth1/hth3/hth5 and the second command for hth2/hth4.

```
Command = "/usr/bin/execrset -c 0 -e",  
Command = "/usr/bin/execrset -c 1 -e"
```

- When the process is restarted after an abnormal shutdown, the commands will be re-executed.

3.26.2. Example

The following is an example configuration of the PRECEDING_COMMAND section.

```
*PRECEDING_COMMAND
.wsm
  Command = "/usr/bin/taskset -c 1"
.htl
  Command = "/usr/bin/taskset -c 1"
.hth
  Command = "/usr/bin/taskset -c 2",
  Command = "/usr/bin/taskset -c 3",
  Command = "/usr/bin/taskset -c 4"
```

3.27. FILTER Section

The FILTER section specifies the FILTER section name that will use the Filter module. The Filter item is set in the NODE, VHOST, or SVRGROUP section.

3.27.1. Configuration Items

The following is the configuration format of the FILTER section.

```
*FILTER
name      # String[256]
          RealPath = Literal[256]          # $ENV, R.PATH
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the LOGGING configuration items.

FILTER section name

- Type: String
- Range: Up to 255 characters
- Specifies the configured FILTER section name.

RealPath (Required)

- Type: String
- Range: Up to 255 characters
- Specifies the path of the Filter module within the physical directory of the server.

3.27.2. Example

The following is an example configuration of the FILTER section.

```
*FILTER
sm_filter
    RealPath = "/usr/local/webtob/config/filter/wbSmISAPI.so"
```

3.28. USERLOGFORMAT Section

Common log format. The format defined in this section can be used in the LOGGING section.

3.28.1. Configuration Items

```
*USERLOGFORMAT
name # String[256]
    Format = Literal[256]
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the USERLOGFORMAT configuration items.

USERLOGFORMAT section name

- Type: String
- Range: Up to 32 characters
- Specifies the name of a log format to define.
- The specified name can be used as an argument for the `*LOGGING.Format`.

Format (Required)

- Type: String
- Range: Up to 255 characters
- Specifies the format for the log message. For more information about the message contents, refer to ["*LOGGING.Format Fields"](#).

3.28.2. Example

The following is an example of configuring the USERLOGFORMAT section.


```
*USERLOGFORMAT
format1
Format = "%h %l %u %t \"%r\" %s %b"
```

3.29. DISK_CACHE Section

Enables disk caching in WebtoB. When a file is in a remote host through a network such as NAS or NFS, this option saves the file in a local disk. Configuration items are disk path, size of a file to be cached, and so on.

3.29.1. Configuration Items

```
*DISK_CACHE
name # String[32]
CacheRoot = Literal[256] # "cache_root/", $ENV, R.PATH
DirLevels = Numeric # 0 (0 - )
MinFileSize = Numeric # 1 (1 - )
MaxFileSize = Numeric # 1000000
RefreshTime = Numeric # 1 (1 - )
RetryWaitCount = Numeric # 5
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the DISK_CACHE configuration items.

DISK_CACHE section name

- Type: String
- Range: Up to 32 characters
- Specifies the name of a disk cache.

CacheRoot (Required)

- Type: String
- Range: Up to 255 characters
- Specifies the path to the disk cache.
- Environment variables can be used as a value for this item. If set to a relative path, it will be converted to an absolute path that uses \$WEBTOBDIR.

DirLevels

- Type: Numeric
- Range: 0 to INT_MAX
- Default value: 0
- Specifies the depth of a directory to be created for disk cache.

MinFileSize

- Type: Numeric
- Unit: bytes
- Range: 0 to INT_MAX
- Default value: 1
- Specifies the minimum file size for disk cache.

MaxFileSize

- Type: Numeric
- Unit: bytes
- Range: 0 to LONG_MAX
- Default value: 1000000
- Specifies the maximum file size for disk cache.

RefreshTime

- Type: Numeric
- Unit: Seconds
- Range: 0 to INT_MAX
- Default value: 1
- Specifies the time period (in seconds) during which a disk-cached response remains valid.
- If set to 0, there is no time limit and disk-cached responses can remain valid indefinitely.

RetryWaitCount

- Type: Numeric
- Range: 0 to INT_MAX
- Default value: 5
- When multiple client requests are received for the same file, disk caching will be initiated by the

first received request.

- In this scenario, subsequent requests will wait and retry until the first request completes disk caching and sends the response. This setting defines the maximum number of retries for those subsequent requests.

3.29.2. Example

The following is an example of configuring the DISK_CACHE section.

```
*DISK_CACHE
dc1
  CacheRoot = "cache/node",
  DirLevels = 5,
  MinFileSize = 8193,
  MaxFileSize = 100000000,
  RefreshTime = 3600
```

3.30. LOG_HANDLER Section

Configuration for WebtoB access logs in a user-specified remote server. It includes server address, path to log files, and connection timeout.

3.30.1. Configuration Items

```
*LOG_HANDLER
name # String[32]
  ServerAddress = Literal[256],
  #ConnectTimeout = Numeric, # 3 (0-)
  #ReconnectTime = Numeric, # 10
  #Timeout = Numeric, # 60
  Resource = Literal[64],
  Protocol = Literal[64] # HTTP
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the LOG_HANDLER configuration items.

LOG_HANDLER section name

- Type: Literal
- Range: Up to 32 characters
- Specifies the name of LOG_HANDLER.

ServerAddress (Required)

- Type: Literal
- Range: Up to 255 characters
- Specifies the address of a remote server to store access logs.

```
ServerAddress = "remote.server.com:80"
```

ConnectTimeout

- Type: Numeric
- Unit: Seconds
- Range: 0 to INT_MAX
- Default value: 3
- Specifies the time period during which a connection request waits for connection to the remote server.

ReconnectTime

- Type: Numeric
- Unit: Seconds
- Range: 0 to INT_MAX
- Default value: 10
- Specifies the time period for waiting before attempting to reconnect to the remote server after a connection failure.

Timeout

- Type: Numeric
- Unit: Seconds
- Range: 0 to INT_MAX
- Default value: 60
- Specifies the time period for keeping the connection to the remote server alive without receiving requests for storing access logs. Once this timeout expires, the connection to the remote server will be terminated.

Resource (Required)

- Type: Literal

- Unit: Up to 63 characters
- Specifies the location to store access logs in the remote server.

Protocol (Required)

- Type: Literal
- Default value: HTTP
- Specifies the protocol to be used for sending requests for access logs in the remote server.

3.30.2. Example

The following configures a LOG_HANDLER section.

```
*LOG_HANDLER
lh1
  ServerAddress = "remote.server.com:8080",
  ConnectTimeout = 3,
  ReconnectTime = 10,
  Timeout = 60,
  Resource = "webtob/accesslog",
  Protocol = "HTTP"
```

3.31. IPGROUP Section

Configures a group of multiple IPs specified in the ACCESS section.

3.31.1. Configuration Items

```
*IPGROUP
name # String[32]
      IP = Literal[8960]
```



Refer to [Configuration Item Description Rules](#) for more information on symbols and details of the IPGROUP configuration items.

IPGROUP section name

- Type: String
- Range: Up to 32 characters

- Specifies the name of the IP group.
- The specified name can be used as an argument for *ACCESS.Allow or *ACCESS.Deny.

IP (Required)

- Type: String
- Range: Up to 8960 characters
- Defines the list of IPs to be grouped. Each IP is separated by a comma (,).

3.31.2. Example

The following example configures an IPGROUP section.

```
*IPGROUP
ipgroup1
    IP = "localhost, 123.123.123.123"

*ACCESS
access1
    Allow = "100.100.100.100, ipgroup1"
```

4. Advanced Configuration

This chapter describes advanced settings including dynamic contents and virtual hosting, and how to configure Tmax and JEUS integration.

4.1. Dynamic Content

If the web server is ready, HTML services are available to use. Web browsers can access HTML documents in WebtoB's Document Root Directory.

This chapter describes the configurations that enable CGI, SSI, and PHP. Also included are examples of configuring dynamic content pages.

4.1.1. CGI

In the world wide web, Common Gateway Interface(CGI) is the basic language used to create web pages to communicate information using HTML. Because HTML can only transmit data from the server to the client, it is limited to static, one-dimensional web pages that lack user-server interaction. One method that addresses the weaknesses of HTML is the use of external programs. CGI is the interface used between external programs and web servers.

An external program is called a CGI program or a CGI script. For example, creating a guest book application that can receive comments from visitors is impossible using only HTML. This example and numerous others involving bilateral communication between client and server highlights the need for external programs. C/C++, Perl, UNIX Shell, or Tcl/Tk are common languages used to create CGI programs.

There are many CGI programs that can make a website interactive. A visitor counter, guest book, bulletin board system, web chat rooms, search engine, banners, uploadable file archives, and a form mail system are just a handful of examples of CGI programs.

A well written CGI program is easily transferable to other web servers. WebtoB is one such web server that a CGI program can easily run on.

While there are many ways to create CGI programs, the details will not be discussed in this document. This chapter describes how to configure the use of CGI in WebtoB.

CGI Configuration

In order to use CGI, the following sections must be configured.

- **SVRGROUP section**

Add the following content to the SVRGROUP section in order to use CGI. The content is later used in the SERVER section to configure server groups.

The following is an example configuration of the SVRGROUP section. If the SvrType item is set to CGI, the server group processes CGI.

```
*SVRGROUP
cgig      NodeName = mynode,
          SvrType = CGI
```

- **SERVER section**

Specifies the server process in the SERVER section. The following is an example configuration of the SERVER section.

```
*SERVER
cgi      SvgName = cgig,
         MinProc = 10,
         MaxProc = 10
```

In the example, the server process named `cgi` is configured. The server group is specified by setting `SvgName` to `cgig`. The server process named `cgi` is now associated with a server group named `cgig`. Since the `SvrType` of `cgig` is `CGI`, the `cgi` server provides CGI services.

- **URI section**

In the URI section, a specific URI can be mapped to target servers. The following example shows that the URI of `/cgi-bin/` is processed by CGI servers.

```
*URI
uri1     Uri = "/cgi-bin/",
         SvrType = CGI
```

- **ALIAS section**

In the ALIAS section, the physical directory that processes a URI can be configured to a location other than to `Docroot`.

The following is an example of ALIAS section configuration.

```
*ALIAS
alias1   Uri = "/cgi-bin/",
         Realpath = "${WEBTOBDIR}/cgi-bin/"
```



Refer to [Environment Configuration Example](#), and [CGI Application Example](#) for more information on configuration and examples respectively.

4.1.2. SSI

This section describes how to use and configure SSI.

SSI Usage

SSI, short for Server Side Includes, can be used to create dynamic web pages. SSI can add Header files or add a function that can automatically adjust the last modified time of a document.

SSI can insert a 'command' in an HTML document. A server parses the SSI document for SSI commands and processes the commands. For example, a SSI command that modifies the last modification time is included in an HTML document; the server parses the command and modifies the last modified time.

Because HTML files that include SSI commands slow performance, SSI is disabled in WebtoB by default. To enable SSI, add a SSI server to the SERVER section in WebtoB environment file so that SSI is processed in another server.

The following is an example of the configuration to enable SSI.

```
*SVRGROUP
ssig      NodeName = mynode, SvrType = SSI

*SERVER
ssi      SvcName = ssig, MinProc = 10, MaxProc = 10
```

SSI Commands

All SSI commands in HTML documents are written in the form of HTML comments.

SSI commands are generally wrapped in HTML comments "`<!--... -->`" as shown in the following. In the example, `arg1` and `arg2` are parameters and `value1` and `value2` are their respective values.

```
<!--#command arg1="value1" arg2="value2" ... -->
```

The following is an example of `flastmod`, a command that outputs the modified time. 'file' is a parameter and "nextel.html" is the parameter value.

```
<!--#flastmod file="nextel.html"-->
```

SSI command execution is variable name dependent. The following is an example.

```
<!--#flastmod virtual="/" -->
```

The `echo` command retrieves the last modified time of the server web site. When the SSI command is executed, environment variables are set. These variables include CGI variables such as `REMOTE_HOST`, `DOCUMENT_NAME`, and `LAST_MODIFIED`.

```
<!--#echo var="LAST_MODIFIED" -->
```

4.1.3. PHP

This section describes how to use and configure PHP.

PHP Usage

PHP is a scripting language that is similar to Perl. PHP is popular due to its simplicity and performance advantages. PHP surpasses Perl in performance, development speed, and extensibility. Moreover, PHP is supported in Linux, UNIX, and Win32, which allows for web application development independent of the web server platform.

PHP can do everything that CGI can do such as retrieving form data, creating dynamic web pages, and managing web browser cookies.

PHP Configuration

In order to use PHP in WebtoB, installation is required. Using PHP is similar to using HTML or CGI, making it is easy to apply.

- **SVRGROUP section**

A PHP group must be defined in the SVRGROUP and SERVER sections.

In the SVRGROUP section a ScriptLoc item points to the path of the php executable module. The path must be a relative path from `${WEBTOBDIR}`. Use the ScriptArgs item to configure the options for PHP executable modules. Files including php.ini must be set to an absolute path.

In the following example, the module path specified in ScriptLoc is `"${WEBTOBDIR}/cgi-bin/php"` and the ScriptArgs is set to an absolute path.

```
*SVRGROUP
phpg      NodeName = mynode,
          ScriptLoc = "/cgi-bin/php", # refers to "${WEBTOBDIR}/cgi-bin/php"
          ScriptArgs = "-c /php_conf/php.ini", # executable module options for php execution
          SvrType = PHP

*SERVER
php       SvgName = phpg, MinProc = 10, MaxProc = 10
```

If a php3 module is required, additional configuration is necessary. Unlike php4 (.php), php3 uses the ".php3" extension. The use of php3 must be configured in the EXT section. The following is an example of the EXT section configuration.

```
*EXT
php3     Mimetype = "application/x-httpd-php3",
          SvrType = PHP
```

PHP Example


The following verifies PHP is installed and was configured successfully in WebtoB.

1. Create "phpinfo.php" in \${WEBTOBDIR}/docs.

```
<?
  Phpinfo();
?>
```


2. Enter the following URL into the browser. The page displays environment information about the installed version of PHP.

http://<IP address>:<PORT>/phpinfo.php

PHP Version 4.1.2


System	HP-UX tmaxh2 B.11.11 U 9000/800 1151494615 unlimited-user license
Build Date	Mar 22 2004
Configure Command	'./configure' '--prefix=/data2/php412' '--without-mysql'
Server API	CGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/data2/php412/lib
ZEND_DEBUG	disabled
Thread Safety	disabled

This program makes use of the Zend Scripting Language Engine:
 Zend Engine v1.1.1, Copyright (c) 1998-2001 Zend Technologies



PHP 4.0 Credits

Configuration

PHP Core

Directive	Local Value	Master Value
allow_call_time_pass_reference	On	On
allow_url_fopen	1	1
always_populate_raw_post_data	0	0
arg_separator.input	&	&
arg_separator.output	&	&
asp_tags	Off	Off
auto_append_file	no value	no value
auto_prepend_file	no value	no value

PHP Test Page

4.2. Virtual Hosting

This section describes Virtual Hosts and how to use them.

4.2.1. Virtual Host Structure

Virtual host is a feature available on web browsers that support HTTP 1.1. Virtual host allows a single web server to work as multiple web servers.

For example, a web newspaper service called “WebtoB Times” is created. This service can operate with a single IP, a domain name, and a single web server. However, multiple domain names can be beneficial so that services can be personalized without having to purchase additional IP addresses. Moreover, multiple domains will allow for easier management of services and future expansion.

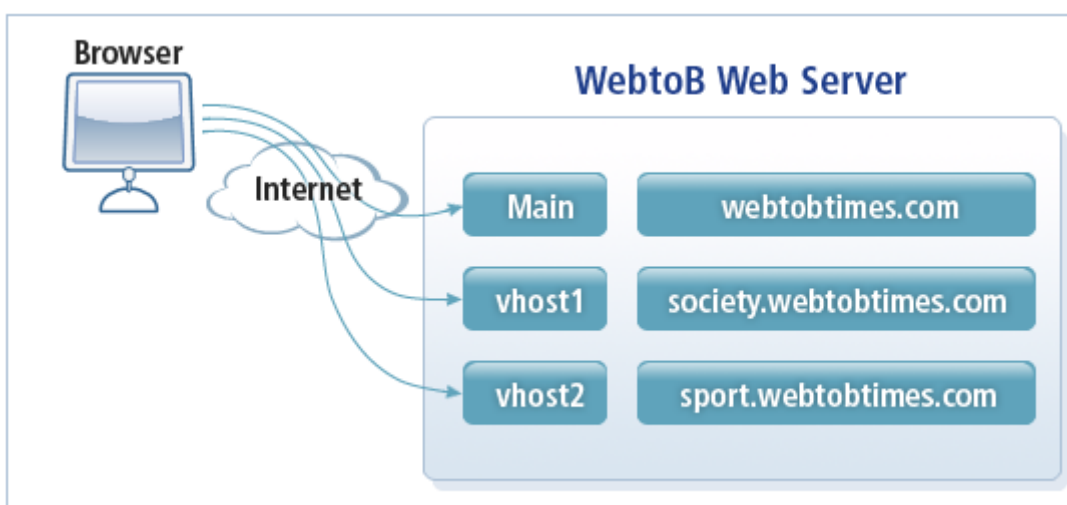
The following domain names are offered to customers to specialize services.

- webtobtimes.com: Main page
- society.webtobtimes.com: Covers social issues
- sports.webtobtimes.com: Covers sports issues

Per the previous sample domain names, a virtual host feature can divide domain names to create HTML documents and to provide other services. To support the new domains, the web server can create an independent virtual host for each society and sports domain. The web server can also have separate document paths and configurations for each virtual host. Supporting multiple domains on a single IP address is called **Name Based Virtual Host**.

Alternatively, there is an **IP Address Based Virtual Host** that uses multiple domains and multiple IP addresses. However, this is less popular because it is difficult for a server to have multiple IP addresses.

The following is a Virtual Host structure. As shown, a single IP address and a single web server can service multiple domains. The virtual host of WebtoB can also separately process services referred by different names in the same IP.



Virtual Host Structure

The following is an example of the WebtoB virtual host configuration of the "WebtoB Times" example from the previous section. For efficient web operation, use VHOST section as follows.

```
*VHOST
society
  Docroot = "society_docs/",
  NodeName = webtob,
  Hostname = "society.webtobtimes.com",
  Port = "8080",
  IconDir = "society_icons/",
  LOGGING = "society_accesslog",
  ERRORLOG = "society_errorlog"

sports
  Docroot = "sports_docs/",
  NodeName = webtob,
  Hostname = "sports.webtobtimes.com",
  Port = "8080",
  IconDir = "sports_icons",
  LOGGING = "sports_accesslog",
  ERRORLOG = "sports_errorlog"
```

4.2.2. Mass Virtual Host

When deploying mass virtual hosts, configuring each virtual host is not necessary.

- **Docroot Parameter Configuration**

This feature is suitable when the DocRoot parameter is the only parameter modified in the mass virtual hosts. The DocRoot parameter of each virtual host is defined by replacing the server domain name value with the directory configuration pattern.

```
Docroot = <literal>    (Default : "docs/")
```

E.g., A client connects to <http://www.tmax.co.kr>. To set the virtual host's docroot to "\${WEBTOBDIR}/docs/www/", configure as follows. Similarly, if a client connects to <http://webtob.tmax.co.kr>, DocRoot is set to "\${WEBTOBDIR}/docs/webtob/".

```
*VHOST
TmaxSoft
  NodeName = tmax,
  Hostname = "www.tmax.co.kr"
  HostAlias = "webtob.tmax.co.kr"
  Port = "80",
  Docroot = "${WEBTOBDIR}/docs/%1/"
```

- **Realpath Parameter Configuration**

Specify the Realpath from ALIAS section when using mass virtual hosts. Set the Realpath parameter by replacing the server domain name value with the directory configuration pattern.

```
Realpath = <literal>    (Default : mandatory)
```

Similar to configuring DocRoot, "%1" is replaced with the first field of the domain requested by the client.

```
*ALIAS
alias1      Uri = "/cgi-bin",
            Realpath = "${WEBTOBDIR}/docs/%1/cgi-bin/"
```

When using mass virtual hosts, the directory configuration pattern can be configured dynamically by using the following directives.

Directive	Description
%p	Changes into the requested port number.
%n	Changes into the Hostname or the Nth element of the IP address. If n is set to 0, the entire string is used. If the value starts with a minus symbol (-), count from the end of the Hostname or IP address. If a plus symbol (+) follows, the rest of the Hostname or IP address is used.
%n.m	Changes into the Mth character of the Nth element. Minus (-) or plus (+) symbols can be used as previously described.
%%	Change into a single percent symbol (%).



When a %-directive is used, an absolute path must be configured for Docroot/Realpath. When an environment variable `${env}` is used, a %-directive can be only used when the changed path is an absolute path.

4.3. Log

This section describes log files and their formats.

4.3.1. Log File

WebtoB stores all logs related to web service requests and services it provides into a log file specified in the environment file. The log file can be analyzed for various client and request data.

The log file, stored in the location specified in the environment file, can be stored anywhere in the machine. Improper log maintenance may cause performance and capacity issues for the server.

- System load is generated when an event is recorded to the log. Although the logging mechanism within WebtoB is efficient, physical limitations, such as disk speed, may hinder system performance.
- While system load is insignificant for few clients, a large scale website can generate gigabytes worth of logs daily. Logs are not recorded if there is insufficient disk capacity, thus requiring

maintenance.

Log data is critical to system administrators. Web services can become more effective by mining user information. E.g., An online shopping mall administrator can obtain customer information to help increase sales. Because of the importance of logs, WebtoB administrators store log files to a specified location for management.

WebtoB creates three types of logs: an access log that is related to user requests; an error log that is related to error messages generated during processing requests; and a system log for the WebtoB system.

Type	Description
Access Log	<p>Access Log File, also known as Transfer Log File, records general activities and client actions in the web server. Information related to user requests and server responses is also recorded. This recorded information is very useful and important.</p> <p>Logs can be enabled in WebtoB's NODE, SVRGROUP, and VHOST sections. If logs are enabled in multiple sections, priority follows the following sequential order: SVRGROUP, VHOST, and NODE. The section where an event was logged has the highest priority.</p>
Error Log	Error log records all errors generated while the user request is being processed.
System Log	System log records events generated during the operation of WebtoB. This data is useful when troubleshooting problems in the web server. Whenever a critical task is being performed, logs are saved so that data is readily available to troubleshoot the system. The system log file can be configured in the NODE section.

4.3.2. Log Format

WebtoB can save access logs in multiple formats by configuring the Format item in the LOGGING section.

CLF (Common Logfile Format) is a standardized text format used by many administrators. Because the format is standardized and originated from NCSA, most web servers support this format. In WebtoB, CLF is the default log format.

4.3.3. Common Log Format

The Common Logfile Format defined in W3C is as follows:

```
remotehost rfc931 authuser [date] "request" status bytes
```

Define log parameters in the following order. Store the log in the location specified in the WebtoB environment file.

Item	Description
remotehost	Remote hostname. (If the DNS hostname is unavailable or DNSLookup is off, enter an IP address.) Refer to http://www.w3.org/Daemon/User/Config/General.html-%20DNSLookup for more information on Remote Host.
rfc931	Name of remote log
authuser	Name of authorized user
[date]	Date and time of the request
"request"	The request line exactly as it came from the client
status	HTTP Status Code returned to the client Refer to http://www.w3.org/Protocols for more information on Status Codes.
bytes	Size of transmitted document

The following is an example of the log files stored in WebtoB.

```
143.248.148.42 - - [13/Feb/2001:16:46:13 +0900] "GET / HTTP/1.1" 304 -
143.248.148.42 - - [13/Feb/2001:16:46:14 +0900] "GET /index_pb.gif HTTP/1.1" 304 -
143.248.148.42 - - [13/Feb/2001:16:46:18 +0900] "GET /index.html HTTP/1.1" 200 7118
143.248.148.42 - - [13/Feb/2001:16:46:18 +0900] "GET /usage.png HTTP/1.1" 304 -
143.248.148.42 - - [13/Feb/2001:16:46:37 +0900] "GET /usage_200102.html HTTP/1.1" 404 148
143.248.148.42 - - [13/Feb/2001:16:47:21 +0900] "GET /index.html HTTP/1.1" 304 -
143.248.148.42 - - [13/Feb/2001:16:47:21 +0900] "GET /usage.png HTTP/1.1" 200 2509
143.248.148.42 - - [13/Feb/2001:16:47:24 +0900] "GET /usage_200102.html HTTP/1.1" 404 148
143.248.148.42 - - [13/Feb/2001:16:47:55 +0900] "GET /index.html HTTP/1.1" 304 -
143.248.148.42 - - [13/Feb/2001:16:47:55 +0900] "GET /usage.png HTTP/1.1" 304 -
143.248.148.42 - - [13/Feb/2001:16:47:57 +0900] "GET /usage_200102.html HTTP/1.1" 200 30798
```

In the example, a "-" in a field indicates missing data. The first column in the example displays the IP address of the connected client. In the following hyphens (-) represent the missing rfc931 and AuthUser data. In the square bracket [], user access time is displayed.

The information wrapped in double quotes is the request line from the client. The xnxt column is the HTTP status code of the response to the client. The last column is size of the object transmitted to the client from the server in bytes.

4.3.4. Configuration

In WebtoB, the log format is configured in the Format item of the LOGGING section.

In order to record logs, configure the environment as follows:

```
*NODE
webmain
  WebtoBDir = "$WEBTOBDIR",
  SHMKEY = 69000,
```



```

DOCRROOT = "docs/",
Port = "5469",
Logging = "accesslog",
    ErrorLog = "errorlog"

```

***LOGGING**

```

accesslog
    Format = "default",
    Filename = "log/access.log",
    Option = "Sync"
errorlog
    Format = "",
    Filename = "log/error.log",
    Option = "Sync"

```

The Logging item in the NODE section specifies the access log, while the ErrorLog item specifies the error log. In the LOGGING section of the configuration, the accesslog and errorlog sections define the log format, filename, and options of their respective logs.

The following directives can be used in format items to customize access logs.

Format	Description
DEFAULT	Default log file format. (Format string: "%h %t \"%r\" %s %b %D")
COMMON	Common log file format. (Format string: "%h %l %u %t \"%r\" %s %b")
COMBINED	Combined log file format. (Format string: "%h %l %u %t \"%r\" %s %b \"%{Referer}i\" \"%{User-Agent}i\"")
%a	Displays the IP address of the machine that sent the request. Same as %h.
%b	Displays the byte of the response object.
%c	Displays whether the response is in the WebtoB internal cache (specified as "hc").
%{_attr_name}_C	Displays '_attr_name_' value of the HTTP Request Cookie Header.
%d	Displays the time a response was sent.
%D	Displays the time spent processing the request. (unit: millisecond)
%{ENV_NAME}e	Displays the environment variable ENV_NAME.
%g	Displays the request identifier used internally by WebtoB.
%h	Displays the IP address of the machine that sent the request. Same as %a.
%H	Displays the HTTP version used.
%{HEADER_FIELD}i	Displays the HEADER_FIELD Header value of the HTTP request.

Format	Description
{_id_name_}	Displays the ID that is used internally when a request is forwarded to JEUS for processing. <ul style="list-style-type: none"> ◦ Displays the Client ID if _id_name is JSVCid. ◦ Displays the request sequence if _id_name is JSVReqSeq.
%m	Displays the HTTP request method.
%p	Displays the server port that received the request.
%q	Displays the query value of HTTP request.
%r	Displays the entire request line of the HTTP request.
%R	Displays all request lines of the HTTP request. Displays request lines that are changed by CheckURL or URLRewrite function.
%s	Displays the HTTP Status Code used in the response.
%t	Displays the time when the request processing completed.
%T	Displays the time spent processing the request. (unit: second)
%u	Displays the user name used in HTTP authentication.
%U	Displays the HTTP Request URI.
%v	Displays the Host Header field value.
%z	If the response is compressed, displays the response size and compression rate before and after the compression.

"%h %l %u %t \"%r\" %s %b" corresponding to CLF, is set as the default value. In the previous example, the accesslog was set to default CLF format.

Including the %d option, several date and time types provided by the strftime function can be specified within {}. The default value is recorded in the same format as the %t directive: "[13/Feb/2001:16:47:24 +0900]".

The following format can be specified using directives.

```

custom_log1
    Format = "%h %l %u %t \"%r\" %s %b",
    Filename = "log/custom1.log"
custom_log2
    Format = "[%Y.%m.%d %H:%M:%S] %a \"%r\" %s %T",
    Filename = "log/custom2.log"

```

4.4. Integration with Tmax

WebtoB utilizes the ASP library to integrate with TmaxSoft TP-Monitor Tmax. Tmax must be installed before integration.

In WebtoB, only a client program can be created using WBAPI, while server programs are used in

Tmax. I.e., the WebtoB client program sends a request to the server program in Tmax. After the client program receives a response, the response is transferred to the WebtoB client (web browser).

4.4.1. Tmax Integration Configuration

Because integration with Tmax uses WBAPI, creating an environment file is the same.

The following example is an environment file.

<apsl.m>

```
*DOMAIN
webtob_apsl

*NODE
tmaxi2
    WebtoBDir="$WEBTOBDIR",
    SHMKEY = 74125,
    Docroot = "docs/",
    Port = "8797",
    HTH = 1,
    Logging = "log1",
    AppDir = "ap/",
    ErrorLog = "log2"

*SVRGROUP
htmlg    NodeName = "tmaxi2", SvrType = HTML
webapg   NodeName = "tmaxi2", SvrType = WEBSTD

*SERVER
wbsvrinit SvgName = webapg, MinProc = 10, MaxProc = 10
wbquery   SvgName = webapg, MinProc = 10, MaxProc = 10
wbsession SvgName = webapg, MinProc = 10, MaxProc = 10

*SERVICE
test      SvrName = wbsvrinit
query    SvrName = wbquery
wbsession SvrName = wbsession

*URI
uri1     Uri = "/svct/", SvrType = WEBSTD

*LOGGING
log1
    Format = "default",
    Filename = "log/access.log",
    Option="Sync"
log2
    Format = "",
    Filename = "log/error.log",
    Option="Sync"
```

4.4.2. Usage Example

Refer to [Integration with Tmax](#) for more information on the apsl.m client program from the previous example.

4.5. Integration with JEUS

WebtoB can integrate with JEUS, TmaxSoft's next generation web application server, which includes advanced features not available on web servers. JEUS, well suited for large data transactions and user management, can help E-commerce web sites to ensure security and transactions to clients.

The integration procedures for WebtoB Standard and JEUS are nearly identical to the integration procedures for WebtoB Enterprise and its servlet engine.

The following files must be configured in order to integrate WebtoB with JEUS.

- WebtoB environment file (e.g., http.m)
- JEUS environment file: WEBMain.xml

The following values must be identical in the WebtoB and JEUS configurations.

- WebtoB-JEUS communication port number
- Number of HTH processes
- Server Name (WebtoB's JSV Server Name and JEUS's Registration ID)
- Number of Processes (WebtoB's MinProc/MaxProc and JEUS's Thread-pool)

4.5.1. WebtoB and JEUS Integration

In WebtoB and JEUS integration, JEUS connects to WebtoB. For best results, WebtoB should service the external network and JEUS the internal network. This is because internal network firewall is very strict with in-bound connections but tolerant with outbound connections.

When connecting JEUS to WebtoB, it is recommended to first start WebtoB and then JEUS. JEUS will attempt to connect to WebtoB during start up. If WebtoB is not online, JEUS will continually try to connect to WebtoB at regular intervals and output error messages to the JEUS log. Once the a successful connection has been made, a connection confirmation message is outputted to the JEUS log. With the successful connection, user requests, such as Servlet and JSP, are sent to JEUS through the connection.

The WebtoB and JEUS connection is persistent and does not disconnect unless a network error or a server failure occurs.

4.5.2. JEUS 6 Integration Configuration (Standard+)

WebtoB Servlet Engine is sufficient for a business that requires simple online business services and

logic. However, if a large-scale online service system is required, an integrated WebtoB and JEUS system is recommended.



This section describes integration with JEUS version 6 and higher. For information about JEUS 7 integration, refer to [Using Embedded Servlet Engine for Enterprise \(JEUS 7\)](#).

WebtoB Environment File Configuration

• **NODE Section**

Specifies the number of HTH processes and the JsvPort that communicates with JEUS.

```
*NODE
    ...
    HTH = 1,
    JsvPort = 9999,
    ...
```

The number of HTH processes specified must be the same as the <hth-count> item value, which is a subitem of <webtob-listener> in the JEUS environment file WEBMain.xml.

JsvPort is a port number that connects to a web container. The web container is unrelated to port numbers that receive requests as an actual web browser. This port must be same as the <port> item value, which is a subitem of <webtob-listener> in WEBMain.xml.

• **SVRGROUP Section**

Specifies a new server group with a SVRTYPE of JSV.

```
*SVRGROUP
jsvg      SvrType = JSV
```

• **SERVER Section**

By connecting a service to a web container, specifies the server.

```
*SERVER
MyGroup   SvgName = jsvg, MinProc = 4, MaxProc = 10
```

The server named MyGroup is associated with the server group jsvg, which is defined in the SVRGROUP section.

MinProc is the minimum number of connections to the web container. MaxProc is the maximum number of connections allowed to the web container. MinProc and MaxProc need to be greater than or equal to the value in <thread-pool>, which is a subitem of <webtob-listener> in WEBMain.xml.

- **URI Section**

URI is the identifier WebtoB uses to decide which server to use. In the URI section, the server name (MyGroup), server type (JSV), and the URI to be serviced are specified.

The following is an example configuration that performs a service on the JSV server (MyGroup) for URI "/examples/".

```
*URI
uri1    Uri = "/examples/",
        SvrType = JSV,
        SvrName = MyGroup
```

JEUS 6 Environment File Configuration

The following path found in **WEBMain.xml** must be modified for the WebtoB connection configuration.

```
$JEUS_HOME/config/<node_name>/<node_name>_servlet_<engine_name>/
```

<WEBMain.xml>

```
<webtob-listener>
  <listener-id>webtob1</listener-id>
  <port>9999</port>
  <webtob-address>localhost</webtob-address>
  <registration-id>MyGroup</registration-id>
  <hth-count>1</hth-count>
  <thread-pool>
    <min>10</min>
    <max>10</max>
    <step>0</step>
    <max-idle-time>30000</max-idle-time>
  </thread-pool>
</webtob-listener>
```

In WEBMain.xml, the items under <webtob-listener>, which is under <webserver-connection>, must be modified. The following are descriptions for each item.

Tag	Description
<port>	The port used to communicate between WebtoB Servlet Engine Web Container and WebtoB. The specified port value must be the same as the JsvPort value in the NODE section of the WebtoB configuration file.
<webtob-address>	Specifies the IP address that connects to WebtoB.
<registration-id>	Used when the first connection to WebtoB is made. <registration-id> must be the same as the server name specified under the SERVER section (E.g., MyGroup).

Tag	Description
<hth-count>	Set as the same value as HTH of the NODE section of WebtoB. An actual connection is made for each HTH, therefore this value must match the number of HTH.
<thread-pool>, <min>, <max>	The minimum and maximum number of threads of Thread Pool. The value should be smaller than or equal to the value of MinProc/MaxProc for the MyGroup server defined in the WebtoB environment file.

A context group can be connected to many web servers. Each web server must have a communication port. Being connected to many web servers is useful when multiple web servers are in different nodes or when receiving requests from multiple types of web servers.

Servlet Engine Starting Up And Shutting Down

WebtoB and WebtoB Servlet Engine can be started by using the steps listed in the following into a command console.

1. Compile the WebtoB environment file.

```
C:\> wscfl -i http.m
```

2. Start WebtoB.

```
C:\>wsboot
```

3. Start JEUS Manager.

```
C:\>jeus
```

4. In order to boot JEUS, use the jeusadmin console tool after authentication.

```
C:\>jeusadmin webmain
Login name>administrator
Password>
JEUS 6.0 Jeus Manager Controller
webmain>
```

webmain is the node that boots JEUS. Authorized user name and password are required to use the jeusadmin command.



The default ID is administrator and the default password is the one entered at JEUS installation.

5. Start up JEUS, which automatically integrates with WebtoB.

```
webmain> boot
```

The booting process is a two-step procedure.

- a. Start JEUS service and then its engine.
- b. During the JEUS server start-up process, the previous command is passed to JEUS Manager.

6. Stop JEUS service and its engine using the following command.

```
webmain> down
```

7. Exit JEUS server using the following command.

```
webmain>jeusexit
```

Example

Refer to [WebtoB and JEUS Integration Configuration](#).

4.5.3. Using Embedded Servlet Engine for Enterprise (JEUS 7)

The servlet engine provided by WebtoB Enterprise or above can be configured in the same way as when integrating JEUS. However, the aforementioned item names for JEUS 6 may be different from the current Servlet Engine version, which is for JEUS 7.

WebtoB Environment File Configuration

Configuration is identical to "WebtoB environment file configuration" in [JEUS 6 Integration Configuration \(Standard+\)](#).

Embedded Servlet Engine Environment File Configuration

For WebtoB connection configuration, **domain.xml** in the following path must be modified.

```
${WEBTOBDIR}/jeus/domains/domain1/config/domain.xml
```

<domain.xml>

```
<webtob-connector>  
  <name>webtob</name>  
  <network-address>  
    <port>9999</port>
```



```

    <ip-address>localhost</ip-address>
  </network-address>
  <hth-count>1</hth-count>
  <thread-pool>
    <number>10</number>
  </thread-pool>
  <registration-id>MyGroup</registration-id>
  <output-buffer-size>8192</output-buffer-size>
  <reconnect-interval>5000</reconnect-interval>
</webtob-connector>

```

Items in the <webtob-connector> section must be modified in domain.xml. Descriptions for each item are as follows:

Tag	Description
<network-address>,<port>	Port number for connecting WebtoB Servlet Engine to WebtoB. Must be same as the JsvPort value in the NODE section of the WebtoB environment file.
<network-address>,<ip-address>	WebtoB IP address to connect to.
<hth-count>	Must be set to the hth-count value of the HTH in the NODE section of WebtoB environment file because a connection is made with each HTH.
<thread-pool>,<number>	The number of threads that must be maintained in the thread pool. The value must be less than or equal to the value of MaxProc for the MyGroup server in the WebtoB environment file.
<registration-id>	Used when the first connection with WebtoB is made. This value needs to be same as the server name specified in the SERVER section (E.g., MyGroup).

Starting and Shutting Down an Embedded Servlet Engine

In the commandline mode, start WebtoB and its embedded servlet engine in steps. For information about configuration, refer to the JEUS7 fix#4 manual.

1. Compile the WebtoB environment file.

```
C:\> wscfl -i http.m
```

2. Start WebtoB.

```
C:\>wsboot
```

3. Use the following command to start JEUS DAS.

```
C:\>startDomainAdminServer -u administrator -p <password>
```

4. Use the following command to start JEUS MS.

```
C:\>startManagedServer -domain jeus_domain -server server1 -u administrator -p <password>
```

Once JEUS MS boots up successfully, open "http://localhost:8088/examples/" in the web browser to check that "examples" has been properly deployed.

5. Use the following command to connect to jeusadmin.

```
C:\>jeusadmin -u administrator -p <password>
Attempting to connect to 127.0.0.1:9736.
The connection has been established to Domain Administration Server adminServer in the domain
jeus_domain.
JEUS7 Administration Tool
To view help, use the 'help' command.
[DAS]jeus_domain.adminServer>
```

Enter the JEUS Admin user name and password. The default user name is 'administrator' and the password is the one that was entered when installing the Servlet Engine.

Once JEUS starts successfully, a prompt will be displayed and is ready for command input.

6. Use WebAdmin to manage the Servlet Engine.

```
http://localhost:9736/webadmin
```

Go to the previous URL from the web browser.

Enter the JEUS Admin user name and password, and then click [Login]. The default user name is 'administrator' and the password is the one that was entered when installing JEUS.

7. Use the following command to terminate JEUS service and engine.

```
[DAS]jeus_domain.adminServer>local-shutdown
```

8. Use the following command to terminate jeusadmin.

```
exit
```

Example

Refer to [WebtoB and JEUS Integration Configuration](#) for more information.

4.6. Connecting to Other Web Application Servers

WebtoB can be integrated with other web application servers than JEUS using Reverse Proxy. A web application server generally has an HTTP Listener, and WebtoB uses reverse proxy setting to connect with another web application server's HTTP Listener.

Multiple web application servers can be configured using the Reverse Proxy Group setting provided in WebtoB Enterprise edition or higher.

Integration of WebtoB with web application servers can be configured in the following files.

- WebtoB environment file (e.g., http.m)
- WAS environment file (e.g., WEBMain.xml)

Check the following items for WebtoB-web application server connection. (Refer to the target web application server's manual for its configuration.)

- Target server's HTTP Listener setting (Listen IP, Port, thread count).
- Application setting for PING check when using persistent connection.
- Routing ID when using Sticky Session routing.
- HTTP Listener's SSL setting when using encrypted communication.

4.6.1. Using Reverse Proxy

WebtoB uses the Reverse Proxy setting to connect to another web application server. The setting can also be used when WebtoB runs in an external network while the web application server runs in an internal network. In this setting, WebtoB can rewrite a response received from the internal server.

WebtoB attempts to connect to the target server when a client initiates an HTTP request. If the server is not running, it returns a 503 error to the client. If the connection is successful, WebtoB forwards the request to the server, receives a response from the server, and then sends it to the client.

WebtoB can reuse the connection by using the persistent connection setting, and this requires the PING check setting between the two servers. If the Min and Max settings are configured, connection is managed according to the number of concurrent requests and SSL communication can be used.

4.6.2. Connecting to Multiple Web Application Servers (Enterprise+)

The WebtoB Enterprise edition provides the Reverse Proxy Group setting for a multi web application server configuration, and the Reverse Proxy setting can be used when connecting to a single server.

WebtoB Environment File Configuration

- **REVERSE_PROXY_GROUP** Section

Configure the following to create a Reverse Proxy Group.

```

*REVERSE_PROXY_GROUP
rproxyG1
    VhostName = "vhost1",
    PathPrefix = "/before/",
    RegExp = "\.(do|jsp)$",
    ServerPathPrefix = "/after/",
    RewriteHtmlUrl = "/after/ /before/",
    HtmlUrl = "a href",
    HtmlUrl = "img src",
    HtmlUrl = "link href",
    HtmlUrl = "script src",
    HtmlUrl = "frame src",
    HtmlUrl = "td background",
    HtmlUrl = "table background",
    HtmlUrl = "iframe src",
    HtmlUrl = "OBJECT CODEBASE",
    RewriteHtmlMaxSize = 4194304,
    WBRoutingCookieKey = "W2BRID"
    #SessionIDCookieKey = "JSESSIONID"

```

The previous setting changes the prefix of the requested service from "/before/" to "/after/".

Since each server provides the same service in a multi web application server configuration, the VhostName, PathPrefix, RegExp, and ServerPathPrefix items in the Reverse Proxy Group and Reverse Proxy must be set identically or set only in the Reverse Proxy Group.

To use a routingid separate from the server's Sticky Session routingid, use the WBRoutingCookieKey setting.

- **REVERSE_PROXY Section**

Configure the Reverse Proxy Group name, the target server's HTTP Listener information (Ip:Port), and each server's connection information.

```

*REVERSE_PROXY
rproxy1
    ServerAddress = "127.0.0.1:8088",
    ReverseProxyGroupName = "rproxyG1",
    MinPersistentServerConnections = 1,
    MaxPersistentServerConnections = 20,
    PersistentServerCheckTime = 50,
    PersistentServerTimeout = 300,
    PersistentServerCheckUrl = "/after/ping.html",
    MaxWebSocketConnections = 20
    #StickySessionRoutingID = "was1_servlet_engine",
    #ProxySSLFlag = Y,
    #ProxySSLName = pssl1

rproxy2
    ServerAddress = "127.0.0.1:8089",
    ReverseProxyGroupName = "rproxyG1",
    MinPersistentServerConnections = 1,
    MaxPersistentServerConnections = 20,
    PersistentServerCheckTime = 50,
    PersistentServerTimeout = 300,
    PersistentServerCheckUrl = "/after/ping.html",

```

```
MaxWebSocketConnections = 20
#StickySessionRoutingID = "was2_servlet_engine",
#ProxySSLFlag = Y,
#ProxySSLName = pssl1
```

In the previous settings, rproxy1(127.0.0.1:8088) and rproxy2(127.0.0.1:8089) are set in the rproxyG1, and requests are routed in RR(Round Robin) fashion. If there are multiple connections in a single Reverse Proxy, connections are assigned to each request in First-Assign (FA) mode.

MaxPersistentServerConnections specifies the maximum number of concurrent requests that can be processed. When the number of requests exceeds this limit, any additional requests are queued.

MinPersistentServerConnections specifies the minimum number of connections that must be maintained even if there are no requests. Connections in excess of this count are disconnected after the PersistentServerTimeout period.

PersistentServerCheckTime specifies the interval for sending a ping message to maintain the connection with the target server, and it must be set to a value less than the KeepAliveTimeout setting of the target server's HTTP Listener. Otherwise, the server will terminate the connection first. WebtoB maintains the connection by sending a ping at the specified interval and receiving a response from the server. This way, WebtoB can configure a connection pool of connections with other servers and dynamically maintain each connection.

A ping message is an HTTP HEAD request created with the PersistentServerCheckUrl setting. The ping message in the previous example is "HEAD /after/ping.html HTTP/1.1". Since the HEAD method only gets the HTTP Response Header, it minimizes the size of the response (pong) to a ping message. The target server must create the "/after/ping.html" application in order to respond to a ping.

When using the WebSocket for the connection, then number of WebSocket connections can be limited by using the MaxWebSocketConnections.

If the WBRoutingCookieKey is set for the Reverse Proxy Group, self-routing is used for rproxy1 and rproxy2. For a response that is processing through rproxy1, the WBRoutingCookieKey and rproxy1 are encoded in Base64 and added to the Set-Cookie header. Any subsequent requests from the same client are routed to rproxy1 by using the encoded value in the cookie. If WBRoutingCookieKey is not used, the StickySessionRoutingID is used with the Reverse Proxy Group's SessionIDCookieKey setting for Sticky Session routing. This requires that the target server add the engine name after the dot(.) character of the JSESSIONID value in the "Set-Cookie" Response Header.

ProxySSLFlag and ProxySSLName are specified for using SSL encryption communication between WebtoB and the target server.

• PROXY_SSL Section

Configure the SSL encryption communication setting between WebtoB and the target web application server.

```
*PROXY_SSL
pssl1
```

```
Verify = 2,  
VerifyDepth = 1,  
RequiredCiphers = "ALL:!ADH:!EXPORT56:RC4+RSA:!SHA:+HIGH:  
+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL",  
CACertificateFile="$ (WEBTOBDIR)/ssl/server.crt",  
CheckPeerValidPeriod = Y
```

Since WebtoB acts as the client and the target server as the server in this case, the target server must be configured with SSL encryption. If configured, WebtoB connects to the target server via TCP, and then requests for an SSL Handshake.

To verify the certificate during the SSL Handshake protocol, set the Verify item to 2 and set it to 0 otherwise. The CACertificateFile setting must be configured to verify the server certificate.

Checking the Configuration, Statistics, and Connection State

Use the wsadmin to check the configuration, statistics, and state of each connection.

1. Checking the Reverse Proxy Group settings

```
wsadmin> cfg -rpg  
REVERSE_PROXY_GROPU(0): Name = rproxyG1,  
VhostName = "vhost1",  
PathPrefix = "/before/",  
ServerPathPrefix = "/after/",  
RewriteHtmlMaxSize = 4194304,  
RewriteHtmlUrl = "/after/ /before/",  
HtmlUrl = "a href",  
HtmlUrl = "img src",  
HtmlUrl = "link href",  
HtmlUrl = "script src",  
HtmlUrl = "frame src",  
HtmlUrl = "td background",  
HtmlUrl = "table background",  
HtmlUrl = "iframe src",  
HtmlUrl = "OBJECT CODEBASE",  
SessionIdCookieKey = "JSESSIONID",  
FlexibleStickySessionRouting = N # (N:0, Y:1),  
ReverseProxyEntries = 2
```

2. Checking the Reverse Proxy settings

```
wsadmin> cfg -rp  
REVERSE_PROXY(0): Name = rproxy1,  
ReverseProxyGroupName = "rproxyG1",  
VhostName = "vhost1",  
PathPrefix = "/before/",  
ServerPathPrefix = "/after/",  
ServerAddress = "127.0.0.1:8088",  
HttpInBufSize = 16384,  
RewriteHtmlMaxSize = 4194304,  
RewriteHtmlUrl = "/after/ /before/",  
HtmlUrl = "a href",  
HtmlUrl = "img src",
```

```

    HtmlUrl = "link href",
    HtmlUrl = "script src",
    HtmlUrl = "frame src",
    HtmlUrl = "td background",
    HtmlUrl = "table background",
    HtmlUrl = "iframe src",
    HtmlUrl = "OBJECT CODEBASE",
    MinPersistentServerConnections = 1,
    MaxPersistentServerConnections = 20,
    PersistentServerCheckTime = 50,
    PersistentServerTimeout = 300,
    PersistentServerCheckUrl = "/after/ping.html",
    ProxySslFlag = N
REVERSE_PROXY(1): Name = rproxy2,
    ReverseProxyGroupName = "rproxyG1",
    VhostName = "vhost1",
    PathPrefix = "/before/",
    ServerPathPrefix = "/after/",
    ServerAddress = "127.0.0.1:8089",
    HttpInBufSize = 16384,
    RewriteHtmlMaxSize = 4194304,
    RewriteHtmlUrl = "/after/ /before/",
    HtmlUrl = "a href",
    HtmlUrl = "img src",
    HtmlUrl = "link href",
    HtmlUrl = "script src",
    HtmlUrl = "frame src",
    HtmlUrl = "td background",
    HtmlUrl = "table background",
    HtmlUrl = "iframe src",
    HtmlUrl = "OBJECT CODEBASE",
    MinPersistentServerConnections = 1,
    MaxPersistentServerConnections = 20,
    PersistentServerCheckTime = 50,
    PersistentServerTimeout = 300,
    PersistentServerCheckUrl = "/after/ping.html",
    ProxySslFlag = N

```

3. Checking the Reverse Proxy SSL settings

```

wsadmin> cfg -pssl
  PROXY_SSL(0): Name = pssl1,
    Verify= 2,
    VerifyDepth = 1,
    CheckPeerValidPeriod = Y          # (N:0, Y:1),
    CACertificatePath = "/home/tmax/server/webtob/ssl/",
    CACertificateFile = "/home/tmax/server/webtob/ssl/server.crt",
    Protocols = "TLSv1, TLSv1.1, TLSv1.2, TLSv1.3",
    RequiredCiphers = "ALL:!ADH:!EXPORT56:RC4+RSA:!SHA:+HIGH:
+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL"

```

4. Checking the statistics and connection information of the Reverse Proxy Group

```

wsadmin> st -rpg

```

```

-----
hth (rpgi)rpgname rproxyname count(qcnt) avg cons remote_ipaddr:port
-----
0 ( 0)rproxyG1 rproxy1 1( 0) 0.0025 1 127.0.0.1:8088
0 ( 0)rproxyG1 rproxy2 1( 0) 0.0021 1 127.0.0.1:8089
-----

```

The following describes each column of the output.

Column	Description
count	Total processed requests
qcnt	Number of queued requests for the Reverse Proxy Group (qcnt for rproxy1 and rproxy2 would be the same)
avg	Average processing time (sec)
cons	Number of connections to the target server

5. Checking the statistics and connection information of each Reverse Proxy

```

wsadmin> st -rp

HTH 0: RDY

-----
(rpi)rproxy_name rpgname status count idle spri clid ssl websocket
-----
( 0)rproxy1 rproxyG1 RDY 1 14 1 -1 N N
( 1)rproxy2 rproxyG1 RDY 1 10 2 -1 N N
-----

```

The following describes each column of the output.

Column	Description
status	Connection status information
count	Number of processed requests
idle	Idle time (sec)
spri	Internal connection index
clid	Client index of the currently processing request
ssl	Option to use SSL
websocket	Whether websocket connection is used

4.7. URLRewrite

Apache's mod_rewrite function has been ported to WebtoB. This function provides a rule-based URL rewriting function.

4.7.1. WebtoB Configuration

To use the URLRewrite function, the URLRewrite item in the NODE section must be set to "Y". In the URLRewriteConfig item, configuration related to Condition and Rule must be set.

The following is an example of the NODE section configuration needed to use the URLRewrite function.

```
*NODE
mynode
  URLRewrite = Y,
  URLRewriteConfig = "${WEBTOBDIR}/config/rewrite.conf"
```

4.7.2. URLRewriteConfig File Configuration

In URLRewriteConfig, RewriteCond and RewriteRule configurations can be used among the mod_rewrite functions of Apache. However, all configurations are not available, and some functions that will be later described are unavailable.



The following description of the URLRewriteConfig configuration was written in reference to "Apache 2.2 mod_rewrite". Some functions may not be supported in WebtoB.

4.7.3. RewriteCond

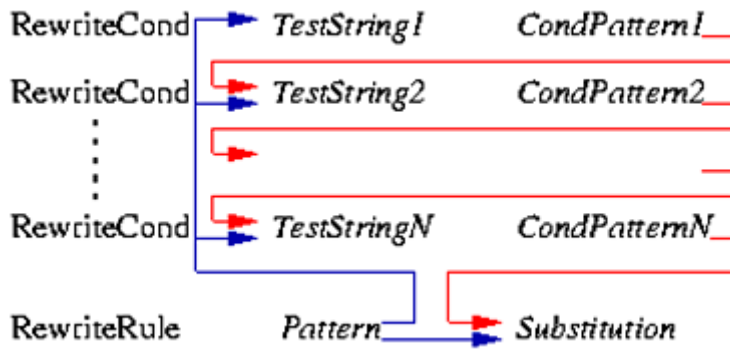
To configure RewriteCond, specify rewriting conditions as follows. Compare the TestString and Condpattern. If the conditions are met, RewriteCond is changed to the pattern defined in RewriteRule configuration.

```
RewriteCond <TestString> <CondPattern> [flags]
```

- <TestString>

In TestString, the following reserved words and general strings can be used.

- \$N (0 <= N <= 9)
 - Refer to the pattern wrapped in parentheses among RewriteRule's patterns.
- %N (1 <= N <= 9)
 - Refer to the pattern wrapped in parentheses among RewriteCond's CondPatterns.
 - \$N and %N have the following Regex Back-Reference structure.



Regex Back-Reference

- `%{SERVER_VARIABLE_NAME}`

Refer to the additional variables among the server environment variables used in CGI.

- `%{ENV:variable}` refers to the environment variable.
- `%{HTTP:header}` refers to HTTP Request Header.
- The following variables can also be used.

Type	Variable
HTTP headers	HTTP_USER_AGENT, HTTP_REFERER, HTTP_COOKIE, HTTP_FORWARDED, HTTP_HOST, HTTP_PROXY_CONNECTION, HTTP_ACCEPT
Connection and request	REMOTE_ADDR, REMOTE_PORT, REMOTE_USER, REMOTE_METHOD, QUERY_STRING, AUTH_TYPE
Server internals	DOCUMENT_ROOT, SERVER_NAME, SERVER_ADDR, SERVER_PORT, SERVER_PROTOCOL, SERVER_SOFTWARE
Date and time	TIME_YEAR, TIME_MON, TIME_DAY, TIME_HOUR, TIME_SEC, TIME_WDAY, TIME
Specials	THE_REQUEST, REQUEST_URI, HTTPS



`%{SSL:variable}`, `%{LA-U:variable}`, `%{LA-F:variable}`, REMOTE_HOST, REMOTE_IDENT, SCRIPT_FILENAME, SCRIPT_USER, SCRIPT_GROUP, SERVER_ADMIN, API_VERSION, REQUEST_FILENAME, IS_SUBREQ variables provided from Apache mod_rewrite cannot be used.

- `<CondPattern>`

CondPattern can use Perl compatible regular expressions. The following additional patterns can also be used.

Directive	Description
<code>!Pattern</code>	Specifies the cases that do not match the specified pattern(s).
<code><CondPattern</code>	Matches when TestString is before CondPattern alphabetically.

Directive	Description
>CondPattern	Matches when TestString is after CondPattern alphabetically.
=CondPattern	Matches when TestString is same as CondPattern.
-d	Matches when TestString is considered a path and it is a directory.
-f	Matches when TestString is considered a path and it is a file.
-s	Matches when TestString is considered a path and the file size is greater than 0.
-l	Matches when TestString is considered a path and it is a symbolic link.
-x	Matches when TestString is considered a path and it is an executable file.
-F	Matches when TestString is considered a path and it is accessible in server configuration.
-U	Matches when TestString is considered a URL and it is accessible in server configuration.



All of these patterns can be prefixed by an exclamation mark '!' to negate their meaning.

The following are descriptions for Regex vocabulary.

Directive	Description
.	Matches any single character.
+	Repeats the previous match one or more times.
*	Repeats the previous match zero or more times.
?	Makes the match optional.
^	Matches the beginning of the string.
\$	Matches the end of the string.
()	Groups several characters into a single unit and captures a match to use in a back reference.
[]	A character class - matches one of the characters.
[^]	A negative character class - matches any character not specified.

- [flags]

The pattern matching method can be modified by using [flag]s in the CondPattern.

Directive	Description
nocase NC	Not case-sensitive in pattern matching.

Directive	Description
ornext OR	If a different RewriteCond exists after RewriteCond, combine the next RewriteCond with the logical OR operator. If not explicitly specified, the next condition is combined with AND.
novary NV	If HTTP Header is used as TestString, do not add Vary Header to Response.

The following is an example of using the [OR] flag.

```
RewriteCond %{HTTP_HOST} ^host1.* [OR]
RewriteCond %{HTTP_HOST} ^host2.* [OR]
RewriteCond %{HTTP_HOST} ^host3.*
RewriteRule ...
```

If the [OR] flag is not used, each RewriteCond/RewriteRule must be written separately.

4.7.4. RewriteRule

RewriteRule configures the rewriting of user requests. If a user request is matched to a RewriteCond, the matched pattern in the user request is replaced by a substitution specified in the RewriteRule configuration.

```
RewriteRule <Pattern> <Substitution> [flags]
```

- <Pattern>
 - Patterns can use Perl compatible regular expressions. If '!' (not) is used, the cases where patterns do not match, can be specified.



If '!' is used, the group pattern (\$N) wrapped in parentheses cannot be used because the cases are for unmatched patterns do not have values for group patterns (\$N).

- If RewriteRule is used alone, pattern matching uses URL-path. If RewriteRule is used with RewriteCond, the final matched pattern is used.

- <Substitution>

Specifies what to substitute the matched URL with. The following values can be used.

Value	Description
file-system path	When specifying the absolute path of the file system that starts with '/', use the file in the user response. However, the configured path must exist in the file-system.
URL-path	When a general URL path is used, use the appropriate resource.

Value	Description
Absolute URL	If an absolute URL is specified (E.g., "http://<hostname>/file.html"), mod_rewrite checks to see whether the hostname matches the current host. If the hostname matches, the scheme and hostname are stripped out and the resulting path is treated as a URL path. Otherwise, an external redirect is performed for the given URL.
-	'-' means no substitution should be performed.
\$N (N=0..9)	Indicates the Nth group pattern among RewriteRule's patterns.
%N (N=1..9)	Indicates the Nth group pattern among the last matched RewriteCond patterns.
%{VARIABLE}	Refers to VARIABLE provided by the server. The items applied to TestString of RewriteCond can be used.
?	The Query string is not changed. In order to change the string, add '?' to the substitute string. In order to delete a query string, place a '?' at the end of the substitute string.



WebtoB does not support RewriteMap provided by mode_rewrite in Apache. `${mapname:key|default}` cannot be used.

- [flags]

- Special actions of RewriteRule can be configured with [flag]s.
- Multiple flags can be used by separating each flag with a comma (,). The following values can be used.

- B (escape backreferences)

The [B] flag turns the '%' escape encoding off. During substitution of the URL string, the '%' character is read as just a regular character and not as a special escape character.

For example, in `"/C%2b%2b"` the '%' character triggers an escape and the string is mapped to `"index.php?show=/C++"`. However, if the [B] option is used, the URL is mapped to `"index.php?show=/C%2b%2b"`.

```
RewriteRule ^(.*)$ index.php?show=$1
```

- chain|C (chained with next rule)

If this option is set, when a rule is not matched, the next rule will not be checked. If the next rule uses the [C] option, it is also skipped.

- cookie|CO=NAME:VAL:domain[:lifetime[:path[:secure[:httponly]]]] (set cookie)

Add the Set-Cookie Header to a response to add a Cookie to a user's browser.

- discardpathinfo|DPI (discard PATH_INFO)

In per-directory context, RewriteRule distinguishes URI and PATH_INFO and combines URI and PATH_INFO each time the rule is applied.

[DPI] options apply RewriteRule without distinguishing PATH_INFO.

- env | E=VAR:VAL (set environment variable)

Adds VAR=VAL to an environment variable. In VAL, regexp backreferences (\$N or %N) can be used. This environment variable can be used in SSI or CGI and can be used as %{ENV:VAR} among RewriteCond patterns.

- forbidden | F (force URL to be forbidden)

Sends a "403 Forbidden" response.

- gone | G (force URL to be gone)

Sends a "410 Gone" response.

- handle | H=Content-handler (force Content handler)

Configures a content-handler.

- last | L (last rule)

Signifies the end of the rewriting process. This option is similar to the break command in C.

- next | N (next round)

Executes the rewriting process again from the start to the changed URL. This option is similar to the continue command in C. Warning: infinite loops are possible.

- nocase | NC (no case)

Turns off case-sensitivity in patterns. E.g., 'A' and 'a' are considered the same.

- noescape | NE (no URI escaping of output)

Do not use %-encoding for URL in the rewriting process.

- nosubreq | NS (not for internal sub-requests)

Stops rewriting in the case of an internal request.

- passthrough | PT (pass through to next handler)

Uses the result of rewriting from another handler.

- qsappend | QSA (query string append)

When rewriting a query string, appends to the existing string instead of overwriting it.

- redirect | R[=code] (force redirect)

When the substitute string is an absolute URL, the URL is forcibly redirected even when the hostname matches a server's host.

If the code is not specified, 302 Moved Temporarily is used. In the code, the status code

can be entered directly or temp (default), permanent, and seeother can be specified.

- skip|S=num (skip next rules)

Skip the next 'num' rules. num is a numeric value.

- type|T=MIME-type (force MIME-Type)

Specifies the content-type of the response.

4.7.5. Examples

The following are examples of URL Rewrite.

- Example 1

The following is an example of the URLRewriteConfig configuration that matches the URL pattern "www.test.com/" and changes it to "www.test.com/rewrite.html".

```
# url rewrite config - ex1
RewriteCond %{HTTP_HOST} ^www\.test\.com$           # if {Host} == "www.test.com"
RewriteRule ^/$ /rewrite.html [L]                  # then "/" > "/rewrite.html"
```

- Example 2

The following is an example of a URLRewriteConfig configuration that changes a request for "www.test.com/temp/xxx.html" (and similar requests) to "www.test.com/temp/temp_error.html" when the file "xxx.html" does not exist in the temp directory.

```
# url rewrite config - ex2
RewriteCond %{REQUEST_FILENAME} !-f
# if {Requested file name} != file or the file pointer
RewriteRule ^/([^/]+)/$1/$1_error.html [L]
# then "/temp/xxx.html" > "/temp/temp_error.html"
```

- Example 3

The following is an example of a URLRewriteConfig configuration that changes to a request for "http://www.test.com:80" (and similar requests) to "https://www.test.com:443" (http to https).

```
# url rewrite config - ex3
RewriteCond %{HTTP_HOST} ^www\.test\.com$
# if {Host} == "www.test.com"
RewriteCond %{SERVER_PORT} 80
# AND {Port} == "80"
RewriteRule .* https://www.test.com:443$0 [R]
# then > "https://www.test.com:443$0" ($0: request uri)
```

- Example 4

The following is an example of a URLRewriteConfig configuration that matches the URL pattern "www.test.com/xxx.html" and changes it to "www.test.com/rewrite.html", ignoring Request URI.

```
# url rewrite config - ex4
RewriteCond %{HTTP_HOST} ^www\.test\.com$           # if {Host} == "www.test.com"
RewriteRule .* http://www.test_new.com [R]         # then > "http://www.test_new.com"
```

- Example 5

The following is an example of a URLRewriteConfig configuration that matches the URL pattern "www.test.com/test.html" and changes it to "www.test_new.com/test.html".

```
# url rewrite config - ex5
RewriteCond %{HTTP_HOST} ^www\.test\.com$
# if {Host} == "www.test.com"
RewriteRule .* http://www.test_new.com$0 [R]
# then > "http://www.test_new.com$0" ($0: request uri)
```

- Example 6

The following is an example of a URLRewriteConfig configuration that returns a "403 Forbidden" error when Request Method is POST and Referer Header does not exist.

```
# url rewrite config - ex6
RewriteCond %{REQUEST_METHOD} POST                 # if {Method} == "POST"
RewriteCond %{HTTP_REFERER} =""                   # AND {Referer} == ""
RewriteRule . - [F]                                # then > 403 Forbidden Return
```

- Example 7

The following is an example of a URLRewriteConfig configuration that processes "/403.html" when a request starts with "/../".

```
# url rewrite config - ex7
RewriteRule ^/\.\./ /403.html [L]                 # Start with "/../" > "/403.html"
```

- Example 8

The following is an example of a URLRewriteConfig configuration that processes "www.test.com/aaa/xxx.html" if the request is "aaa.test.com/xxx.html" and processes "www.test.com/bbb/xxx.html" if the request is "bbb.test.com/xxx.html".

```
# url rewrite config - ex8
RewriteCond %{HTTP_HOST} ^(aaa|bbb)\.test\.com
# if {Host} == ("aaa.test.com" OR "bbb.test.com")
RewriteRule .* /%1$0 [L]
# then "/xxx.html" > "/(aaa|bbb)/xxx.html" (%1: Opening parenthesis of RewriteCond, $0: request uri)
```


- Example 9

The following is an example of a URLRewriteConfig configuration that rewrites "www.test.com/test.do?query=value1" to "www.test.com:8080/test.do?query=value1", and "www.test.com/test.do?query=value2&.." to "www.test.com:8080/test.do?query=value2&..".

```
# url rewrite config - ex9
RewriteCond %{QUERY_STRING} ^query=value1$ [OR]
# if {QueryString} == "query=value1"
RewriteCond %{QUERY_STRING} ^query=value2&
# OR {QueryString} == "query=value2&.."
RewriteRule .* http://www.test.com:8080$0 [R]
# then > "http://www.test.com:8080$0" ($0: request uri)
```

- Example 10

The following is an example of a URLRewriteConfig configuration that processes "www.test.com/aaa/test.html" if the request is "aaa.test.com/test.html"(starts without 'www').

```
# url rewrite config - ex10
RewriteCond %{HTTP_HOST} !^www\.test\.com$
# if {Host} != www.test.com
RewriteCond %{HTTP_HOST} ^([a-zA-Z0-9]+)\.test.com$
# AND {Host} == "(alphanumeric characters).test.com"
RewriteRule .* /%1/$0 [L]
# then > "(alphanumeric string)/$0" (%1: Opening parenthesis of RewriteCond, $0: request uri)
```

- Example 11

The following is an example of a URLRewriteConfig configuration that processes 403 Forbidden if the request is for css or js files when Referer Header does not exist in Request Header.

```
# url rewrite config - ex11
RewriteCond %{HTTP_REFERER} !. # if {HTTP_REFERER} == ""
RewriteRule \.(css|js)$ - [F] # then "*.css|*.js" > 403 Forbidden Return
```

- Example 12 (REQUEST_URI)

The following is an example of a URLRewriteConfig configuration that matches the URL pattern "/user@somehost.com/" and changes it to "/req_test.jsp?blogId=user@somehost.com".

```
# url rewrite config - ex12 (REQUEST_URI)
RewriteCond %{REQUEST_URI} /([a-zA-Z0-9_-]+)@[a-zA-Z0-9_-]+(\.com|net|co.kr)?/?$
RewriteRule . /req_test.jsp?blogId=%1 [PT,L]
```

- Example 13 (HTTP_HOST)

The following is an example of a URLRewriteConfig configuration that redirects requests when the HTTP Request Header's host ends with "tmaxsoft.com" and the URL starts with "/redirect/" to "http://www.tmaxsoft.com/redirect.html".

```
# url rewrite config - ex13 (HTTP_HOST)
RewriteCond %{HTTP_HOST} tmaxsoft.com$
RewriteCond %{REQUEST_URI} /redirect/.*$
RewriteRule . http://www.tmaxsoft.com/redirect.html [R,L]
```

4.8. Filter Module Development (Enterprise+)

The `usrinc/httpfilt.h` file created during WebtoB installation provides the ISAPI (Internet Server Application Program Interface) Filter Interface. By including this file, a Filter module can be developed using the filter process provided by the WebtoB Enterprise edition.

The following is a sample filter source code that checks the client and request information.

```
/**
 * file: sample.c for sample_filter
 * Compilation example on Linux 32-bit
 * $ cc -m32 -D_REENTRANT -fPIC -I/webtob/usrinc -g -pthread -o sample.so sample.c -shared
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "httpfilt.h"

#define REDIRECT_PAGE "302 Moved Temporarily"

int wb_handler; /* WebtoB HTH count for only filters process */
time_t current_time;

int init()
{
    int n;

    /* set current time */
    time(&current_time);

    n = 0;
    /* n = init_check(); */

    return n;
}

void url_redirect(HTTP_FILTER_CONTEXT *pfc, char *url)
{
    pfc->ServerSupportFunction(pfc,
                              SF_REQ_SEND_RESPONSE_HEADER,
                              (LPVOID) REDIRECT_PAGE,
                              (LPDWORD) url,
                              0);

    return;
}

/**
```

```

* called when WebtoB is booted..
**/
BOOL WINAPI
GetFilterVersion(HTTP_FILTER_VERSION * pVer)
{
    DWORD dwRet;

    /* Version 4.0 */
    /* pVer->dwFilterVersion = MAKELONG(0, 4); */
    pVer->dwFilterVersion = HTTP_FILTER_REVISION;
    strncpy(pVer->lpszFilterDesc, "filter sample", SF_MAX_FILTER_DESC_LEN);

    /* init when webtoB is booting.. */
    wb_handler = 1; /* default */
#ifdef 0
    /* only filters process is used.. */
    if (0 < pVer->dwFlags) {
        wb_handler = pVer->dwFlags;
        pVer->dwFlags = 0;
        dwRet = init();
        if (dwRet < 0) {
            /* init failed.. */
            return 0;
        }
    }
#endif

    /* flags */
    pVer->dwFlags = (
        SF_NOTIFY_SECURE_PORT      |
        SF_NOTIFY_NONSECURE_PORT   |
        SF_NOTIFY_READ_RAW_DATA    |
        SF_NOTIFY_PREPROC_HEADERS  |
        SF_NOTIFY_URL_MAP          |
        SF_NOTIFY_AUTHENTICATION   |
        SF_NOTIFY_SEND_RAW_DATA    |
        SF_NOTIFY_LOG               |
        SF_NOTIFY_END_OF_NET_SESSION |
        SF_NOTIFY_ORDER_DEFAULT
    );

    return 1;
}

/**
 * run-time callback
 **/
DWORD WINAPI HttpFilterProc(HTTP_FILTER_CONTEXT *pfc, DWORD NotificationType, VOID * pvNotification)
{
    DWORD dwRet;

    /* init when the first request.. */
    dwRet = init();
    if (dwRet < 0) {
        /* init failed.. */
        char redirect_page[256];

        sprintf(redirect_page, "Location: %s\r\n", "error.html");
    }
}

```

```

url_redirect(pfc, redirect_page);

return SF_STATUS_REQ_FINISHED;
}

printf("\n\n start -> HttpFilterProc: NotificationType=%d, hth=%d \n",
        NotificationType, pfc->ulReserved);

/* Direct the notification to the appropriate routine for processing. */
switch (NotificationType) {
case SF_NOTIFY_PREPROC_HEADERS:
    dwRet = process_preproc_headers(pfc, (PHTTP_FILTER_PREPROC_HEADERS) pvNotification);
    break;
case SF_NOTIFY_SEND_RAW_DATA:
    dwRet = SF_STATUS_REQ_FINISHED_KEEP_CONN;
    break;
case SF_NOTIFY_READ_RAW_DATA:
case SF_NOTIFY_URL_MAP:
case SF_NOTIFY_AUTHENTICATION:
case SF_NOTIFY_LOG:
case SF_NOTIFY_END_OF_NET_SESSION:
default:
    dwRet = SF_STATUS_REQ_NEXT_NOTIFICATION;
    break;
}
return dwRet;
}

/**
 * process_preproc_headers
 * GetHeader()
 * - get the request header information
 * GetServerVariable()
 * - get the server variable information
 * AddResponseHeaders()
 * - add Response Header
 * SetHeader()
 * - add Request Header
 * ServerSupportFunction()
 * - set 302 Redirect page
 */
DWORD process_preproc_headers(HTTP_FILTER_CONTEXT *pfc, HTTP_FILTER_PREPROC_HEADERS *pvNotification)
{
    /* watch out buffer size */
    int size;
    char request_method[32];
    char request_uri[8192];
    char client_ip[256];
    char server_port[32];
    char content_type[256];
    char content_length[256];
    char body[10240];
    char s_header[8192];
    char cookie[8192];
    char add_res_header[8192];
    char redirect_page[8192];

    int redirect_test = 0;
    int setcookie_test = 1;

```

```

/* ssl y/n */
printf("Request is SSL? %d \n", pfc->fIsSecurePort);

/* get request method */
size = sizeof(request_method);
memset(request_method, 0, size);
/* pvNotification->GetHeader(pfc, "method", request_method, &size); */
pfc->GetServerVariable(pfc, "REQUEST_METHOD", request_method, &size);
printf("Request Method? %s \n", request_method);

/* get request url */
size = sizeof(request_uri);
memset(request_uri, 0, size);
/* pvNotification->GetHeader(pfc, "url", request_uri, &size); */
pfc->GetServerVariable(pfc, "REQUEST_URI", request_uri, &size);
printf("Request url? %s \n", request_uri);

/* get client ip */
size = (int) sizeof(client_ip);
memset(client_ip, 0, size);
pfc->GetServerVariable(pfc, "REMOTE_ADDR", client_ip, &size);
printf("Client ip? %s \n", client_ip);

/* get server port */
size = (int) sizeof(server_port);
memset(server_port, 0, size);
pfc->GetServerVariable(pfc, "SERVER_PORT", server_port, &size);
printf("Server Port? %s \n", server_port);

/* get Content-Type Request Header */
size = (int) sizeof(content_type);
memset(content_type, 0, size);
pvNotification->GetHeader(pfc, "Content-Type", content_type, &size);
printf("Content-Type? %s \n", content_type);

/* get Content-Length Request Header */
size = (int) sizeof(content_length);
memset(content_length, 0, size);
pvNotification->GetHeader(pfc, "Content-Length", content_length, &size);
printf("Content-Length? %s \n", content_length);

/* get Cookie Request Header */
size = (int) sizeof(cookie);
memset(cookie, 0, size);
pvNotification->GetHeader(pfc, "Cookie", cookie, &size);
printf("Cookie? %s \n", cookie);

/* get special header */
size = sizeof(s_header);
memset(s_header, 0, size);
pvNotification->GetHeader(pfc, "Special-Header", s_header, &size);
printf("Special-Header? %s \n", s_header);

/* get body data */
if (content_length) {
    size = (int) sizeof(body);
    memset(body, 0, size);
    pvNotification->GetHeader(pfc, "body", body, &size);
}

```

```

    printf("Body? %s \n", body);
}

/* add Response Header on processing request */
if (setcookie_test) {
    sprintf(add_res_header, "Set-Cookie: %s=%s; path=/; domain=test\r\n", "TestKey", "TestValue");
    pfc->AddResponseHeaders(pfc, add_res_header, 0);
}
sprintf(add_res_header, "AddResponseHeader: test");
pfc->AddResponseHeaders(pfc, add_res_header, 0);

/* add Request Header */
pvNotification->SetHeader("addReuestHeader", "test value");

/* 302 redirect */
if (redirect_test) {
    sprintf(redirect_page, "Location: %s\r\n", "test/redirect.html");
    url_redirect(pfc, redirect_page);

    return SF_STATUS_REQ_FINISHED;
}

return SF_STATUS_REQ_NEXT_NOTIFICATION;
}

```

5. Starting Up and Shutting Down

This chapter describes how to start up and shut down WebtoB after it has been configured.

5.1. Starting WebtoB

5.1.1. Before Starting WebtoB

Check the following items before starting WebtoB.

- Does a binary WebtoB environment file exist?
- Do WebtoB executable programs (wsm, htl, hth, htmls, cgis, ssis, etc.) in the WebtoBDir path of the NODE section exist?

After checking these items, WebtoB is ready to start.

5.1.2. wsboot

The wsboot program starts the WebtoB system.

- Usage

```
> wsboot [-f binary WebtoB environment file name][-T][-i][-configuration file name][-g server group name]
        [-A][-S server name][-s server name][-k number count][-w][-b]
        [-B boot in Block Listen mode][-o CLOPT string][-h]
```

- Options

- Options for starting WebtoB:

Option	Description
<code>[-f <i>binary WebtoB environment file name</i>]</code>	WebtoB operates according to its environment file. The <code>-f</code> option allows the user to specify the binary environment file. If the binary environment file is not specified the default file, <code>wsconfig</code> , is used.
<code>[-T]</code>	Runs WebtoB management processes (WSM, HTL, HTH). To provide external services, each server must be running.
<code>[-i]</code>	WebtoB is started after the successful execution of the command "wscfl -i http.m".
<code>[-c <i>configuration file name</i>]</code>	WebtoB is started after the successful execution of the command "wscfl -i configuration file name".

- Options applied after wsboot (used for starting processes):

Option	Description
<code>[-g <i>server group name</i>]</code>	Starts all server processes in the target server group. The server group must be defined in the SRVGROUP section of the environment file.
<code>[-A]</code>	Starts all server processes in the SERVER section of the WebtoB environment file. This option must be used with the <code>-T</code> option so that management processes are running.
<code>[-S <i>server name</i>]</code>	Starts the MinProc number of server processes in the target server.
<code>[-s <i>server name</i>]</code>	Starts only the processes in the target server. The server name must be defined in the SERVER section of the WebtoB environment file.
<code>[-k <i>number count</i>]</code>	Use this option with the <code>[-s]</code> option. The <code>[-k]</code> option is used to specify the number of server processes to be started. The sum of this number and number of active server processes cannot be greater than the MaxProc value specified in the SERVER section. If <code>[-k]</code> is not specified, only a single server process is started.
<code>[-w]</code>	Starts WebtoB server processes individually instead of collectively.
<code>[-b]</code>	Starts a backup server if any.
<code>[-B <i>booting in Block Listen mode</i>]</code>	Starts WebtoB in the blocked state where client requests are ignored.
<code>[-o CLOPT string]</code>	<code>[-o]</code> file name. Saves the standard output .
<code>[-h]</code>	Shows help information.

- Examples

The following is an example of starting all WebtoB management processes and service server processes specified in the wsconfig environment file.

```
$ wsboot
```

The following is an example of starting WebtoB management processes WSM, HTL, and HTH specified in the wsconfig2 environment file.

```
$ wsboot -T -f wsconfig2
```

The following is an example of starting all service server processes specified in the wsconfig environment file.

```
$ wsboot -A
```


The following is an example of starting MinProc number of svr1 processes specified in the environment file.

```
$ wsboot -S svr1
```

The following is an example of starting 5 svr1 processes specified in the wsconfig2 environment file.

```
$ wsboot -s svr1 -k 5 -f wsconfig2
```

When WebtoB system is started, the binary WebtoB environment file content is loaded onto shared memory and each nodes starts WebtoB feature processes (WSM, HTL, HTH) from the WEBTOBDIR directory specified in its environment file. Defined service server processes are then started in order.

The following is the start-up order of WebtoB processes.

```
WSM → HTL → HTH → Server processes
```

5.2. Shutting Down WebtoB

Similar to wsboot, the WebtoB system shutdown process applies the settings in the binary WebtoB environment file. Shared memory is eliminated and active WebtoB processes (WSM, HTL, HTH) and application server processes are terminated.

5.2.1. wsdown

The wsdown command shuts down the WebtoB system. The shutdown options are similar to the wsboot options.

- Usage

```
wsdown [-f binary WebtoB environment file name][-A]  
        [-S server name][-s server name][-k count]  
        [-g server group name][-p server number]  
        [-i][-w delay(within 5 seconds)][-k count][-h]
```

Options	Description
<i>[-f binary WebtoB environment file name]</i>	Binary WebtoB environment file name must be specified. If the file name is not specified, wsconfig is used by default.
<i>[-A]</i>	Shuts down all service server processes.
<i>[-S server name]</i>	Shuts down the currently operating target service server processes.

Options	Description
<code>[-s server name]</code>	Shuts down the target server processes. Shuts down the specified number of target server processes. (Similar to <code>wsboot</code>)
<code>[-k count]</code>	This option must be used with the [-s] option.
<code>[-g server group name]</code>	Terminates server processes in the target server group. The group name must be defined in the SVRGROUP section of the WebtoB environment file.
<code>[-p server number]</code>	Terminates target server process using process numbers. Unlike terminating with the [-s] option, this option can terminate specific processes by using process numbers (<code>spr_no</code>). Process numbers can be listed using the <code>st -p</code> command in <code>wsadmin</code> .
<code>[-i]</code>	Immediately executes the <code>wtdown</code> command. Normally, the <code>wtdown</code> command allows running processes to finish tasks before termination. However, the [-i] command immediately and forcibly terminates all processes.
<code>[-w delay(within 5 seconds)]</code>	Executes the <code>wtdown</code> command after the specified time. The specified time must be less than 5 seconds.
<code>[-h]</code>	Shows help information.

6. WebtoB Administration

WebtoB provides management functions such as status monitoring and system control. For example, system environment configuration can be accessed and modified dynamically without restarting the system. It also provides service-specific statistical data such as the total number of processed requests, average processing time, number of requests in the queue, etc.

WebtoB provides the aforementioned functions through the console management program `wsadmin` and the web program `WebAdmin`. This chapter describes the programs and commands provided for WebtoB administration.

6.1. `wsadmin` Console Management Program

`wsadmin` provides a text-based management environment. `wsadmin` waits in the command prompt and executes a command that has been entered.

`wsadmin` is executed as follows:

```
$ wsadmin
```

After `wsadmin` has been executed, the following message is displayed at the prompt to show that `wsadmin` is running.

```
--- Welcome to WebtoB Admin (Type "quit" to leave) ---  
$$1 webtob (wsadm) [2016-02-03T15:07:52]:
```

To display help for `wsadmin`, use the `'help<command>'` command.

```
$$2 webtob (wsadm) [2016-02-03T15:08:07]: help st  
Summary: stat, st: thread(process) and service state statistics  
Usage: stat (st) -v [servername] | -j [jsvservername] | -p [sprname] |  
      -s [servicename] | -rpg [rpgname] | -rp [rproxyname] |  
      -h [hthno] | -t | -T  
      -v [servername1,servername2,..] : status for each Server  
      -j [jsvservername1, jsvservername2,..] : status for each JSV Server(Jengine)  
      -p [sprname1,sprname2,..] : status for each Server Process  
      -s [servicename1,servicename2,..] : status for each Service  
      -rpg [rpgname1,rpgname2,..] : status for each ReverseProxyGroup  
      -rp [rproxyname1,rproxyname2,..] : status for each ReverseProxy  
      -h [hthno] : statistics of each HTH  
      -t : statistics of each HTH threads  
      -T : statistics of HTH threads group
```

To terminate `wsadmin`, use the `quit` (q) command.

The following are commands provided by wsadmin.

Command	Abbreviation	Description
cacherefresh	(cr)	Deletes responses stored in the HTTP response cache.
cachelist		Outputs the response information stored in the HTTP response cache.
cliinfo	(ci)	Checks the connected web browser.
clilisten	(cl)	Controls the Listen Port of the WebtoB client.
config	(cfg)	Displays the environment configuration.
discon	(ds)	Forcibly disconnects active clients.
history	(hist)	Displays the last 50 commands issued.
help	(h)	Displays help information.
hthmem		Records memory usage in a file.
ll		Changes the log level dynamically.
logend	(loge)	Stops logging.
logstart	(logs)	Starts logging.
logsearch		Scans the content of the log file.
patchinfo		Displays post-release patch information.
qpurge	(qp)	Deletes queued requests.
quit	(q)	Terminates wsadmin.
rebootsvr	(rbs)	Reboots the server program.
repeat	(r)	Repeats a command.
restat		Initializes the statistics of server processes.
resume	(rs)	Resumes a suspended server process.
set		Modifies the current environment value dynamically.
stat	(st)	Displays statistical data of processes, threads, and services.
suspend	(sp)	Suspends running server processes.
svrinfo	(si)	Displays server information.
webtobinfo	(wi)	Displays WebtoB system information.
wsboot	(boot)	Starts WebtoB. Same as wsboot .
wsdown	(down)	Terminates WebtoB. Same as wsdown .
!		Repeats the previous command.

6.1.1. Environment Configuration

6.1.1.1. webtobinfo (wi)

Displays WebtoB system environment settings including the system version, expiration date, and allowed maximum number of users.

- Usage

```
> wi
```

- Example

```
$$1 webtob (wsadm) [2016-02-03T15:12:27]: wi

License: CLOUD Enterprise edition
Version=WebtoB 5.0 SP 0 Fix #0 Linux-K2.6_x86 FD16384 B41 epoll 2016/02/03

    maxuser = UNLIMITED,
    node_count = 1,
    svgrpcount = 0,
    svr_count = 0, svc_alloc_count = 512

WebtoB All Node Info: node_count = 1:
-----
  no  name    nodeport  racport  shmkey  shmsize0  shmsize1  shmsize2  hth
-----
  0   webtob    7777     3333    196608   200360   1978068    1748    1
```

6.1.1.2. config (cfg)

Displays the currently running WebtoB's environment settings including the Domain, Node, Server Group, Server, and Service sections with their default values.

- Usage

```
> config [-d][-n][-vh VHOST name][-g SVRGROUP name][-v SERVER name]
        [-s SERVICE name][-dir DIRECTORY name][-u URI name][-a ALIAS name]
        [-l LOGGING name][-e EXT name][-ssl SSL name][-pssl PROXY_SSL name]
        [-tcpgw TCPGW name][-rproxy REVERSE_PROXY name]
        [-rpg REVERSE_PROXY_GROUP name][-ll LOGLEVEL name][-headers HEADERS name]
        [-access ACCESS name][-pc PRECEDING_COMMAND name]
        [-t HTH_THREAD name]
```

Option	Description
[-d]	DOMAIN section.
[-n]	NODE section.

Option	Description
<code>[-vh <i>VHOST name</i>]</code>	VHOST section or the settings of the specified VHOST.
<code>[-g <i>SVRGROUP name</i>]</code>	SVRGROUP section or the settings of the specified SVRGROUP.
<code>[-v <i>SERVER name</i>]</code>	SERVER section or the settings of the specified SERVER.
<code>[-s <i>SERVICE name</i>]</code>	SERVICE section or the settings of the specified SERVICE.
<code>[-dir <i>DIRECTORY name</i>]</code>	DIRECTORY section or the settings of the specified DIRECTORY.
<code>[-u <i>URI name</i>]</code>	URI section or the settings of the specified URI.
<code>[-a <i>ALIAS name</i>]</code>	ALIAS section or the settings of the specified ALIAS.
<code>[-l <i>LOGGING name</i>]</code>	LOGGING section or the settings of the specified LOGGING.
<code>[-e <i>EXT name</i>]</code>	EXT section or the settings of the specified EXT.
<code>[-ssl <i>SSL name</i>]</code>	SSL section or the settings of the specified SSL.
<code>[-pssl <i>PROXY_SSL name</i>]</code>	PROXY_SSL section or the settings of the specified PROXY_SSL.
<code>[-tcpgw <i>TCPGW name</i>]</code>	TCPGW section or the settings of the specified TCPGW.
<code>[-rproxy <i>REVERSE_PROXY name</i>]</code>	REVERSE_PROXY section or the settings of the specified REVERSE_PROXY.
<code>[-rpg <i>REVERSE_PROXY_GROUP name</i>]</code>	REVERSE_PROXY_GROUP section or the settings of the specified REVERSE_PROXY_GROUP.
<code>[-ll <i>LOGLEVEL name</i>]</code>	LOGLEVEL section or the settings of the specified LOGLEVEL.
<code>[-headers <i>HEADERS name</i>]</code>	HEADERS section or the settings of the specified HEADERS.
<code>[-access <i>ACCESS name</i>]</code>	ACCESS section or the settings of the specified ACCESS.
<code>[-pc <i>PRECEDING_COMMAND name</i>]</code>	PRECEDING_COMMAND section or the settings of the specified PRECEDING_COMMAND.
<code>[-t <i>HTH_THREAD name</i>]</code>	HTH_THREAD section or the settings of the specified HTH_THREAD.

- Example

The following is an example of using the `-n` option to display the NODE section settings. For more information on the NODE configuration items, refer to [Configuration Items](#).

```

$$$ webtob (wsadm) [2016-02-03T15:14:46]: cfg -n
  NODE(0): Name = webtob,
           HostName = "webtob",
           DocRoot = "/root/webtob_docroot/",
           SvrRoot = "/root/wb-5000-clean/",
           Method = "GET, POST, HEAD, OPTIONS",
           ShmKey = 54000,
           Hth = 1,
           HthQTimeout(hqt) = 0,
           NodePort = 7777,
           Port = "8080",
           JsvPort = 9999,
           ...
           Logging = "log1",

```

```
ErrorLog = "log2",
SysLog = "syslog",
...
CheckURL = Y,
CheckURLTo = "euc-kr",
CheckURLFrom = "utf-8",
SSIMaxDepth = 16
```

6.1.1.3. history (hist)

Displays the command history.

- Usage

```
> history
```

- Example

```
$$5 webtob (wsadm) [2016-02-03T15:14:47]: history
5: history
4: ci -s
3: ci
2: ci -s
1: ci
```

6.1.1.4. !

Re-executes the last executed command. Use the '!n' option to repeat a specific executed command where 'n' specifies the nth last executed command.

- Usage

```
> !
```

- Example

```
$$6 webtob (wsadm) [2016-02-03T15:14:49]: !4 ci -s
  Clients  Unique IPs  Dropped
  -----  -
HTH 0      0            0
All  0      0            0
```

6.1.2. Active State Information

6.1.2.1. cliinfo (ci)

Displays the environment settings, such as the current status, IP address, number of processed requests, of the currently connected client (usually a web browser).

- Usage

```
> ci [-s][-S][-vh Virtual Host name][-h HTH number]
```

Option	Description
[-s]	Displays basic statistics of all connected clients.
[-S]	Displays information about each connected client.
[-vh <i>Virtual Host name</i>]	Displays information about the clients connected to a specified Virtual Host.
[-h <i>HTH number</i>]	Displays information about the clients connected to a specified HTH.

- Example

- When used without options

The following is an example of using the ci command without any options. Each row represents a client.

```
$$1 webtob (wsadm) [2016-02-03T15:14:56]: ci
HTH  0:  RDY
-----
no  status count idle  local_ipaddr:port  remote_ipaddr:port  spri  user ssl
-----
 0  RDY    0    2  172.16.1.107:8080  172.16.1.100:1951  -1    N
 0  RUN    4    0  172.16.1.107:8080  172.16.1.202:60572  24    N
 1  QED    1    0  172.16.1.107:8080  172.16.1.202:60600  -1    N
 2  RUN    4    0  172.16.1.107:8080  172.16.1.202:60575  26    N
 3  RUN    0    0  172.16.1.107:8080  172.16.1.202:60601  28    N
 4  RUN    4    0  172.16.1.107:8080  172.16.1.202:60577  22    N
 5  RUN    0    0  172.16.1.107:8080  172.16.1.202:60594  23    N
 6  QED    1    0  172.16.1.107:8080  172.16.1.202:60598  -1    N
 7  RUN    0    0  172.16.1.107:8080  172.16.1.202:60596  25    N
 8  QED    1    0  172.16.1.107:8080  172.16.1.202:60597  -1    N
 9  QED    2    0  172.16.1.107:8080  172.16.1.202:60595  -1    N
10  RUN    4    0  172.16.1.107:8080  172.16.1.202:60584  21    N
11  QED    2    0  172.16.1.107:8080  172.16.1.202:60592  -1    N
12  RUN    3    0  172.16.1.107:8080  172.16.1.202:60587  29    N
13  RUN    4    0  172.16.1.107:8080  172.16.1.202:60588  30    N
14  QED    1    0  172.16.1.107:8080  172.16.1.202:60599  -1    N
15  RUN    1    0  172.16.1.107:8080  172.16.1.202:60593  27    N
-----
HTH  RDY  QED  RUN  ETC total
 0    0    6   10   0    16
-----
```


HTH	RDY	QED	RUN	ETC	total
0	0	6	10	0	16
Total	0	6	10	0	16

The following describes each output item.

Item	Description
status	<p>Client state within the server.</p> <ul style="list-style-type: none"> ◦ RDY: Receiving a request from the client. ◦ RUN: A client request is being processed on the server. ◦ QED: Received a client request and identified the server to forward the request to. However, the request is waiting in the server queue because the server is busy processing other requests.
count	Number of requests sent by the client.
idle	Client idle time with no data exchanges.
local_ipaddr:port, remote_ipaddr:port	Server and client IP:PORT info.
spri	Sever and client IP address.
user	<p>Client type when the client is not an HTTP client.</p> <ul style="list-style-type: none"> ◦ tcpgw-c: TCP gateway client. ◦ tpcgw-s: TCP gateway server. ◦ conn-c: Client that is processing requests via CONNECT method. ◦ conn-s: Server that is processing requests via CONNECT method. ◦ rproxy-s: Server connected to Reverse Proxy. ◦ rproxy-ws-c: Client that is using WebSocket on the Reverse Proxy server. ◦ rproxy-ws-s: Server that is using WebSocket on the Reverse Proxy server. ◦ internal-c: Client that is using internal redirect, etc.
ssl	Specifies whether the client is using an SSL connection.

- Using the -s option

The -s option displays the total number of clients, unique IP addresses, the number of clients that have been disconnected without being registered in HTH.

```

$$1 webtob (wsadm) [2016-02-03T15:15:23]: ci -s
      Clients Unique IPs Dropped
      -----
HTH 0      16           0         0
  All      16           0         0

```

6.1.2.2. svrinfo (si)

Displays active servers information.

- Usage

```
> si [server name, server name,..]
```

Option	Description
[server name, server name,..]	Displays information of all servers or the specified servers.

- Example

The following is the command output.

```

$$1 webtob (wsadm) [2016-02-03T15:17:20]: si
-----
 hth  svrname (svri)  status  reqs  count  cqcnt  aqcnt  qpcnt  emcnt  rsent  rbent
-----
  0  MyGroup  ( 1)  RDY    12    12    0      0      0      0      0      0
  0  cgi      ( 2)  RDY     0     0    0      0      0      0      0      0
  0  php      ( 3)  RDY   352   352    0      0      0      0      0      0

```

The following describes each output item.

Item	Description
svrname	Server name specified in the SERVER section.
(svri)	Internal server index.
status	Client state within the server. <ul style="list-style-type: none"> • RDY: Request can be processed. There are server processes connected to WebtoB. • NRDY: Request cannot be processed. There are no Server processes connected to WebtoB. • BLK: Server is suspended by administrator command. Request cannot be processed.
reqs	Number of requests sent to the server.

Item	Description
count	Number of processed requests.
cqcnt	Number of requests in the queue.
aqcnt	Cumulative number of queued requests. (Cumulative cqcnt value)
qpcnt	Number of queued requests that have been removed from the queue due to request timeout, qp command, etc.
emcnt	Number of times the cqcnt has exceeded the MaxQCount.
rsCnt	Number of server restarts due to abnormal termination.
rbcnt	Number of reboots due to the server execution of the rbs command.

6.1.2.3. stat (st)

Displays statistics on the running processes and services of the currently running server.

Dynamic information including the current server process state, names of running services, number of processed services, service states, number of queued requests, etc. are displayed.

- Usage

```
> st [-v server name,server name,...][-j JSV server name,JSV server name,...]
    [-p server process name,server process name,...]
    [-rpg Reverse Proxy Group name,Reverse Proxy Group name,...]
    [-rproxy Reverse Proxy name,Reverse Proxy name,...]
    [-tcpgw TCPGW name,TCPGW name,...]
    [-s service name,service name,...][-h HTH number]
    [-T][-t]
```

Option	Description
<code>[-v server name, server name,..]</code>	Displays states of servers. Same as svrinfo .
<code>[-j JSV server name, JSV server name,..]</code>	Displays statistical information of JSV servers.
<code>[-p server process name, server process name,..]</code>	Displays states of individual server processes.
<code>[-rpg Reverse Proxy Group name, Reverse Proxy Group name,..]</code>	Displays Reverse Proxy Group stats.
<code>[-rproxy Reverse Proxy name, Reverse Proxy name,..]</code>	Displays states of each Reverse Proxy connections.
<code>[-tcpgw TCPGW name, TCPGW name,..]</code>	Displays states of each TCPGW connections.

Option	Description
<code>[-s service name, service name,..]</code>	Displays service states.
<code>[-h HTH number]</code>	Displays internal stats of HTH processes.
<code>[-T]</code>	Displays internal stats on each thread type within HTH processes.
<code>[-t]</code>	Displays internal stats on each thread within HTH processes.

- Example

- Using the `-p` option

The following is an example of using the `-p` option to display server process information.

```

$$1 webtob (wsadm) [2016-02-03T15:43:22]: st -p

HTH 0(23786): RDY
-----
-
  svr_name  svgname    spr_no(pid)  status    reqs    count    avg(rt)  clid  svc  v
  contime
-----
-
  php       phpg       220( 23789)  RDY       0       0       0.0000( 0)  -1   - 0
  14187
  ...
  MyGroup1  jsvg       120( 0)     RDY       0       0       0.0000( 0)  -1   - 1
  14160
           0 jengineid(ZG9tYWLuMS9hZG1pb1N1cnZ1cg==)(domain1/adminServer)
  MyGroup1  jsvg       121( 1)     RDY       0       0       0.0000( 0)  -1   - 1
  14153
           0 jengineid(ZG9tYWLuMS9hZG1pb1N1cnZ1cg==)(domain1/adminServer)
  ...

```

The following describes each output item.

Item	Description
<code>svr_name</code>	Server name specified in the SERVER section.
<code>svgname</code>	Server group name specified in the SVRGROUP section.
<code>spr_no</code>	Internally assigned number by WebtoB.
<code>pid</code>	Process ID. (For JSV server, this refers to the ID received from the worker thread of each connection)

Item	Description
status	Current status. <ul style="list-style-type: none"> ◦ RDY: Requests can be processed. There are server processes connected to WebtoB. ◦ NRDY: Requests cannot be processed. There are no server processes connected to WebtoB. ◦ RUN: The process is currently processing a request. ◦ BRUN: The process, while in the process of sending a response, is temporarily awaiting for the flow control.
reqs	Number of requests sent to the server.
count	Number of processed requests.
avg	Average processing time (sec).
(rt)	Time elapsed while processing the current request.
clid	Client ID of the current request.
svc	Service name of the EXT or URI section of the request.
v	WJP (WebtoB-JEUS Protocol) version information if connected to JEUS. For internal server process, set it to 0.
contime	Time elapsed since connecting to HTH.

- Printing stats at regular intervals

A command can be executed at regular intervals for monitoring and debugging purposes.

The following is an example of executing 'st -s' 30 times in a 5-second interval.

```
$$10 webtob (wsadm(wsmon)) [2009/10/22:12:37:56]: r -i 5 -k 30 st -s
```

The following is an example of executing 'st -p' for 30 sec in a 5-second interval.

```
$$10 webtob (wsadm(wsmon)) [2009/10/22:12:37:56]: r -i 5 -f 30 st -p
```

- Using the -tcpgw option

The following is an example of using the -tcpgw option to output TCPGW information.

```
$$4 webtob (wsadm) [2018-07-19T12:08:08]: st -tcpgw
```

```
-----
hth (tcpgwi)tcpgwname   count  avg   cons  remote_ipaddr:port
-----
0 ( 0/ 0)tcpgw1         2    15.7293  0    127.0.0.1:8088
0 ( 1/ 0)tcpgw2         0     0.0000  0    192.168.0.1:18088
0 ( 1/ 1)tcpgw2         0     0.0000  0    192.168.0.1:28088
-----
```

```

0 ( 1/ 2)tcpgw2      0  0.0000  0  192.168.0.1:38088
0 ( 1/ 3)tcpgw2      0  0.0000  0  192.168.0.1:48088
0 ( 1/ 4)tcpgw2      0  0.0000  0  192.168.0.1:58088
...

```

The following describes each output item.

Item	Description
hth	hth number.
tcpgwname	TCPGW name.
count	Number of processed TCP connections.
avg	Average TCP connection duration.
cons	Number of active connections.
remote_ipaddr:port	Server IP address and port number.

- Using the -T option

The following is an example of using the -T option to display the thread stats.

```

$$$1 webtob (wsadm) [2016-02-03T15:45:17]: st -T

HTH 0: RDY
-----
no  thread_type  status  threads  atasks  ptasks  qtasks
-----
0  ACCESSLOG     RDY     1        18      18      0
1  WORKER        RDY     12       401     401     0
2  SENDFILE      RDY     4         6       6       0
...

```

The following describes each output item.

Item	Description
thread_type	Thread type.
status	Current status. <ul style="list-style-type: none"> ◦ RDY: Waits for a new task to process. ◦ NRDY: Cannot process any tasks.
threads	Thread count for the thread_type.
atasks	Total number of tasks.
ptasks	Total number of processed tasks.
qtasks	Number of currently queued tasks.

- Using the -t option

The following is an example of using the `-t` option to display the thread stats.

```

$$1 webtob (wsadm) [2016-02-03T15:44:57]: st -t

HTH 0: RDY
-----
no  thread_id      status  elapsed   atasks   ptasks   qtasks   task_type
-----
0   ACCESSLOG      RDY     4         18       18       0        NONE
1   WORKER001      RUN     0         34       33       1        SSLWRITE
2   WORKER002      RDY    31         34       34       0        NONE
3   WORKER003      RDY    31         34       34       0        NONE
4   WORKER004      RDY    31         34       34       0        NONE
5   WORKER005      RDY    31         34       34       0        NONE
6   WORKER006      RDY    31         33       33       0        NONE
7   WORKER007      RDY    31         33       33       0        NONE
8   WORKER008      RDY    31         33       33       0        NONE
9   WORKER009      RDY    31         33       33       0        NONE
10  WORKER010      RDY    31         33       33       0        NONE
11  WORKER011      RDY    31         33       33       0        NONE
12  WORKER012      RDY    31         33       33       0        NONE
13  SENDFILE001    RDY     4          2         2         0        NONE
14  SENDFILE002    RDY     4          1         1         0        NONE
15  SENDFILE003    RDY     4          1         1         0        NONE
16  SENDFILE004    RDY     4          2         2         0        NONE
...

```

The following describes each output item.

Item	Description
thread_id	Thread ID.
status	Current status. <ul style="list-style-type: none"> ◦ RDY: Waits for a new task to process. ◦ NRDY: Cannot process any tasks. ◦ RUN: Currently processing a task.
elapsed	Time elapsed since the start or end of processing a task.
atasks	Total number of tasks.
ptasks	Total number of processed tasks.
qtasks	Number of currently queued tasks.
task_type	Task currently being processed. When no tasks are being processed, 'NONE' is printed. If no task has been started while the thread was woken up, the status is 'RUN', and the task type is 'NONE'.

6.1.3. Suspend and Resume Server Processes

6.1.3.1. suspend (sp)

Suspends currently running server process's activities. When a server process is suspended, it finishes its current task and then enters an idle state. Services in queue are left untouched and new service requests are pooled into the queue.

If service is interrupted due to a server application error, the server process must be suspended. Suspending a server process is useful when it is impossible to process a request due to an unknown error and additional work is required.

- Usage

```
> suspend [-v server name]
```

Option	Description
<code>[-v server name]</code>	Suspends the specified server.

- Example

The following is an example of suspending a PHP server. Check the server status with `st -v` to check that the status has been changed to BLK.

```
$$1 webtob (wsadm) [2016-02-03T15:56:57]: suspend -v php
Server/php is supended
$$2 webtob (wsadm) [2016-02-03T15:57:07]: st -v

-----
 hth  svrname (svri)  status    reqs    count cqcnt    aqcnt qpcent emcent rscent rbcent
-----
  ...
  0  php           ( 3)  BLK      539     539     0     403     0     0     0     0
  ...
```

6.1.3.2. resume (rs)

Resumes a server process suspended by [suspend](#). A resumed server process starts processing services in the queue and goes into the RDY state.

- Usage

```
> resume [-v server name]
```

Option	Description
<code>[-v server name]</code>	Resumes the specified server ready to processing requests.

- Example

The following is an example of resuming a suspended PHP server. Use the command `st -v` to check that the status has been changed to RDY.

```
$$3 webtob (wsadm) [2016-02-03T15:57:27]: st -v resume -v php
Server/php is resumed
$$4 webtob (wsadm) [2016-02-03T15:57:37]: st -v
-----
 hth  svrname (svri)  status  reqs  count cqcnt  aqcnt qpcent  ement rscent rbcent
-----
  ...
  0  php          ( 3)  RDY    539    539    0     403    0     0     0     0
  ...
```

6.1.4. Queue Purge

6.1.4.1. qpurge (qp)

A large number of requests can fill server queues, cause delays, and create difficulties with servicing queued requests. The queue purge command can clear these queues to keep the system robust. This feature is useful for banks or public offices where hundreds of thousands of requests are processed each day. Deleted applications can be processed again at the client's request. WebtoB manages separate queues for each server process. The administrator tool can clear queues individually to remove bottlenecks in the system. Per-application processing is possible, helping other applications to process effectively.

A service that is deleted by qpurge sends an error as follows:

```
503 Service Temporarily Unavailable.
```

- Usage

```
> qpurge [-v server name]
```

Option	Description
<code>[-v server name]</code>	Deletes the requests in the specified server queue.

- Example

The following is an example of deleting requests in a PHP server queue. `Purged_count` displays number of requests deleted.

```
$$11 webtob (wsadm) [2016-02-03T15:58:52]: qpurge -v php
q for svr php is purged: purged_count = 6
```

6.1.5. Dynamic Configuration Change

6.1.5.1. set

This command can dynamically modify values in the configuration file. Not every variable can be modified dynamically so check the list of available variables using the `cfg` command in wsadmin.

- Usage

```
> set [-n NODE name][-vh VHOST name][-g SVRGROUP name][-v SERVER name][-s SERVICE name]
```

Option	Description
<code>[-n NODE name]</code>	Configures the NODE section.
<code>[-vh VHOST name]</code>	Configures the specified VHOST section.
<code>[-g SVRGROUP name]</code>	Configures the specified server group.
<code>[-v SERVER name]</code>	Configures the specified server.
<code>[-s SERVICE name]</code>	Configures the specified service.

- Example

The following is an example of checking the MaxQCount value by using `cfg -v php` and changing its value by using the `set` command. After executing the `set` command, the MaxQCount value is changed to 100.

```
$$10 webtob (wsadm) [2016-02-03T16:01:32]: cfg -v php
SERVER(3): Name = php,
         SvgName = phpg,
         MinProc = 10,
         MaxProc = 10,
         MaxQCount(mq) = 0,
         ...
$$11 webtob (wsadm) [2016-02-03T16:01:42]: set -v php mq 100
new value (100) is set for section = SERVER, name = php, fld = mq
$$12 webtob (wsadm) [2016-02-03T16:01:52]: cfg -v php
SERVER(3): Name = php,
         SvgName = phpg,
         MinProc = 10,
         MaxProc = 10,
         MaxQCount(mq) = 100,
         ...
```

6.1.6. Client Disconnection

6.1.6.1. discon (ds)

A client connection that is currently connected or idle can be forcibly disconnected. First use the `ci` command to obtain client information and then use the `discon` command.

- Usage

```
> discon [-h HTH number] -a | -i idle time | -c Client ID [-f]
```

Option	Description
<code>[-h HTH number]</code>	Disconnects clients that are connected to a specified HTH.
<code>-a</code>	Disconnects all clients that are connected to HTHs.
<code>-i idle time</code>	Disconnects clients that exceed the specified amount of time (unit: second).
<code>-c Client ID</code>	Disconnects specified clients (client ID displayed by <code>ci</code>).
<code>[-f]</code>	Disconnects a connection immediately.

- Example

The following is an example of checking a client ID by using the `ci` command and then disconnecting the client 15 forcibly by using `discon` command.

```
$$16 webtob (wsadm) [2016-02-03T16:02:42]: ci
HTH 0: RDY
-----
no  status count idle  local_ipaddr:port  remote_ipaddr:port  spri  user
-----
 15  RDY     0    2   172.16.1.107:8080   172.16.1.202:36505  -1
...

$$17 webtob (wsadm) [2016-02-03T16:02:55]: discon -c 15
client (hth0: 15) is disconnected
```

6.1.7. Others

6.1.7.1. cachelist

Displays the response data stored in the current WebtoB HTTP response cache. Due to the large data, the results are stored in a separate file.

- Usage

```
> cachelist
```

- Example

The following is an example of the response to the request `"/index.html"` stored in cache. The rest

of the request path is used only for internal server debugging.

```
$$$1 webtob (wsadm) [2016-02-03T16:02:15]: cacheList
Cache contents files are created in /root/wb-5000-clean/log/cachelist/. Please check the
directory.

Sample output file:
0 webtob:8080/index.html 5 0 2016/02/03:14:59:45 10 0 327043 5 2016/02/03:15:48:56 (1253083736)
Total cached response=1
Total content length=5
```

6.1.7.2. cacherefresh (cr)

Deletes the response data from the WebtoB HTTP response cache.

- Usage

```
> cacherefresh {-a | -h | -i | -j | -r | -u URL}
```

Option	Description
[-a]	Deletes all cached responses.
[-h]	Only deletes cached HTML and similar texts when the SVRTYPE is HTML.
[-i]	Only deletes cached images when the SVRTYPE is HTML.
[-j]	Only deletes cached responses when the SVRTYPE is JSV.
[-r]	Only deletes cached responses when the SVRTYPE is Reverse Proxy.
[-u URL]	Only deletes responses that match the specified URL using the fnmatch method.

- Example

The following is an example of deleting all responses from the cache.

```
$$$1 webtob (wsadm) [2016-02-03T16:04:57]: cacherefresh -a
```

The following is an example of only deleting responses of "test.domain.com/test.html".

```
$$$2 webtob (wsadm) [2016-02-03T16:05:07]: cacherefresh -u test.domain.com/test.html
```

6.1.7.3. clilisten (cl)

Removes or creates the listen ports that are used by WebtoB to receive client connections.

- Usage

```
> clilisten {on | off}
```

Option	Description
on	Creates listen ports.
off	Removes listen ports. Once removed, a new client cannot access WebtoB. Clients that are already connected to HTH will be disconnected after processing current requests.

- Example

The following is an example of removing listen ports.

```
$$18 webtob (wsadm) [2016-02-03T16:05:17]: clilisten off  
client listen blocked
```

The following is the result of checking listen ports from a Linux console. It is assumed the listen port is 8080.

```
[root@webtob ~]# netstat -ant  
Active Internet connections (servers and established)  
Proto Recv-Q Send-Q Local Address           Foreign Address         State  
tcp      0      0 0.0.0.0:9090            0.0.0.0:*               LISTEN  
tcp      0      0 0.0.0.0:681             0.0.0.0:*               LISTEN  
tcp      0      0 0.0.0.0:111             0.0.0.0:*               LISTEN  
tcp      0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN  
tcp      0      0 127.0.0.1:631           0.0.0.0:*               LISTEN  
...
```

The following is the result after executing `clilisten off`. It shows that listen port 8080 has been removed.

```
[root@webtob ~]# netstat -ant  
Active Internet connections (servers and established)  
Proto Recv-Q Send-Q Local Address           Foreign Address         State  
tcp      0      0 0.0.0.0:9090            0.0.0.0:*               LISTEN  
tcp      0      0 0.0.0.0:681             0.0.0.0:*               LISTEN  
tcp      0      0 0.0.0.0:111             0.0.0.0:*               LISTEN  
tcp      0      0 127.0.0.1:631           0.0.0.0:*               LISTEN  
...
```

6.1.7.4. logstart (logs), logend (loge)

Records executed commands and outputs to log files. A log file is created in the current directory.

From the log data, peak service time, unnecessary server process, and queueing status can be checked to analyze the overall system. The `logstart` command starts logging and the `logend`

command stops logging.

- Usage

```
> logstart (logs)
```

```
> logend (loge)
```

- Example

The result of the `st -p` command is saved in the following `log.txt` file.

```
$$1 webtob (wsadm) [2016-02-03T16:05:55]: logstart log.txt
logging start ok
$$2 webtob (wsadm) [2016-02-03T16:05:59]: st -p
...
$$3 webtob (wsadm) [2016-02-03T16:06:05]: logend
logging end ok
$$4 webtob (wsadm) [2016-02-03T16:06:25]:
```

6.1.7.5. logsearch

Scans the content of a specified log file.

- Usage

```
> logsearch [keyword] [logfile]
```

Option	Description
[keyword]	String to look for in a log file. Case-insensitive.
[logfile]	Log file in which the string is looked for. Both absolute and relative paths are supported. <ul style="list-style-type: none">• When using a relative path, the directory containing <code>syslog</code> is the reference location.• When using an absolute path, caution is needed because it grants access to all files with relevant permissions.

- Example

The following example searches the content of `syslog`.

```
$$1 webtob (wsadm) [2020-08-05T14:58:35]: logsearch successfully system.log_08052020
[2020-08-05T10:44:16] [HTH(22378)] [I] [HTH-00074] Successfully connected and registered to HTL.
HTH is ready to receive client connections.
```

```
[2020-08-05T10:44:16] [HTL(22377)] [I] [HTL-00021] Successfully registered HTH. Start listening
for this HTH. index=0, fd=6
[2020-08-05T14:57:29] [HTH(4901)] [I] [HTH-00074] Successfully connected and registered to HTL.
HTH is ready to receive client connections.
[2020-08-05T14:57:29] [HTL(4900)] [I] [HTL-00021] Successfully registered HTH. Start listening
for this HTH. index=0, fd=6
```

6.1.7.6. II

The LOGLEVEL configuration can be changed dynamically. Refer to [Configuration Items](#) for more information on logger names, levels, and options.

- Usage

```
> ll [Logger name][-l level | -o option | -rs seconds | -rf fileSize]
```

Option	Description
[Logger name]	Configures the logger supported by WebtoB.
[-l level]	Changes the log level to the specified level.
[-o option]	Changes the logger option to the specified option.
[-rs seconds]	Changes the logger's rotateBySeconds setting to seconds.
[-rf fileSize]	Changes the logger's rotateByFileSize setting to fileSize.

- Example

The following command changes logger ".hth" level to DEBUG. The changed value can be checked by using `cfg -ll`.

```
$$6 webtob (wsadm) [2016-02-03T16:06:35]: ll .hth -l DEBUG
Log level is successfully updated. logger=.hth, level=DEBUG, options=
```

6.1.7.7. patchinfo

Displays post release patch information.

- Usage

```
> patchinfo
```

- Example

The following is an example of executing `patchinfo` from a released WebtoB version.

```
$$1 webtob (wsadm) [2016-02-03T16:07:35]: patchinfo
```

```
-----  
Released (2015/01/29) : 5.0.0.0-B41.0.0  
-----
```

6.1.7.8. rebootsvr (rbs)

Replaces an active server process with a new process.

Set **WEBTOB_BKAPPPDIR** to the environment variable, copy new program files to a temporary directory, and then replace the server process using the following command.

- Usage

```
> rebootsvr new_file old_file
```

Option	Description
<i>new_file</i>	New file (object file).
<i>old_file</i>	Current file (object file).

6.1.7.9. repeat (r)

Monitors the current information at a specified interval.

- Usage

```
> repeat [-i second] [-k count| -f second] command
```

Option	Description
<i>[-i second]</i>	Interval (unit: second) to repeat the command.
<i>[-k count]</i>	Repeats a command the specified number of times.
<i>[-f second]</i>	Repeats for the specified time (unit: second).
<i>command</i>	Command to repeat.

- Example

The following is an example of repeating the command **'st -s'** 30 times every 5 seconds.

```
$$$ webtob (wsadm) [2016-02-03T16:08:02]: r -i 5 -k 30 st -p
```


6.1.7.10. restat

Initializes statistics of server processes. Use this command to reset a specific service or server process statistical information.

- Usage

```
> restat [-a | -v server name]
```

Option	Description
[-a]	Initializes statistics of all servers processes.
[-v <i>server name</i>]	Initializes statistics of a specified server's processes.

- Example

The following is an example of initializing statistics of processes on a server named "MyGroup".

```
$$1 webtob (wsadm) [2020-08-11T17:25:07]: restat -v MyGroup
```

6.2. wsmon Console Monitoring Program

wsmon provides a text-based monitoring environment where it executes commands entered at the prompt.

Start the wsmon program as follows:

```
$ wsmon
```

The following prompt displays when wsmon starts.

```
--- Welcome to WebtoB Mon (Type "quit" to leave) ---  
$$1 webtob (wsmon) [2016-02-03T16:08:22]:
```

Use the **'help <command name>'** to display help information about a specific command.

```
$$$5 webtob (wsmon) [2016-02-03T16:08:29]: help ci  
Summary: cliinfo, ci: show client properties  
Usage: cliinfo (ci) [-s | -S] [-vh vhostname] [-h hthno]  
    -s : summury of the client info  
    -S : detail summury of the client info  
    -vh vhostname : specify VHOST  
    -h hthno : specify a HTH number to view
```

Use the quit (q) command to terminate wsmon.

```
$$1 webtob (wsmon) [2016-02-03T16:08:42]: quit
```

The following are the wsmon commands. Refer to each command section for details.

Command	Abbreviation	Description
cachelist		Prints the responses in HTTP response cache.
cliinfo	(ci)	Prints the web browser information.
config	(cfg)	Prints environment configuration settings.
history	(history)	Prints the last 50 commands that have been executed.
help	(h)	Prints help information.
logend	(log e)	Stops logging.
logstart	(log s)	Starts logging.
repeat	(r)	Repeats a command.
stat	(st)	Prints stats on process and service states.
svrinfo	(si)	Prints server information.
webtobinfo	(wi)	Prints WebtoB system information.
patchinfo		Prints post release patch information.
quit	(q)	Terminates wsmon.
!		Repeats the previous command.



Since wsmon commands are similar to those of **wsadmin console management program** except in the output results, refer to [wsadmin Console Management Program](#) for descriptions of each command.

6.3. Commands

The following are commands provided by WebtoB.

Option	Description
wscfl	Compiles the text WebtoB environment file and creates wsconfig (binary WebtoB environment file).
wsuncfl	Inversely converts the compiled wsconfig (binary WebtoB environment file) to a text environment file.

Option	Description
wsgst	Refers to the binary WebtoB environment file and creates a service table.
wsracd	Supports WebAdmin.
wsboot	Executes all or part of the WebtoB system.
wtdown	Terminates all or part of the WebtoB system.

6.3.1. wscfl

wscfl compiles the text type WebtoB environment file and creates wsconfig (binary WebtoB environment file). wscfl can be used on an operating system that WebtoB system is installed on.

If an error occurs while compiling the input file, the binary WebtoB environment file is not created and compilation stops. The WebtoB environment file made by this command is used in wsboot, and wtdown.

- Usage

```
wscfl -i text WebtoB environment file name [-o binary WebtoB environment file name][-b][-v]
```

Option	Description
<i>-i text WebtoB environment file name</i>	Compiles the target environment file. Used to specify a text based WebtoB environment file name. It is a required option. A path can be specified. If the specified environment file is not found, an error message is displayed.
<i>[-o binary WebtoB environment file name]</i>	Used to specify the binary WebtoB environment file name. Specifying a path is optional. If a path is not specified, the file is created in the current wscfl working directory. If omitted, the file name is set to "wsconfig" by default.
[-b]	Backs up the WebtoB environment file.
[-v]	Checks the WebtoB version.

- Example

- Using [-i] option

The following is an example of creating the binary environment file, wsconfig, in the current directory by compiling http.m, a text type WebtoB environment file.

```
$ wscfl -i http.m
```

The following is an example of creating the binary environment file, wsconfig2, in the current

directory by compiling http.m.

```
$wscfl -i http.m -o wsconfig2
```

- Using [-v] option

The following is an example of checking the WebtoB version.

```
$wscfl -v
```

6.3.2. wsuncfl

wsuncfl inversely converts the compiled wsconfig (binary WebtoB environment file) into a readable text, environment file. wsuncfl can be used on the operating system where WebtoB is installed.

When the original environment file is missing and only the binary file exists, the original environment file can be restored using wsuncfl.

- Usage

```
$ wsuncfl [-i binary WebtoB environment file name] [-o text WebtoB environment file name] [-v]
```

Option	Description
<i>[-i binary WebtoB environment file name]</i>	Binary environment input file to decompile. Specifying a path is optional. If the specified file is not found, an error message is displayed. If omitted, the file name is set to "wsconfig".
<i>[-o text WebtoB environment file name]</i>	Decompiled Text WebtoB environment file name. Required. Specifying a path is optional. If a path is not specified, the file is created in the current wscfl working directory.
<i>[-v]</i>	Checks the WebtoB version.

- Example

The following is an example of decompiling a binary environment file (wsconfig) into a text environment file (sample_r.m) in the current directory.

```
$ wsuncfl -o sample_r.m
```

6.3.3. wsgst

wsgst refers to the binary WebtoB environment file and creates a service table. wsgst can be used on the operating system where WebtoB is installed.

wsgst refers to the SERVER and SERVICE sections in the binary WebtoB environment file made by the wscfl command. wsgst also creates a service table for each server. The table is a list of services provided by the server process. The table is compiled when the server program is being compiled. The table is used to find service locations while the server process computes.

"<server name>_svctab.c", the output of the wsgst command, is created in the directory 'svct' under the directory specified by WEBTOBDIR. The server name is specified in the SERVER section. The contents of each file is the service name provided by the server specified in the SERVICE section.

The services specified in the WebtoB environment file must also be specified in the service table of the server. I.e., if the server name, the service name, or the service's SVRNAME is changed in the SERVER or SERVICE section of the WebtoB environment file, the server program must be recompiled with the service table.

- Usage

```
$ wsgst [-f binary WebtoB environment file name]
```

Option	Description
<code>[-f <i>binary WebtoB environment file name</i>]</code>	Used to specify the binary WebtoB environment file that is referred to. The result of the wscfl command is referred to by wsboot and wsdown. Specifying a path is optional. If the option is omitted, the output file is named "wsconfig" by default.

- Example

The following is an example of creating a service table by referring to wsconfig that is located in the config directory under the directory specified by WEBTOBDIR.

```
$ wsgst
```

The following is an example of creating service tables by servers by referring to the wsconfig2 environment file in the /user1/park/WebtoB/bin directory.

```
$ wsgst -f /user1/park/webtob/config/wsconfig2
```

For example, if an environment file is registered (shown in the following), a service table named webaps_svctab.c is created in the /user1/park/webtob/svct directory.

```
...
```

```
*SERVER
webaps      SVGNAME=webapsg

*SERVICE
svc1        SVRNAME=webaps
svc2        SVRNAME=webaps
...

```

6.3.4. wsracd

wsracd must be started before starting WebAdmin or SysMaster. wsracd can be used on an OS with WebtoB system installed (WebAdmin support is planned for future releases).

- Usage

```
$ wsracd [-k] [-f binary WebtoB environment file name] [-a] [-d]
```

Option	Description
[-k]	Option to not use the binary WebtoB environment file. If set, the default port is used instead. If the WEBTOB_RAC_PORT is set, its value is used as the port (wsracd usually uses this option).
[-f <i>binary WebtoB environment file name</i>]	Executes with the port set in the user specified binary WebtoB environment file.
[-a]	Executes in the anonymous (no authentication) mode. If omitted, authentication is required to connect to wsracd using an OS account. To use the OS account, <code>#{WEBTOBDIR}/bin/wsracd</code> must be changed to the root account and the sticky bit must be set.
[-d]	Execute in the debug mode.

- Example

The following is an example of executing wsracd with the binary WebtoB environment file. If no option is specified, `#{WEBTOBDIR}/config/wsconfig` is used by default. If `wsconfig` file does not exist, an error message is displayed.

```
$ wsracd -a
```

The following is an example of using the options specified with wsboot to execute wsracd.

```
$ wsracd -k
```

The following is an example of using the specified binary file '/user1/park/webtob/config/wsconfig2' to execute wsracd.

```
$ wsracd -f /user1/park/webtob/config/wsconfig2
```

6.3.5. wsmkppd

wsmkppd is used to set the PassPhraseDialog item of the SSL(PROXY_SSL) section.

WebtoB prompts for the password at startup if an encrypted private key is set in the SSL(PROXY_SSL) section. To avoid having to enter the password at startup, set the PassPhraseDialog item. wsmkppd creates a passphrase file that stores the passphrase to apply to the PassPhraseDialog. For information about using the PassPhraseDialog, refer to [SSL Section](#) and [PROXY_SSL Section](#).

- Usage

```
$ wsmkppd [-p passwd] ppd_filename sslname
```

Option	Description
<code>[-p passwd]</code>	Password for the encrypted private key.
<code>ppd_filename</code>	Name of the file that stores the wsmkppd execution results, which will be applied to the PassPhraseDialog item.
<code>sslname</code>	Name set in SSL(PROXY_SSL).

- Example

wsmkppd prompts for the password if it is not supplied using the -p option.

- The following is an example of calling wsmkppd without any options.

```
$ wsmkppd
Usage: wsmkppd [-p passwd] ppd_filename sslname
      -p passwd: password of private key.
      ppd_filename: output file for PassPhraseDialog.
      sslname: name of SSL(PROXY_SSL) section.
```

- The following is an example of executing wsmkppd with the ppd_filename and sslname options.

```
$ wsmkppd ssl.ppd ssl1
New password: (Enter password)
Confirm: (Re-enter password)
Adding Password for ssl1

$ls -al ssl.ppd
-rw----- 1 webtob webtob 19 Feb 19 10:33 ssl.ppd
```

- The following is an example of executing `wsmkppd` with the `-p` option and 'mypasswd'. If `ppd_filename` already exists, the specified password is added to the existing file.

```
$ wsmkppd -p mypasswd ssl.ppd ssl2
Adding Password for ssl2

$ls -al ssl.ppd
-rw----- 1 webtob webtob 42 Feb 19 10:42 ssl.ppd
```

- The following is an example of the result file created by executing `wsmkppd`.

```
$ cat ssl.ppd
ssl1:gJDP+OmN+wI=:
ssl2:8eG+iZjNiu5k:
```

The following is an example of an SSL section settings.

```
*SSL
ssl1      CertificateFile = "/webtob/ssl/newcert.pem",
          CertificateKeyFile = "/webtob/ssl/newcert.pem",
          PassPhraseDialog = "file:/webtob/ssl/ssl.ppd"
```


7. WebtoB Tuning

WebtoB is a commercial web server that targets large-scale professional web services. WebtoB is an enterprise web server that supports a wide range of client needs. WebtoB provides fine tuning functions to optimize the server for any given environment. A well-tuned web server has significant advantages in both efficiency and performance.

This chapter describes the WebtoB tuning items for various environments and those that are important for administrators.

7.1. HTH and HTH_THREAD Configuration

As described earlier, WebtoB has a dedicated process that manages connections with client browsers. This architecture is highly efficient in that a single process, HTH process, manages all task processing, but the process's stability can adversely affect the system performance. An HTH process uses multiple threads and their management can have significant effect on the system performance. Hence, it is important to configure them with care.

7.1.1. HTH

In WebtoB, all user connections are managed by and come through the HTH process. The optimal number of connections managed by a single HTH process depends on the operating system. Each operating system has a different limit on the number of connections that a single process can open.

If a large amount of work is assigned to a process, the CPU usage by the process increases and the process itself may overload. Thus, if many simultaneous connections exist, more processes should be used to distribute the workload. The maximum number of connections that can be handled by a single HTH process is related to the FD setting. To process more concurrent users, therefore, increase the connection limit (maximum 16384) that a single process can open, or increase the HTH value (max 100) in the NODE section.

When the WebtoB environment file is compiled, WebtoB queries the operating system for the maximum connection per process value. The administrator should use this information and predict simultaneous connections to determine the number of HTH processes to use.

In WebtoB, the HTH process manages a majority of request processing. If WebtoB is used as a standalone, set HTH to the number of server CPU cores for efficiency and also consider the thread configuration options. Note that if HTH is set to a value greater than the number of CPU cores, overload may occur due to context-switching.



If the number of HTHs increases, managing the HTHs becomes a burden and HTH caching may be distributed. Therefore, if the number of concurrent users is not large, using the default value for the HTH item (1) of NODE section is recommended.

7.1.2. HTH Thread

Unlike other processes, an HTH process uses multiple threads. The threads must be adjusted carefully to minimize their effect on the system performance.

An HTH process is composed of a selector thread that manages thread communication, and other worker threads.

The following describes each worker thread types.

- Worker Thread

A worker thread handles most of the tasks related to request processing. It is used mostly to read in request file content for HTML requests. If using SSL, it handles SSL handshake, encryption, decryption, and related tasks. It also performs compression as needed, and other tasks including HTTP authentication. It is best to configure the number of worker threads according to the number of CPU cores instead of using a large value. A worker thread does not block on a request since it is not designated to a request but returns the processing to the selector thread after performing its assigned task (file read, etc.).

- Sendfile Thread

A sendfile thread is used to process the sendfile function supported in some OSs. It is usually used to process large request files in the blocking mode. This must be set and used selectively only for large files.

- AccessLog Thread

An accesslog thread is used to record logs after HTH processes an HTTP request. If this thread is not used, the selector thread must send a request to WSM to write logs to the AccessLog. It is recommended to use the default setting.



Since thread management overhead incurs when the number of threads increases, it is recommended to use the default setting when the number of concurrent users is not excessive to prevent context-switching between threads. Note that since worker threads are created for an HTH process, care must be taken to keep the total number of worker threads at a tolerable level.

7.1.3. Example

As the number of SSL processing increases, it is common to process most of the requests in SSL. Since SSL processing requires high CPU usage, it is important to use CPU efficiently. WebtoB supports process and thread configuration according to the CPU environment as shown in the following example.

For example, it is assumed that the number of SSL processing is high in the following server environment.

- Server OS: IBM AIX
- Number of physical CPUs: 8

- Number of SMT-THREADING threads per physical CPU: 4
- Number of virtual CPUs: 32

For the optimal setting in the previous environment, the administrator can set the HTH process count according to the number of physical CPUs and worker thread count according to the SMT-THREADING thread count per physical CPU.

The number of HTH processes and their threads can be specified in the WebtoB environment file as follows:

```
*DOMAIN
webtob1

*NODE
webmain
    WebtoBDir = "$WEBTOBDIR",
    SHMKEY = 69000,
    Docroot = "docs/",
    AppDir = "ap/",
    Port = "5469",
    HTH = 8,
    IndexName = "index.html",
    Logging = "accesslog",
    ErrorLog = "errorlog"

*HTH_THREAD
hworker
    WorkerThreads = 4
```

Using the example configuration, WebtoB starts with 8 HTH processes. Each HTH process uses 4 worker threads and the total number of threads is 32. For efficiency and high performance, it is recommended to configure the process and thread counts according to the physical CPU and virtual CPU of the server.

Since each HTH process distributes the tasks according to each process' load, overload does not occur on a single HTH process.

7.2. Server Process Configuration

Other web servers include all service routines, such as HTML, CGI, and SSI, in a single process. Thus, when a client connects to a server, a single process provides all requested services by a client. In this architecture, it is advantageous that a single process can handle all user requests. However, the process includes unnecessary functions resulting in overhead. To combat this overhead, WebtoB supports the division of the required services. The following is a simple example.

7.2.1. Example

Assume that a shopping mall website has the following distribution of users.

- 60% HTML users
- 20% SERVLET users
- 10% CGI users
- 10% SSI users

Users are generally concentrated in one or two services.

Issues

For most web servers, the number of process set to service is 100. This number is required when 100 concurrent users access the website.

However, looking at the user distribution, 60% of the processes service HTML, 20% service SERVLET, and only 10% each service CGI and SSI. Since other web server processes contain service routines for all available services, most of the service routines are wasting memory. The inefficiency becomes more apparent when users repeatedly request HTML services.

Actions

WebtoB addresses the issue in the following way.

WebtoB separates each service and enables to set the number of processes per service. HTML, the most common service, is handled by an HTH worker thread while other services, such as CGI and SSI, are each assigned to an independent process. Instead of using a single process, each service type is processed separately to allow adjusting the process count for each type.

In the previous example, if it is assessed that there are many CGI application users, the administrator can increase the process count for CGI and reduce those for other processes to prevent resource waste. In this case, WebtoB can be configured with 6 HTH worker threads, 2 SERVLET processes, 1 CGI process, and 1 SSI process.

These settings can be configured in the **SERVER Section's MinProc** and **MaxProc** items in the WebtoB configuration file. Refer to [SERVER Section](#) for more information.

The following is an example of the SERVER section configuration.

```
*SERVER
jsv1      SvgName = jsvg, MinProc = 2, MaxProc = 10
cgi       SvgName = cgig, MinProc = 1, MaxProc = 10
ssi       SvgName = ssig, MinProc = 1, MaxProc = 2
```

Item	Description
MinProc	<p>The number of processes created when WebtoB starts.</p> <p>Specify the initial values in the above cases as follows:</p> <ul style="list-style-type: none"> ◦ Servlet(jsv1): 2 ◦ CGI(cgi): 1 ◦ SSI(ssi): 1
MaxProc	Specifies the maximum number of processes. MaxProc allows for dynamic modification of WebtoB.

Once WebtoB is initialized, the number of processes can be adjusted in the range between MinProc and MaxProc. MinProc cannot have a value greater than MaxProc. Configure in the **wsadmin** management tool.

7.3. Tuning Configuration for BRUN

Server process information can be checked by executing the **st-p** command in the wsadmin environment as follows.

```
$ wsadmin> st -p information

HTH 0(12689): RDY
-----
svr_name  svgname   spr_no(pid) status    count    avg(rt)  clid svc v
-----
cgis      cgig      0( 12690) BRUN      0        0.0000( 6)  0 cgi 0
```

The previous example shows a BRUN status. The BRUN (Blocked Run) status indicates that the server process has not responded to HTH and has momentarily stopped. A server process usually goes into the BRUN state when a client's response speed is slow, when processing a large response, or if the network is slow.

Although a user may determine that the service failed because the server process is in the BRUN status, BRUN does not mean the service failed. If BRUN occurs frequently, it is likely that other requests are being queued.

The tuning points for BRUN, which can be called a type of service delay, are **HttpOutBufSize** and **FlowControl**. Through these two settings, BRUN occurrences can be reduced.



BRUN does not occur during HTML processing. For HTML processing, the worker thread reads in ReadBufSize units, sends the data to the selector thread, and the selector thread initiates a flow control to send the response. Hence, the worker thread does not block.

7.3.1. HttpOutBufSize and FlowControl

The following is an example of HttpOutBufSize and FlowControl configurations set to reduce BRUN occurrences.

```
*SERVER
cgis      SvgName = cgig, MinProc = 5, MaxProc = 10,
          HttpOutbufSize = 4096, FlowControl = 50
```

- **HttpOutBufSize**

HttpOutBufSize (default value: 8192 bytes) is the buffer size used when a server process creates a response and sends it to HTH.

For example, when CGIS processes a 1024KB response, the response is divided into 256 (1024KB / 4KB) pieces of HttpOutBufSize (4KB) size and are sent to HTH. Instead of processing the entire response at once, each piece is sent in a 4KB buffer one at a time. Therefore, the HttpOutBufSize configuration includes the size of the response sent to HTH as well as the number of data pieces to send if the response size exceeds HttpOutBufSize.

Note that for JSV servers, response transmission must be configured on the server and HTH only uses the setting for creating a FlowControl buffer.

- **FlowControl**

HTH receives a response from the server process into the FlowControl buffer, whose size is (HttpOutBufSize * FlowControl), and sends it to the client. Hence, the FlowControl must be configured by adjusting the HttpOutBufSize value and FlowControl size (default value: 50).

If the client is slow to receive the response, the FlowControl buffer may become full and HTH stops reading data from HTMLS since it can no longer receive response from the server process. As a result, BRUN occurs and processing is temporarily suspended until the client receives the response data and the FlowControl buffer becomes half empty at which point HTH resumes reading data from HTMLS.



Increasing the HTH FlowControl buffer size also increases the memory usage by HTH. For example, HTH can use as much data as the FlowControl buffer size (default 200KB = 4k * 50) per request, and the memory usage increases proportionally to the number of requests.

7.3.2. How to Reduce BRUNs

To reduce the number of BRUN occurrences, adjust the FlowControl buffer size with the FlowControl and HttpOutBufSize settings.

The following is an example of changing the HttpOutBufSize and FlowControl settings when processing many 1MB files.

```
*SERVER
```

```
cgis      SvgName = cgig, MinProc = 5, MaxProc = 10,  
         HttpOutbufSize = 8192, FlowControl = 100
```



Note that as `HttpOutBufSize` and `FlowControl` increase, the maximum amount of memory used by HTH can also increase.

Before version 4.1.5, the `HttpOutBufSize` configuration set the file size that HTH cached. If `HttpOutBufSize` is set to a larger value, the amount of memory that HTH uses as cache also increases.

8. WebtoB Security

By default, WebtoB supports both authentication and SSL (Secure Socket Layer) which are popular web security features that protect client information and prevent data leakage. They are required features of e-commerce websites and are supported by most web servers.

8.1. Authentication Method

Authentication method is quite simple. A client sends a user name and password to a web server. The web server verifies the user name and password exist in an encrypted password file. If the user name and password found in the encrypted file match, the user is authenticated.

Authentication can be done individually or in user groups. Configuring the authentication for all users is also possible. Browser and server interaction is as follows.

First, a user connects to a server and requests a document. The classified document can only be accessed by users with permission. The server responds back to user with the "Authentication Required [HTTP 401]" alert. The user inputs its user name and password, which the web server authenticates. Because user name and password are sent to the server via Internet, there is a risk of information leakage. Thus, the user name and password are encrypted when sent to the server.

8.2. SSL

WebtoB supports SSL (Secure Socket Layer), a popular security measure among e-commerce websites. This section describes SSL and then describes how to use SSL in WebtoB. SSL is provided by most web servers, making it convenient to use it.

8.2.1. SSL v3.0

SSL uses X.509 certificates to authenticate communication between server and client. SSL varies in terms of key length: 40 bit and 128 bit. Because 40 bit key SSL is not reliably secure, 128 bit key SSL is recommended. The current SSL version is 3.0.

SSL is a layer based protocol. A message in each layer can contain a length, description, and content fields. When a message is sent through SSL, SSL divides the message into manageable sized blocks and selectively compresses the data. SSL then adds MAC (Message Authentication Codes) to the compressed data and transmits the resulting data. When the data arrives, the data is decrypted and the MAC is authenticated. SSL then decompresses the data to its original form. Afterwards, SSL rearranges blocks and sends the resulting message to the upper level.

To connect to a web server that supports SSL, "**https://***" must be used instead of using the ubiquitous URL expression "**http://***". "**http**" uses Port 80, whereas SSL based "**https**" uses Port 443, allowing a single web browser to use both of "**http://***" and "**https://***" at a same time.

Before SSL is fully working, groundwork is performed at the "handshake" stage.

The groundwork is as follows:

- Client and server exchange their certificates. The signature and expiration date of each certificate is then checked.
- The client generates a private key that is used for encryption and MAC generation. The key is then encrypted with the server's public key and sent to the server.
- When receiving services, the client specifies the encryption algorithm and hash function type to be used. During this process, the client offers the server a list of encryption algorithms that can be supported. The server chooses an algorithm from the list.

8.2.2. SSL vs. SHTTP

SHTTP is a protocol developed by EIT (Enterprise Integration Technology). In SHTTP, headers are added to existing HTTP protocols for security functions. Whereas SSL receives encryption and authentication in units of services, SHTTP receives encryption and authentication in units of messages.

Similar to SSL, encryption and the encryption level can be selected among various encryptions while the client makes a connection to the server using SHTTP.

Characteristics of SHTTP encryptions and algorithms supported by SHTTP are as follows.

- Uses RSA, Diffie-Hellman or Kerberos authentication.
- Public key certificate: X.509 or PKCS-6
- Digital signature algorithm: RSA or DSA(Digital Signature Algorithm).

8.2.3. SSL Encryption

Encryption must be done in either 40 bit or 128 bit key length encoding. The longer the key length, the more secure the encryption. Web servers generally support internal 128 bit encoding. In order to support 128 bit encryption, both the server and client must support it.

8.2.4. Ciphers

A cipher is an algorithm used in encryption. In general when a cipher encrypts data, the more bits that are used, the harder the data becomes to decrypt. In any bilateral encryption process, both parts must use the same cipher. Many kinds of ciphers exist and a server must support the most popular. If a client tries a SSL connection to a server, the client notifies the server of its preferred cipher.

WBSSL currently supports the following ciphers:

Cipher suite	Protocols	KeyExch.	Authen.	Encryption	Mac
TLS_AES_256_GCM_SHA384	TLSv1.3	any	any	AESGCM(256)	AEAD
TLS_CHACHA20_POLY1305_SHA256	TLSv1.3	any	any	CHACHA20/POLY1305(256)	AEAD

TLS_AES_128_GCM_SHA256	TLSv1.3	any	any	AESGCM(128)	AEAD
ECDHE-RSA-AES256-GCM-SHA384	TLSv1.2	ECDH	RSA	AESGCM(256)	AEAD
ECDHE-ECDSA-AES256-GCM-SHA384	TLSv1.2	ECDH	ECDSA	AESGCM(256)	AEAD
ECDHE-RSA-AES256-SHA384	TLSv1.2	ECDH	RSA	AES(256)	SHA384
ECDHE-ECDSA-AES256-SHA384	TLSv1.2	ECDH	ECDSA	AES(256)	SHA384
ECDHE-RSA-AES256-SHA	SSLv3	ECDH	RSA	AES(256)	SHA1
ECDHE-ECDSA-AES256-SHA	SSLv3	ECDH	ECDSA	AES(256)	SHA1
SRP-DSS-AES-256-CBC-SHA	SSLv3	SRP	DSS	AES(256)	SHA1
SRP-RSA-AES-256-CBC-SHA	SSLv3	SRP	RSA	AES(256)	SHA1
SRP-AES-256-CBC-SHA	SSLv3	SRP	SRP	AES(256)	SHA1
DH-DSS-AES256-GCM-SHA384	TLSv1.2	DH/DSS	DH	AESGCM(256)	AEAD
DHE-DSS-AES256-GCM-SHA384	TLSv1.2	DH	DSS	AESGCM(256)	AEAD
DH-RSA-AES256-GCM-SHA384	TLSv1.2	DH/RSA	DH	AESGCM(256)	AEAD
DHE-RSA-AES256-GCM-SHA384	TLSv1.2	DH	RSA	AESGCM(256)	AEAD
DHE-RSA-AES256-SHA256	TLSv1.2	DH	RSA	AES(256)	SHA256
DHE-DSS-AES256-SHA256	TLSv1.2	DH	DSS	AES(256)	SHA256
DH-RSA-AES256-SHA256	TLSv1.2	DH/RSA	DH	AES(256)	SHA256
DH-DSS-AES256-SHA256	TLSv1.2	DH/DSS	DH	AES(256)	SHA256
DHE-RSA-AES256-SHA	SSLv3	DH	RSA	AES(256)	SHA1
DHE-DSS-AES256-SHA	SSLv3	DH	DSS	AES(256)	SHA1
DH-RSA-AES256-SHA	SSLv3	DH/RSA	DH	AES(256)	SHA1
DH-DSS-AES256-SHA	SSLv3	DH/DSS	DH	AES(256)	SHA1
DHE-RSA-CAMELLIA256-SHA	SSLv3	DH	RSA	Camellia(256)	SHA1
DHE-DSS-CAMELLIA256-SHA	SSLv3	DH	DSS	Camellia(256)	SHA1
DH-RSA-CAMELLIA256-SHA	SSLv3	DH/RSA	DH	Camellia(256)	SHA1
DH-DSS-CAMELLIA256-SHA	SSLv3	DH/DSS	DH	Camellia(256)	SHA1
GOST2001-GOST89-GOST89	SSLv3	GOST	GOST01	GOST89(256)	GOST89
GOST94-GOST89-GOST89	SSLv3	GOST	GOST94	GOST89(256)	GOST89
ECDH-RSA-AES256-GCM-SHA384	TLSv1.2	ECDH/RSA	ECDH	AESGCM(256)	AEAD
ECDH-ECDSA-AES256-GCM-SHA384	TLSv1.2	ECDH/ECDSA	ECDH	AESGCM(256)	AEAD
ECDH-RSA-AES256-SHA384	TLSv1.2	ECDH/RSA	ECDH	AES(256)	SHA384
ECDH-ECDSA-AES256-SHA384	TLSv1.2	ECDH/ECDSA	ECDH	AES(256)	SHA384
ECDH-RSA-AES256-SHA	SSLv3	ECDH/RSA	ECDH	AES(256)	SHA1
ECDH-ECDSA-AES256-SHA	SSLv3	ECDH/ECDSA	ECDH	AES(256)	SHA1
AES256-GCM-SHA384	TLSv1.2	RSA	RSA	AESGCM(256)	AEAD
AES256-SHA256	TLSv1.2	RSA	RSA	AES(256)	SHA256
AES256-SHA	SSLv3	RSA	RSA	AES(256)	SHA1
CAMELLIA256-SHA	SSLv3	RSA	RSA	Camellia(256)	SHA1
PSK-AES256-CBC-SHA	SSLv3	PSK	PSK	AES(256)	SHA1
ECDHE-RSA-AES128-GCM-SHA256	TLSv1.2	ECDH	RSA	AESGCM(128)	AEAD
ECDHE-ECDSA-AES128-GCM-SHA256	TLSv1.2	ECDH	ECDSA	AESGCM(128)	AEAD
ECDHE-RSA-AES128-SHA256	TLSv1.2	ECDH	RSA	AES(128)	SHA256
ECDHE-ECDSA-AES128-SHA256	TLSv1.2	ECDH	ECDSA	AES(128)	SHA256
ECDHE-RSA-AES128-SHA	SSLv3	ECDH	RSA	AES(128)	SHA1
ECDHE-ECDSA-AES128-SHA	SSLv3	ECDH	ECDSA	AES(128)	SHA1
SRP-DSS-AES-128-CBC-SHA	SSLv3	SRP	DSS	AES(128)	SHA1
SRP-RSA-AES-128-CBC-SHA	SSLv3	SRP	RSA	AES(128)	SHA1
SRP-AES-128-CBC-SHA	SSLv3	SRP	SRP	AES(128)	SHA1
DH-DSS-AES128-GCM-SHA256	TLSv1.2	DH/DSS	DH	AESGCM(128)	AEAD
DHE-DSS-AES128-GCM-SHA256	TLSv1.2	DH	DSS	AESGCM(128)	AEAD
DH-RSA-AES128-GCM-SHA256	TLSv1.2	DH/RSA	DH	AESGCM(128)	AEAD
DHE-RSA-AES128-GCM-SHA256	TLSv1.2	DH	RSA	AESGCM(128)	AEAD
DHE-RSA-AES128-SHA256	TLSv1.2	DH	RSA	AES(128)	SHA256
DHE-DSS-AES128-SHA256	TLSv1.2	DH	DSS	AES(128)	SHA256
DH-RSA-AES128-SHA256	TLSv1.2	DH/RSA	DH	AES(128)	SHA256
DH-DSS-AES128-SHA256	TLSv1.2	DH/DSS	DH	AES(128)	SHA256
DHE-RSA-AES128-SHA	SSLv3	DH	RSA	AES(128)	SHA1
DHE-DSS-AES128-SHA	SSLv3	DH	DSS	AES(128)	SHA1
DH-RSA-AES128-SHA	SSLv3	DH/RSA	DH	AES(128)	SHA1

DH-DSS-AES128-SHA	SSLv3	DH/DSS	DH	AES(128)	SHA1
DHE-RSA-SEED-SHA	SSLv3	DH	RSA	SEED(128)	SHA1
DHE-DSS-SEED-SHA	SSLv3	DH	DSS	SEED(128)	SHA1
DH-RSA-SEED-SHA	SSLv3	DH/RSA	DH	SEED(128)	SHA1
DH-DSS-SEED-SHA	SSLv3	DH/DSS	DH	SEED(128)	SHA1
DHE-RSA-CAMELLIA128-SHA	SSLv3	DH	RSA	Camellia(128)	SHA1
DHE-DSS-CAMELLIA128-SHA	SSLv3	DH	DSS	Camellia(128)	SHA1
DH-RSA-CAMELLIA128-SHA	SSLv3	DH/RSA	DH	Camellia(128)	SHA1
DH-DSS-CAMELLIA128-SHA	SSLv3	DH/DSS	DH	Camellia(128)	SHA1
ECDH-RSA-AES128-GCM-SHA256	TLSv1.2	ECDH/RSA	ECDH	AESGCM(128)	AEAD
ECDH-ECDSA-AES128-GCM-SHA256	TLSv1.2	ECDH/ECDSA	ECDH	AESGCM(128)	AEAD
ECDH-RSA-AES128-SHA256	TLSv1.2	ECDH/RSA	ECDH	AES(128)	SHA256
ECDH-ECDSA-AES128-SHA256	TLSv1.2	ECDH/ECDSA	ECDH	AES(128)	SHA256
ECDH-RSA-AES128-SHA	SSLv3	ECDH/RSA	ECDH	AES(128)	SHA1
ECDH-ECDSA-AES128-SHA	SSLv3	ECDH/ECDSA	ECDH	AES(128)	SHA1
AES128-GCM-SHA256	TLSv1.2	RSA	RSA	AESGCM(128)	AEAD
AES128-SHA256	TLSv1.2	RSA	RSA	AES(128)	SHA256
AES128-SHA	SSLv3	RSA	RSA	AES(128)	SHA1
SEED-SHA	SSLv3	RSA	RSA	SEED(128)	SHA1
CAMELLIA128-SHA	SSLv3	RSA	RSA	Camellia(128)	SHA1
IDEA-CBC-SHA	SSLv3	RSA	RSA	IDEA(128)	SHA1
PSK-AES128-CBC-SHA	SSLv3	PSK	PSK	AES(128)	SHA1
ECDHE-RSA-RC4-SHA	SSLv3	ECDH	RSA	RC4(128)	SHA1
ECDHE-ECDSA-RC4-SHA	SSLv3	ECDH	ECDSA	RC4(128)	SHA1
ECDH-RSA-RC4-SHA	SSLv3	ECDH/RSA	ECDH	RC4(128)	SHA1
ECDH-ECDSA-RC4-SHA	SSLv3	ECDH/ECDSA	ECDH	RC4(128)	SHA1
RC4-SHA	SSLv3	RSA	RSA	RC4(128)	SHA1
RC4-MD5	SSLv3	RSA	RSA	RC4(128)	MD5
PSK-RC4-SHA	SSLv3	PSK	PSK	RC4(128)	SHA1
ECDHE-RSA-DES-CBC3-SHA	SSLv3	ECDH	RSA	3DES(168)	SHA1
ECDHE-ECDSA-DES-CBC3-SHA	SSLv3	ECDH	ECDSA	3DES(168)	SHA1
SRP-DSS-3DES-EDE-CBC-SHA	SSLv3	SRP	DSS	3DES(168)	SHA1
SRP-RSA-3DES-EDE-CBC-SHA	SSLv3	SRP	RSA	3DES(168)	SHA1
SRP-3DES-EDE-CBC-SHA	SSLv3	SRP	SRP	3DES(168)	SHA1
EDH-RSA-DES-CBC3-SHA	SSLv3	DH	RSA	3DES(168)	SHA1
EDH-DSS-DES-CBC3-SHA	SSLv3	DH	DSS	3DES(168)	SHA1
DH-RSA-DES-CBC3-SHA	SSLv3	DH/RSA	DH	3DES(168)	SHA1
DH-DSS-DES-CBC3-SHA	SSLv3	DH/DSS	DH	3DES(168)	SHA1
ECDH-RSA-DES-CBC3-SHA	SSLv3	ECDH/RSA	ECDH	3DES(168)	SHA1
ECDH-ECDSA-DES-CBC3-SHA	SSLv3	ECDH/ECDSA	ECDH	3DES(168)	SHA1
DES-CBC3-SHA	SSLv3	RSA	RSA	3DES(168)	SHA1
PSK-3DES-EDE-CBC-SHA	SSLv3	PSK	PSK	3DES(168)	SHA1

8.3. Certificate Service

8.3.1. Certificate Authority

In an open network, the encryption of user data and user identification can be performed using a public key encryption algorithm. The core of this algorithm is composed of an encryption key and a decryption key.

A private key is the decryption key which is held private by the decrypting party. The public key is publicly available so that other people can send message to the private key holder by using the matching public key. Verifying that the public key is owned by the actual user is a potential problem.

Certification Authority, CA is the institute that validates public keys.

Certification Authority is a third party trusted by both message sender and receiver. A certificate from Certification Authority is exchanged between sender and receiver. The certificate is validated before the sender and receiver are authenticated and then the message is encrypted. Three types of certificates can be issued from CA: CA Root Certificate, Web Server Certificate, and User Certificate.

8.3.2. Certificate

Certificate is a widely recognized guarantee of the identity of a software, organization, or person. A certificate helps a user to verify a server is trustworthy. Likewise, a server needs to know whether a user is trustworthy or not.

Both server and client certificates exist. A certificate contains its certification authority and where the certificate can be validated. A certificate is generally used to validate a user and to secure user data. A certificate used as a digital signature has legal significance in some countries.

An online certificate is a document that contains information about its certification object. An online certificate can be created personally, or by a certification authority. Online certificates follow the ITU (International Telecommunication Union) X.509 format or the RSA Data Security PKCS-6 and PEM format. Online certificates often have the filename "cert.crt". Generally, this file excludes a private key and holds just a public key.

The following is an example of PKCS-6.

```
Certificate:
Data:
Version: 0 (0x0)
Serial Number: 02:41:00:00:01 ----- ①
Signature Algorithm: MD2 digest with RSA Encryption ----- ②
Issuer: C=US, O=RSA Data Security, Inc.,
OU=Secure Server Certification Authority ----- ③
Validity:
Not Before: Wed Nov 9 15:54:17 1994
Not After: Fri Dec 31 15:54:17 1999 ----- ④
Subject: C=US, O=RSA Data Security, Inc.,
OU=Secure Server Certification Authority
Subject Public Key Info:
Public Key Algorithm: RSA Encryption
Public Key:
Modulus:
00:92:ce:7a:c1:ae:83:3e:5a:aa:89:83:57:ac:25:
01:76:0c:ad:ae:8e:2c:37:ce:eb:35:78:64:54:03:
e5:84:40:51:c9:bf:8f:08:e2:8a:82:08:d2:16:86:
37:55:e9:b1:21:02:ad:76:68:81:9a:05:a2:4b:c9:
4b:25:66:22:56:6c:88:07:8f:f7:81:59:6d:84:07:
65:70:13:71:76:3e:9b:77:4c:e3:50:89:56:98:48:
b9:1d:a7:29:1a:13:2e:4a:11:59:9c:1e:15:d5:49: dd:2d:d6:c8:1e:7b
Exponent: 65537 (0x10001)
Signature Algorithm: MD2 digest with RSA Encryption
Signature:
88:d1:d1:79:21:ce:e2:8b:e8:f8:c1:7d:34:53:3f:61:83:d:
b6:0b:38:17:b6:e8:be:21:8d:8f:00:b8:8b:53:7e:44:67:1:
22:bd:97:27:e0:9c:85:cc:4a:f6:85:3b:b2:e2:be:92:d3:e:
```

```
0d:e9:af:5c:0e:0c:46:95:ff:a1:1c:5e:3e:e8:36:58:7a:7:
a6:0a:f8:22:11:6b:c3:09:38:7e:26:bb:73:ef:00:bd:02:a:
f3:14:0d:30:3f:61:70:7b:20:fe:32:a3:9f:b3:f4:67:52:d:
b4:ee:84:8c:96:36:20:de:81:08:83:71:21:8a:0f:9e:a9
```

In the sample certificate,

- ① is the certificate serial number.
- ② indicates the certificate uses the RSA encryption algorithm and the MD2 (Message Digest 2) message compression algorithm.
- ③ displays information about the certificate authority: C (country), O (organization), and OU (organization unit).

Category	Description
C	country
O	Organization
OU	Organization Unit

- ④ shows that this certificate is valid from November 9, 1994 15:54:17 to December 31, 1999 15:54:17. CA, or Certificate Authority, is a server of the organization that issues certificates.

8.3.3. Certificate Issuing Policy

Secure communication requires a certificate for both client and server. The certificate is a guarantee to a client that the server will provide the right service. The certificate guarantees to a server that the client is the actual requester. The server is obligated to issue a certificate and may request a certificate from a client based on the certificate issuing policy.

In general, there are four types of certificate issuing policies.

- No certificate required.
- Client can optionally provide a valid certificate. If a certificate is provided, it should be verified by the CA (Certification Authority) and match the certificate held by the server.
- Client must provide a valid certificate.
- Client can optionally provide a valid certificate. If a certificate is provided, it does not need to be verified by the CA (Certification Authority) and match the certificate held by the server.

Generally, websites that require a user ID and password do not require a certificate (certificate issuing policy 0), where secure communication is based on the server certificate. On more secure websites, such as online banking sites, certificate issuing policy type 1 is used. In order to use this service, click on **[Submit Certificate]** to submit the certificate to the website. (Generally, a browser can provide the certificate without direct user action.) An electronic bankbook is executed, the server is issued a certificate, and mutual authentication is acquired. Using a certificate is safer than type 0, which only requires a password.

Certificate issuing policy type 1 is similar to how ATM machines are used. An ATM machine takes a card issued from the bank and a password to complete transactions. The card issued from bank is similar to a certificate.

8.3.4. Receiving a Server Certificate from Verisign

Verisign is the most famous private certificate authority in the world. SSL Server Certificates can be purchased from the Verisign homepage (www.verisign.com). An SSL Certificate can be either purchased or a trial certificate can be downloaded. Purchasing a certificate, however, requires additional steps.

A certificate is a guarantee from the certifying organization. Hence, when purchasing a certificate, Verisign requests a business license. If the business is registered in D&B Business Database (Dun & Bradstreet), this step can be very simple. Most of the registered businesses currently are U.S companies.

SSL has a more complex structure and authentication mechanism than described earlier. Moreover, it ensures higher data security since it provides an encryption layer that protects data between TCP and HTTP protocol levels, making it highly difficult to decrypt any data even when sent through an open network like the Internet. Due to its high degree of security, SSL is widely used for security-critical online commerce businesses in Korea.

However, the data encryption between TCP and HTTP protocol levels may affect performance. In particular, other SSL-supporting web server products often experience slow processing speeds due to complex integrations with SSL packages. This is because these web servers use SSL as an external function, creating a loose connection between SSL and the internal engines of the servers. In contrast, WebtoB integrates SSL with internal engines, prioritizing efficiency and eliminating any factors that could compromise performance.

Despite its complexity, using SSL is remarkably simple and straightforward. Virtually all major web browsers support SSL, automatically recognizing any user requests beginning with 'https' and routing them to web servers with SSL capabilities. Subsequently, the web servers authenticate users using the methods described earlier. Importantly, all these processes occur seamlessly within the web servers and client browsers, requiring no manual intervention from the user.

8.4. Using Authentication and SSL

This section describes how to use SSL, and the commands used in authentication.

8.4.1. Authentication

wsmkpw

Setting up authentication in WebtoB is covered in the **AUTHEN section** in the environment configuration. Refer to [AUTHENT Section](#) for more information.

Besides using the WebtoB configuration to authenticate, a valid user name and password are required. Use the **wsmkpw** executable file to create a user name and password. wsmkpw is located under the "bin/" directory, where WebtoB is installed.

The following are wsmkpw usage and descriptions of options.

```
$ wsmkpw [-d] [-p passwd] [-r realm] <file_path> <username>
```

- Item

Item	Description
file_path	Full path to the certificate file
user_name	User name to be used for authentication

- Option

Option	Description
[-d]	Creates a password to authenticate as Digest.
[-p passwd]	The password is entered as an option of a command. If this option is used, the password is exposed in the terminal and may be recorded in a log.
[-r realm]	Configures the realm for Digest authentication.

The following is an example of creating a file named passwd in the directory "/data1/gloria/webtob/auth" and saving a new user named "newuser". The password is encrypted and stored internally. WebtoB uses this file for authentication.

```
wsmkpw /data1/gloria/webtob/auth/passwd newuser  
New Password: *****  
Retype Password: *****
```

Configuring Authentication

If the 'passwd' file is specified in the UserFile item of the **AUTHENT section** in the WebtoB environment configuration, WebtoB uses this file for authentication.

After registering the user name and password by using wsmkpw, add the user name and password to the desired item. Whenever a user requests the item, WebtoB executes the authentication process.

This type of authentication was at one point very popular for its simplicity. However, because the encryption algorithm is simple and easy to decrypt, this authentication process is poor for e-commerce websites. Moreover, in some cases user passwords were leaked onto the internet and decrypted by hackers, resulting in a demand for a more secure authentication and encryption method. In response to this problem, Netscape developed Secure Socket Layer. To enable authentication and SSL in WebtoB, the environment file must be modified.

The **AUTHENT section**, which defines the authentication security in WebtoB, must first be defined. In addition, the items defined in the AUTHENT section must also be defined in each server group. Authentication can be configured in each server group unit.

If a user requires authentication for CGI, define the authentication name in the AUTHENT section of the server group where CGI is defined. Refer to [Configuration Items](#) for more information.

The following shows how to configure the AUTHENT section.

```
*AUTHENT
AUTHENT name      Type,
                  UserFile
```

Item	Description
AUTHENT name	Specifies the name.
Type	Specifies the encryption method. Commonly used encryption methods are Basic, Digest, MD5, and RSA. WebtoB supports Basic and Digest.
UserFile	Specifies the password file created by wsmkpw. Because the password file is used by the AUTHENT section, the file position must be written. The password file must be created before being used.

The following is an example of the AUTHENT section configuration.

```
*AUTHENT
authent1          Type = Basic,
                  UserFile = "/usr/local/webtob/bin/pwfile"
```

Once the environment file is configured as the example shown previously, authentication is ready for use. The final step is to connect the items defined in the AUTHENT section. The WebtoB configuration item used by the user is 'AUTHENT name'. The 'AUTHENT name' item is used to implement the authentication method defined in the AUTHENT section. Refer to [AUTHENT Section](#) for more information on configuration.

The following is an example of configuring authentication for CGI. (authent1 is configured in the previous example.)

```
*SVRGROUP
cgig      NodeName = webmain, SvrType = CGI,
          AuthentName = authent1
```

In the previous example, WebtoB executes authentication whenever a client requests a CGI service. Note that this authentication is performed in the server group unit. Therefore, if CGI group is configured, all corresponding servers execute authentication. For a single, specific CGI to execute authentication, two server groups must be configured.

The following is an example.

```
*SVRGROUP
cgig      NodeName = webmain, SvrType = CGI
cgig_authent NodeName = webmain, SvrType = CGI,
          AuthentName = authent1

*SERVER
cgi1      SvgName = cgig
cgi2      SvgName = cgig_authent

*URI
uri1      Uri = "/cgi-bin/", SvrType = CGI,
          SvrName = cgi1
uri2      Uri = "/cgi/", SvrType = CGI,
          SvrName = cgi2

*ALIAS
alias1    Uri = "/cgi-bin/",
          Realpath = "${WEBTOBDIR}/cgi-bin/"
alias2    Uri = "/cgi/",
          Realpath = "${WEBTOBDIR}/cgi/"
```

After configuring the two groups as shown in the previous example, configure the server group `cgig_authent` for authentication and configure other servers using `cgig`. This configuration is applied to all that can be specified in the `SVRGROUP` section.

8.4.2. Configuring SSL

WebtoB requires additional configurations for SSL. Similar to authentication and logging, SSL is configured in its own dedicated section. Refer to [SSL Section](#) for more information.

In order for WebtoB to use SSL, `SslFlag` must be configured in the `NODE` section or in the `VHOST` section where the Virtual Host will use SSL. By default, WebtoB is not set to use SSL.

`SslFlag` item is a boolean value. '`SslFlag = Y`' triggers the use of SSL. Because the WebtoB is set to not use SSL by default, setting `SslFlag` to '`N`' is unnecessary.

The following is an example of using the `SslFlag` item.

```
SslFlag = Y
```

After setting `SslFlag` to `Y`, SSL must be configured.

The following is the format of SSL section.

```
*SSL
SSL NAME      Certificatefile, CertificateKeyFile,
              CACertificatePath, CACertificateFile,
              VerifyDepth, VerifyClient, RandomFile,
              ...
```

The previous example is the basic format for SSL configuration. Each SSL should be configured

similarly to the AUTHENT section configuration. For the SSL configuration, SSLNAME item is used. Specify the SSLNAME item the same as the SSL section configuration.



It should be noted that authentication must be set in the SVRGROUP section and the SSL section must be set in the NODE section or the VHOST section where SSL is used.

The following is an example of specifying the SslName item.

```
SslName = "ssl1"
```

Define the ssl1 item in the SSL section. The following is an example of SSL section configuration.

```
*SSL
ssl1    CertificateFile = "/user/webtob/ssl/newcert.pem",
        CertificateKeyFile = "/user/webtob/ssl/newcert.pem",
        RandomFile = "/user/webtob/bin/.rnd, 2048",
        RandomFilePerConnection = "/user/webtob/bin/.rnd, 512",
        VerifyClient = 0,
        VerifyDepth = 10,
        FakeBasicAuth = Y
```

The following is an example of applying SSL. In order to use SSL in the NODE section, configure the SslFlag and SslName item.

```
*NODE
test    WebtoBDir = "/user/webtob",
        SHMKEY = 69000,
        HTH = 1,
        Port = "80",
        SslFlag = Y,
        SslName = "ssl1",
        ...
```

In the example, the NODE is set to service SSL using "ssl1", which is configured in the SSL section.

9. WBAPI

WebtoB provides new APIs that help enhance CGI application program functionality.

This chapter describes the APIs provided by WebtoB.

9.1. Overview

In the existing CGI process, when a user request comes in, a new process starts to handle the request. If a user makes a new request, the operating system forks the process corresponding to the program. When a process is forked, a copy of the process is created and made into a child process of the original process. Forking can affect system performance. The burden on the system that forking creates caused existing CGI users to seek new applications, resulting in appearance of Servlet, PHP, ASP, and JSP.

These programs are not written in C but in Java or Script language. While these programs may be easy for programming, discarding existing C programs and developing a new program is also a burden.

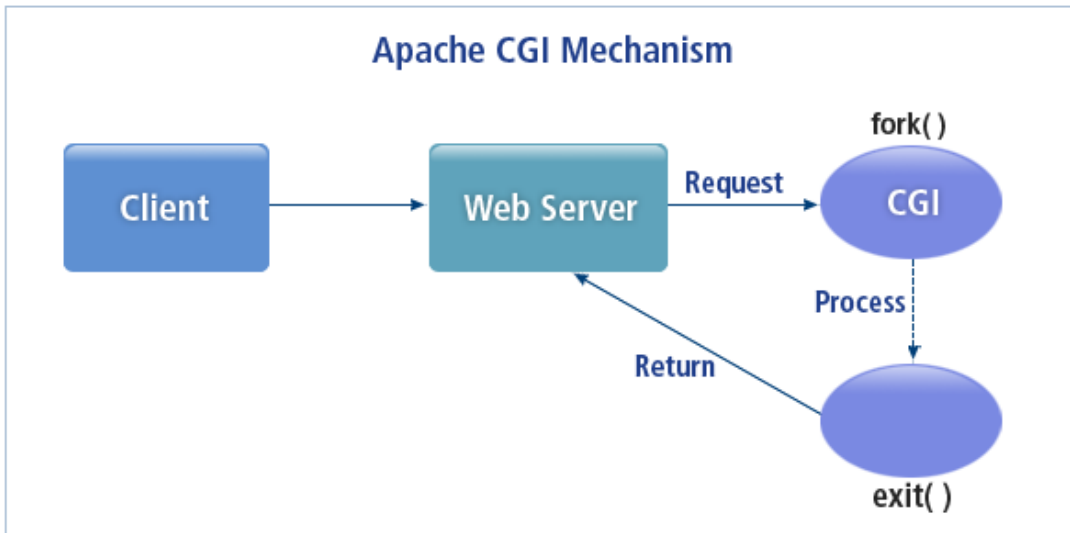
CGI developers must consider whether to discard their existing CGIs and introduce new programs, or to continue operating existing programs and increase the number of servers to reduce CGI burden. WebtoB introduces new APIs to resolve the developers' problems.

As an API for C programs, WBAPI reduces CGI program burden and minimizes CGI program modifications. Through these APIs, many users who use existing CGI programs can increase system performance.

9.2. Concept

As previously mentioned, when a new request comes in, the new program is forked and the request is handled and returned to the web server. The forking process becomes a burden when a system has to process many jobs.

The following figure shows how the CGI mechanism works in the Apache web server.



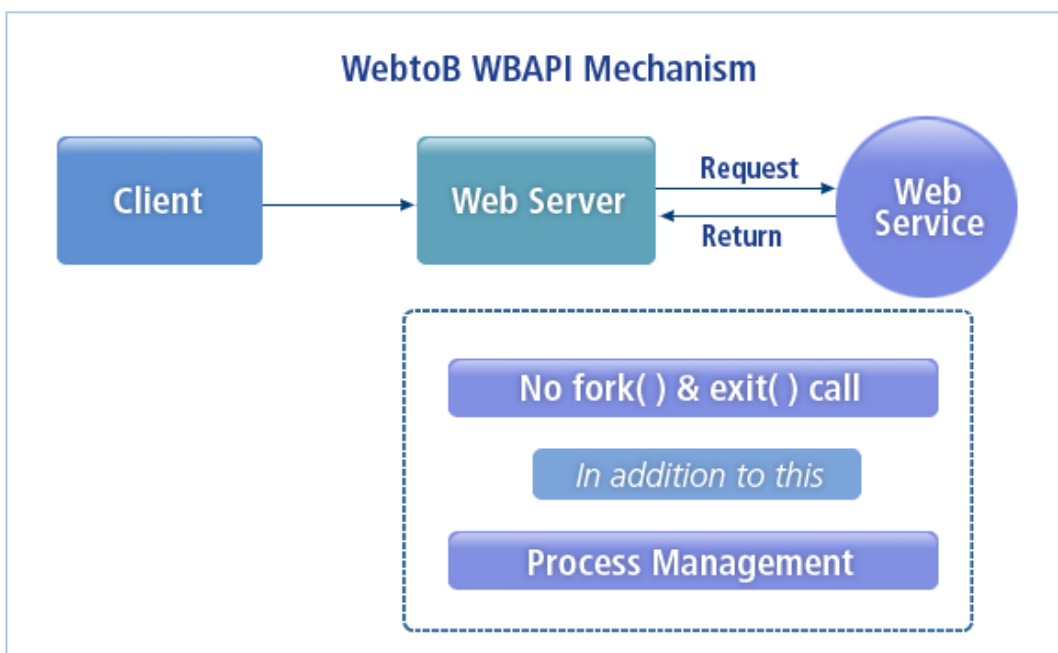
CGI Operation in Apache

In the figure, the structure creates a CGI forking load whenever the web server has a new request.

WBAPI helps resolve the forking problem. In WebtoB, a process is not forked whenever a user sends a request. Instead, WebtoB forks these programs in advance (when web server boots up) and puts these programs into memory. It doesn't exit the process after the user request is processed but waits in the memory for future requests. By not exiting the process and not forking, WebtoB decreases the burden on the system.

WebtoB not only reduces burden but also manages programs designed by WBAPI. Thus, if the program terminates abnormally due to user or design error, a new program is started immediately. WebtoB can also adjust the number of programs. If the number of requests for a specific program is high, the program is executed more frequently in advance to reduce load.

The mechanism of WebtoB's WBAPI call is as follows.



WBAPI in WebtoB

WBAPI is different from general functions that start with main(). A program written with WBAPI is generated as an independent process and works as a WebtoB internal process. Thus, the function does not start with main() but with Service Name. Service Name must be the starting name of the function and must be registered in the WebtoB environment file. Refer to [Environment Configuration](#) for more information.

9.3. Environment Configuration

In order to use WBAPI in WebtoB, WebtoB environment configuration and some programming are required. I.e., In the WebtoB environment configuration file, register the program's development by using WBAPI and how the program will be used.

The following is an example of the WebtoB environment file. The bolded sections are the basic configuration needed to use WBAPI.

```
*DOMAIN
webtob1

*NODE
webmain
    WebtoBDir = "/usr/local/webtob",
    SHMKEY = 99000,
    Docroot = "/usr/local/webtob/docs",
    AppDir = "/usr/local/webtob/ap",
    Hostname = "webmain.tmax.co.kr",
    Port = "8080",
    JsvPort = 9099

*SVRGROUP
cgig    NodeName = webmain, SvrType = CGI
jsvg    NodeName = webmain, SvrType = JSV
ssig    NodeName = webmain, SvrType = SSI
wbapg  NodeName = webmain, SvrType = WEBSTD

*SERVER
cgi     SvgName = cgig, MinProc = 10, MaxProc = 10
ssi     SvgName = ssig, MinProc = 2, MaxProc = 2
jsv1    SvgName = jsvg, MinProc = 10, MAXProc = 10
webaps  SvgName = wbapg, MinProc = 5, MAXProc = 5

*SERVICE
write_board  SvrName = webaps

*URI
uri1  Uri ="/svct/", SvrType = WEBSTD, SvrName = webaps
```

- SVRGROUP section

Configure a server group named wbapg in the host machine named webmain. The following example specifies the WEBSTD server type to use WBAPI.

```
wbapg      NodeName = webmain, SvrType = WEBSTD
```

- SERVER section

The webaps server is defined as `wbapg`, which was defined in the `SVRGROUP` section.

```
webaps      SvgName = wbapg, MinProc = 5, MaxProc = 5
```

The `SVRGROUP` and `SERVER` sections must have the same basic configuration. In most environments, the configuration is the same except for the host machine name. In order to use an actual service, each service must be configured.

- SERVICE section

Register the program developed by `WBAPI`. The program name is same as the function name. Different from the C language style, the program developed by `WBAPI` generally starts with `write_board()`. Because the function starting name becomes the service name, the service name is used to register. Add the `SERVICE` section after the `SERVER` section.

```
*SERVICE  
write_board  SvrName = webaps
```

- URI section

Configure the `URI` section when using the service made by `WBAPI`. If a request begins with the specified `URI`, such as `"/svct/"`, the request will be executed as a `WBAPI` service.

```
*URI  
uri1      Uri ="/svct/",  
          SvrType = WEBSTD,  
          SvrName = webaps
```

9.4. Service Table Creation

In order to start `WebtoB`, compilation through `wscfl` is usually required. After compiling with `wscfl`, a service table must be created in order to use `WBAPI`. Create a table by using the `wsgst` command in the `bin/` directory. If `wsgst` is executed, `WebtoB` creates a service table that has the `_svctab.c` extension in the specified path.

wsgst

The following are description of how to use a service table by using the `wsgst` command.

- Usage

```
$ wsgst [-f binary WebtoB environment file name ]
```

Option	Description
<code>[-f <i>binary WebtoB environment file name</i>]</code>	WebtoB environment file used to create a service table. The default environment file is <code>wconfig</code> .

- Example

The following is an example of using `wsgst`.

```
$ wsgst (References wconfig by default)
$ wsgst -f wconfig2 (When using wscfl -o)
```

This completes the preparations necessary to use WBAPI. Compile the program implemented with WBAPI to generate an executable file and register it as shown in the previous example.

9.5. Program Compilation

Programs developed by WBAPI use special libraries, service tables provided by WebtoB, and a C language library. During compilation, the libraries and service tables must be linked.

The following is an example of a makefile for compilation in UNIX/Linux.

```
#Makefile.common
TARGET= webaps
APOBJS= ap.o

#hp 32bit
CFLAGS = -Ae +DA1.1 +DD32 -O -I$(WEBTOBDIR)
#hp 64bit
#CFLAGS = -Ae +DA2.0W +DD64 +DS2.0 -O -I$(WEBTOBDIR)
#sun 32bit
#CFLAGS = -O -I$(WEBTOBDIR)
#sun 64bit
#CFLAGS = -xarch=v9 -O -I$(WEBTOBDIR)
#Linux
#CFLAGS = -O -I$(WEBTOBDIR)
#ibm 32bit
#CFLAGS = -q32 -O -I$(WEBTOBDIR) -brtl
#ibm 64bit
#CFLAGS = -q64 -O -I$(WEBTOBDIR) -brtl

WEBTOB_INCDIR = $(WEBTOBDIR)/usrinc
WEBTOB_BINDIR = $(WEBTOBDIR)/bin
WEBTOB_LIBDIR = $(WEBTOBDIR)/lib
OBJS = $(APOBJS) $(SVCTOBJ)
SVCTOBJ = $(TARGET)_svctab.o

.SUFFIXES : .v
.c.o:
    $(CC) $(CFLAGS) -c $<

wbaps: $(APOBJS) svct
```

```
$(CC) $(CFLAGS) -L$(WEBTOB_LIBDIR) -o $(TARGET) $(OBJS) $(LIBS) $(USERLIBS)
```

```
svct:
```

```
cp $(WEBTOBDIR)/svct/$(TARGET)_svctab.c .  
$(CC) $(CFLAGS) -I$(WEBTOB_INCDIR) -c $(TARGET)_svctab.c
```

The following is example of a makefile for Windows.

```
#Makefile for Windows  
TARGET = webaps.exe  
  
LIBS    = /link $(WEBTOBDIR)\lib\aps.lib ws2_32.lib  
CFLAGS  = /D _DEBUG /I $(WEBTOBDIR) /Gd /MD /nologo  
  
APOBJ   = $(TARGET:.exe=.obj)  
SVCTSRC = $(TARGET:.exe=_svctab.c)  
SVCTOBJ = $(TARGET:.exe=_svctab.obj)  
OBJS    = $(APOBJ) $(SVCTOBJ)  
  
$(TARGET): svct $(OBJS)  
    cl $(CFLAGS) -o $@ $(OBJS) $(LIBS)  
    copy $@ $(WEBTOBDIR)\ap  
  
svct:  
    copy $(WEBTOBDIR)\svct\$(SVCTSRC)  
  
clean:  
    -del $(OBJS) $(TARGET)
```

9.6. Starting and Using WBAPI

After preparations for WBAPI have been completed, WebtoB and the programs developed by WBAPI start up and load into memory as process type.

CGI conversion is a function of WBAPI. WBAPI can be used to convert existing CGIs, to develop new applications, and may prove useful for developers accustomed to C programming. WebtoB will continue to support and improve WBAPI to eliminate scalability or compatibility problems.

In addition, It is possible to use service functions provided by an existing TP-Monitor through WBAPI.

9.7. WBAPI Types

WBAPI consists of INIT/DONE, ALLOC, GET, PUT/SET, SEND, COOKIE, SESSION, and ETC. This section gives a simple description of each API and its functions.

9.7.1. INIT/DONE API

Functions in the INIT/DONE API execute the initialization/termination routines that must be called

before starting/terminating a service process.

The following is a list of INIT/DONE functions.

Function	Description
wbSvrInit()	Initializes the server.
wbSvrDone()	Notifies WebtoB that the service is complete.

9.7.2. ALLOC API

Functions in the ALLOC API allocate and deallocate user memory.

The following is a list of ALLOC functions.

Function	Description
wbMalloc()	Allocates the user specified amount of memory.
wbFree()	Deallocates the memory allocated by wbMalloc().

9.7.3. GET API

Functions in the GET API read user information from a user request such as request data and method values.

The following is a list of GET API functions.

Function	Description
wbGetAuthType()	Extracts user authentication type.
wbGetContentLength()	Extracts the size of the content sent by the client.
wbGetDocumentRoot()	Extracts the WebtoB Home directory.
wbGetHdr()	Extracts the value of a specific key from the request header.
wbGetDateHdr()	Extracts the data header.
wbGetIntHdr()	Extracts the specified header value as an integer.
wbGetNthHdr()	Extracts the nth header value.
wbGetHdrCount()	Extracts the total number of headers of the request sent by the user.
wbGetData()	Extracts the value of the key from the request data field.
wbGetNthKey()	Extracts the nth key data from the request.
wbGetNthData()	Extracts the nth piece of data from the request data.
wbGetDataCount()	Extracts the number of pieces of data entered from the request.
wbGetValue()	Extracts the nth piece of data of a specific key value from the request.
wbKeyOccur()	Extracts the number of key values from the request data.

Function	Description
wbGetMethod()	Reads the HTTP method as an integer from the request.
wbGetParsedURI()	Extracts the URI information of the request from the client.
wbGetPathInfo()	Extracts the relative path information from the request.
wbGetPathTranslated()	Extracts the absolute path information from the request.
wbGetQueryString()	Extracts a query string from the request URL.
wbGetProtocol()	Extracts the protocol information from the request.
wbGetRemoteUser()	Extracts the user name who sent the request.
wbGetRequestURI()	Extracts the request URI.
wbGetRemoteAddr()	Extracts the remote host IP of the requestor.
wbGetRemoteHost()	Extracts the remote host name of the requestor.
wbGetRemoteIdent()	Extracts the user names found from the server.
wbGetReqLine()	Extracts the first line pointer of the request.
wbGetFileName()	Extracts the file name.
wbGetFileLen()	Extracts the file size.
wbGetScheme()	Extracts the protocol information of the requested service.
wbGetScriptFileName()	Extracts the absolute path where the requested WBAPI service is executed.
wbGetScriptName()	Extracts the path where the requested service is executed.
wbGetServerName()	Extracts the server's host name.
wbGetServerPort()	Extracts the server's port number.
wbGetServerSoftware()	Extracts the server's software information.
wbGetTranslatedURI()	Extracts the actual path by analyzing the requested service URI.
wbGetRequestURL()	Extracts the request URL value.

9.7.4. PUT/SET API

Functions in the PUT/SET API are used to respond to user requests. The functions, similar to **printf** in C or **print script** in Perl, output data processed by WBAPI.

In WBAPI, once a request is processed, the results are stored in a structure called **WBSVCINFO**. The structure is then returned to the web browser. PUT/SET API functions store the data in WBSVCINFO.

The call order of PUT API must be followed for WBAPI to run properly.

- There are the functions that must be called prior to other PUT functions.

`wbSetStatus (WBSVCINFO *rqst, int status, char *status_msg)` must be called prior to calling any function so a response header can be generated.

- Likewise, PUT API functions that create headers must be called before functions that create data

fields.

The following is a list of PUT/SET functions.

Function	Description
wbPutHdr()	Sets a response header.
wbPutIntHdr()	Adds a response header.
wbSetStatus()	Sets the status value of the request.
wbPutStr()	Prints a string value.
wbPut()	Prints user specified sized data.
wbPrint()	Prints user specified content.
wbPutFile()	Reads a file and sends it to the web browser. Used for downloading files.
wbPutPartialFile()	Reads a part of a file and sends it to the browser. File is read starting from offset up to the specified size. If specified size is 0, the entire file is read. Used for downloading files.

9.7.5. SEND API

Functions in the SEND API respond to user requests. Service processing results that are stored in WBSVCINFO structure using PUT API functions. The results are then returned to the browser using SEND API functions.

The following is a list of SEND functions.

Function	Description
wbFlush()	Writes everything in the buffer.
wbSendError()	Checks for errors in the status code.
wbSendRedirect()	Returns the response to the specified address.
wbReturn()	Notifies whether WebtoB is returned.

9.7.6. COOKIE API

Functions in the COOKIE API create, read, and process HTTP style cookies.

The following is a list of COOKIE functions.

Function	Description
wbCreateCookie ()	Creates a new cookie.
wbGetCookie()	Returns a cookie from a list of cookies sent by the web browser.
wbPutCookie()	Adds a response to the specified cookie.

Function	Description
wbCookieGetDomain ()	Returns a domain from the specified cookie.
wbCookieGetName()	Returns the specified cookie name.
wbCookieGetPath()	Returns the cookie path.
wbCookieGetValue()	Returns the cookie value.
wbCookieGetVersion()	Returns the cookie version.
wbCookieSetComment()	Configures the cookie comment field.
wbCookieSetDomain()	Configures a cookie's domain.
wbCookieSetMaxAge()	Configures a cookie's lifespan.
wbCookieSetPath()	Configures a path for the cookie.
wbCookieSetSecure()	Decides whether the cookie should be sent in a secure protocol such as SSL.
wbCookieSetValue()	Configures a new value in the cookie.
wbCookieSetVersion()	Configures a new version of the cookie.

9.7.7. SESSION API

Functions in the SESSION API create sessions and retrieve session data.

The following is a list of SESSION functions.

Function name	Description
wbGetRequestedSessionId()	Extracts the session ID from a user request
wbGetSession()	Creates a session and retrieves the session value, if no session was created.
wbIsRequestedSessionIdValid()	If the requested session is valid and is currently being used, extracts the true value.
wbSessionGetId()	Extracts the unique ID allocated to the session.
wbSessionGetValueNames()	Extracts the array that includes all object names bound to the session.
wbSessionGetCreationTime()	Extracts the session creation time.
wbSessionGetValue()	Extracts the object value and name bound to the session.
wbSessionSetValue()	Binds the object value and name specified in the session.
wbSessionIsNew()	Extracts the value that shows whether the session is new or not.
wbSessionRemoveValue()	Deletes the object bound to the specified name.
WbSessionInvalidate()	Invalidates the session.
wbSessionGetMaxInactiveInterval()	Extracts the session maintenance time specified in the session.

Function name	Description
wbSessionSetMaxInactiveInterval()	Configures the session maintenance time for the specified session.
wbSessionGetLastAccessTime()	Extracts the time when client sent the last request.

9.7.8. ETC API

Functions in the ETC API upload files and extracts the error number from WebtoB

The following is a list of ETC functions.

Function	Description
wbGetErrno()	Extracts the error number from WebtoB.
wbSaveFile()	Uploads a file.

9.8. CGI Conversion Using WBAPI

WBAPI will provide multi-thread functions in the future. For this, a specific data structure must be declared in each function. If a WBAPI program is written first, the service name is registered. At the time the program is registered, the structure **WBSVCINFO** is declared. WBSVCINFO must be declared in the head of each WBAPI function. All WBAPI functions in a program must have a structure with the service name declared as the first argument.

This section describes how WBAPI is used to convert existing CGI programs.

9.8.1. CGI Program

The following is an example of implementing a simple bulletin board with CGI.

```
#include "qDecoder.h"

int strcheck(char *str) {
    if (str == NULL)
        return 0;
    if (strlen(str) == 0)
        return 0;
    return 1;
}

int main(void) {
    char *name, *title, *doc;
    char *email, *homepage;

    /* Get input values */
    name = qValue("writer");
    title = qValue("title");
```

```

doc = qValue("doc");
email = qValue("email");
homepage = qValue("homepage");

/* Check if input values are valid */
if (!strcheck(name))
    qError("Enter a name.");
if (!strcheck(title))
    qError("Enter a title.");
if (!strcheck(doc))
    qError("Enter a body.");
if (strcheck(email))
    if (!qCheckEmail(email))
        qError("Enter a valid E-Mail address.");
if (strcheck(homepage))
    if (!qCheckURL(homepage))
        qError("Enter a valid Homepage URL.");

/* Output result – Input check */
qContentType("text/html");
printf("<HTML>\n\n");
printf("<HEAD><TITLE>Bestbook bulletin board – Post check </TITLE></HEAD>\n\n");
printf("<BODY>\n");
printf("<H2> Bestbook bulletin board - Post check </H2>\n");
printf("<HR WIDTH=600 ALIGN=left>\n<BR>");

if (strcheck(email))
    printf("<A HREF=\\\"mailto:%s\\\">%s</A>", email, name);
else printf("%s", name);

if (strcheck(homepage))
    printf("<SMALL><A HREF=\\\"%s\\\">%s</A></SMALL>", homepage, homepage);

printf("You posted as follows.<BR><BR>\n");
printf("<TABLE WIDTH=600 BORDER=1>\n");
printf("<TR><TD><B>Title</B> : %s</TD></TR>\n", title);
printf("<TR><TD>%s</TD></TR>\n", doc);
printf("</TABLE>\n\n<BR>Posted in the board successsfully.\n");
printf("</BODY>\n\n</HTML>");
qFree();
}

```

Most CGI program structures are simple. After performing a specific job based on the user input value sent through QUERY_STRING, the program returns the result via HTML. Making an HTML document is done through the **printf** function.

By default, several functions in the program are provided by C. The qValue function reads input values while the strcheck function checks the validity of a variable value. These functions are also supported by a simple library type.

The previous CGI program is analyzed as follows:

1. The CGI program uses variables named writer, title, doc, email, and homepage. These variables are user specified values. In the following program, the writer, title, doc, email, and homepage values are extracted from a request using the qValue function and set their respective CGI variables.

```

name = qValue("writer");
title = qValue("title");
doc = qValue("doc");
email = qValue("email");
homepage = qValue("homepage");

```

- After extracting each variable value, verify the values were inputted to each variable by using the `strcheck` function.

```

if (!strcheck(name))
    qError("Enter a name.");

if (!strcheck(title))
    qError("Enter a title.");

if (!strcheck(doc))
    qError("Enter a body.");

if (strcheck(email))
    if (!qCheckEmail(email))
        qError("Enter a valid E-Mail address.");

if (strcheck(homepage))
    if (!qCheckURL(homepage))
        qError("Enter a valid Homepage URL.");

```

- If all the user variables sent to CGI are valid, all functions can be called successfully. Afterwards, the values must be posted onto the bulletin board and the output type must be made.

The following function shows that the output format supported by this CGI program is HTML.

```

qContentType("text/html");

```

- Use the `printf` function to output the resulting value. This process similar to editing an HTML document.

The following example outputs HTML tags by using the `printf` function.

```

printf("<HTML>\n\n");
printf("<HEAD><TITLE>Bestbook bulletin board - Post check</TITLE></HEAD>\n\n");
printf("<BODY>\n");
printf("<H2> Bestbook bulletin board - Post check</H2>\n");
printf("<HR WIDTH=600 ALIGN=left>\n<BR>");

if (strcheck(email))
    printf("<A HREF=\"mailto:%s\">%s</A>", email, name);
else printf("%s", name);

if (strcheck(homepage))
    printf("<SMALL>(<A HREF=\"%s\">%s</A></SMALL>", homepage, homepage);

printf("You posted as follows.<BR><BR>\n");

```

```
printf("<TABLE WIDTH=600 BORDER=1>\n");
printf("<TR><TD><B>Title</B> : %s</TD></TR>\n", title);
printf("<TR><TD>%s</TD></TR>\n", doc);
printf("</TABLE>\n\n<BR>Posted in the board successfuly.\n");
printf("</BODY>\n\n</HTML>");
```

9.8.2. WBAPI Program

The following is a WBAPI program that has the same functionality and structure as a CGI program.

1. Values for each variable must be extracted.

A CGI program uses the `qValue` function but WBAPI uses the **`wbGetData`** function. `wbGetData` extracts each variable value.

```
writer = wbGetData(rqst, "writer");
title = wbGetData(rqst, "title");
doc = wbGetData(rqst, "doc");
email = wbGetData(rqst, "email");
homepage = wbGetData(rqst, "homepage");
```

2. Use `strcheck` to check the read values. The process is same as that of a CGI program.
3. WBAPI can read values from a header as well as read user environment variables. The following is done using the **`wbGetHdr`** function.

The following example reads the Content-Length value.

```
hd = wbGetHdr(rqst, "Content-Length");
```

4. After variable value extraction is complete, an output document must be made.

CGI programs use the `printf` function while WBAPI uses the **`wbPrint`** function. Each instance of the `printf` function in the CGI program can be replaced with `wbPrint`. While the resulting format may differ slightly, the differences are limited to HTML tags. There are no differences when making a bulletin board.

The header information made by `printf` in CGI program is made by using **`wbPutHdr`** in WBAPI. Use `wbPrint` for HTML content but use `wbPutHdr` for header content.

The following is an example of creating a header.

```
wbPutHdr("ContentType", "text/html");
```

The following is an example of creating HTML tags.

```
wbPrint(rqst, "<HTML>\n\n");
wbPrint(rqst, "<HEAD> <TITLE> webtob Write Board </TITLE> </HEAD>\n\n");
wbPrint(rqst, "<BODY>\n");
```



```
wbPrint(rqst, "<H2> Bestbook</H2>\n");
wbPrint(rqst, "<H3>Writer is%s </H3> \n",writer);
wbPrint(rqst, "<H3>Title is%s </H3> \n",title);
wbPrint(rqst, "<HR>\n");
wbPrint(rqst, "<H2> %s </H2> \n",doc);
wbPrint(rqst, "<H3> newvalue :%s%d</H3> \n",hd,rt);
wbPrint(rqst, "<HR ALIGN=left>\n<BR>");
wbPrint(rqst, ".<BR><BR>\n");
wbPrint(rqst, "<TABLE WIDTH=600 BORDER=1>\n");
wbPrint(rqst, "</TABLE>\n\n<BR>.\n");
wbPrint(rqst, "</BODY>\n\n</HTML>");
```

5. After completing the HTML and header sections, instead of using return, use **wbReturn** to send the results to WebtoB.

```
wbReturn(rqst, WBSUCCESS);
```

WBSUCCESS within a parenthesis means the result was successful.

Appendix A: Environment Configuration Example

This appendix provides examples of a basic environment file and an environment file for integrating with JEUS.

A.1. Basic Configuration

The following is a basic WebtoB environment configuration file without JEUS integration.

```
*DOMAIN
webtob

*NODE
mynode
  WebtoBDir = "$WEBTOBDir",
  SHMKEY = 56000,
  Docroot = "docs/",
  Port = "8080",
  HTH = 1,
  Logging = "accesslog",
  ErrorLog = "errorlog",
  SysLog = "systemlog"

*HTH_THREAD
hworker
  WorkerThreads = 8

*SVRGROUP
cgig
  NodeName = mynode,
  SvrType = CGI
ssig
  NodeName = mynode,
  SvrType = SSI

*SERVER
cgi
  SvrName = cgig,
  MinProc = 4,
  MaxProc = 10
ssi
  SvrName = ssig,
  MinProc = 2,
  MaxProc = 10

*URI
cgi_bin
  Uri = "/cgi-bin/",
  Svrtype = CGI

*EXT
htm
```

```

Mimetype = "text/html",
SvrType = HTML

*ALIAS
alias
  Uri = "/cgi-bin/",
  Realpath = "${WEBTOBDIR}/cgi-bin/"

*LOGGING
accesslog
  Format = "default",
  Filename = "log/access_%Y%M%D%.log"
errorlog
  Format = "ERROR",
  Filename = "log/error_%Y%M%D%.log"
systemlog
  Format = "SYSLOG",
  Filename = "log/system_%Y%M%D%.log"

```

A.2. WebtoB and JEUS Integration Configuration

The following is an example of a WebtoB environment configuration file used for JEUS integration.

```

*DOMAIN
webtob

*NODE
mynode
  WebtoBDir = "$WEBTOBDIR",
  SHMKEY = 84565,
  Docroot = "docs/",
  Port = "8080",
  HTH = 1,
  Logging = "accesslog",
  ErrorLog = "errorlog",
  SysLog = "systemlog",
  JsvPort = 9999

*HTH_THREAD
hworker
  WorkerThreads = 8

*SVRGROUP
htmlg
  NodeName = mynode,
  SvrType = HTML
jsvg
  NodeName = mynode,
  SvrType = JSV

*SERVER
MyGroup
  SvgName = jsvg,
  Minproc = 10,
  MaxProc = 10

```

```

*URI
examples
  Uri = "/examples/",
  SvrType = JSV

*EXT
htm
  Mimetype = "text/html",
  SvrType = HTML
html
  Mimetype = "text/html",
  SvrType = HTML
jsp
  Mimetype = "application/jsp",
  SvrType = JSV

*LOGGING
accesslog
  Format = "default",
  Filename = "log/access_%Y%M%D%.log",
  Option="Sync"
errorlog
  Format = "ERROR",
  Filename = "log/error_%Y%M%D%.log"
systemlog
  Format = "SYSLOG",
  Filename = "log/system_%Y%M%D%.log"

```

A.2.1. JEUS 6 Integration Configuration

The following is an example of a JEUS environment configuration file. Configure <webtob-listen> for WebtoB integration.

<WEBMain.xml>

```

<!-- JEUS Configuration File -->
<?xml version="1.0"?>
<!DOCTYPE web-container PUBLIC "-//Tmax Soft., Inc.//DTD WEB Main Config 4.0//EN"
      "http://www.tmaxsoft.com/jeus/dtd/4.0/web-main-config.dtd">
<web-container>
  <context-group>
    <group-name>MyGroup</group-name>
    <group-docbase>webapps</group-docbase>
    <session-config>
      <timeout>20</timeout>
      <shared>>true</shared>
    </session-config>
    <logging>
      <error-log>
        <target>stdout</target>
        <level>information</level>
        <buffer-size>0</buffer-size>
        <valid-day>1</valid-day>
      </error-log>
      <user-log>
        <target>file</target>

```

```

        <buffer-size>0</buffer-size>
        <valid-day>1</valid-day>
    </user-log>
    <access-log>
        <target>file</target>
        <buffer-size>0</buffer-size>
        <valid-day>1</valid-day>
        <log-format>
            <time-format>default</time-format>
        </log-format>
    </access-log>
</logging>
<context>
    <context-name>examples</context-name>
    <context-path>/examples</context-path>
</context>
<context>
    <context-name>test</context-name>
    <context-path>/test</context-path>
</context>
<webserver-connection>
<!--
    <http-listener>
        <listener-id>http1</listener-id>
        <port>8989</port>
        <output-buffer-size>8192</output-buffer-size>
        <thread-pool>
            <min>25</min>
            <max>30</max>
            <step>2</step>
            <max-idle-time>1000</max-idle-time>
        </thread-pool>
    </http-listener>
-->
    <webtob-listener>
        <listener-id>webtob1</listener-id>
        <port>9999</port>
        <hth-count>1</hth-count>
        <webtob-address>192.168.1.43</webtob-address>
        <registration-id>MyGroup</registration-id>
        <thread-pool>
            <min>10</min>
            <max>10</max>
            <step>2</step>
        </thread-pool>
        <disable-pipe>true</disable-pipe>
    </webtob-listener>
</webserver-connection>
</context-group>
</web-container>

```

A.2.2. Embedded Servlet Engine Integration Configuration (JEUS 8)

The following is an example of an environment configuration file used for integration with an embedded Servlet engine. Configure `<webtob-connector>` for WebtoB connection.

<domain.xml>

```
<!-- JEUS 8(Embedded servlet engine) configuration file -->
<?xml version="1.0" encoding="UTF-8"?>
<domain xmlns="http://www.tmaxsoft.com/xml/ns/jeus" version="7.0">
  <id>8322013</id>
  <admin-server-name>adminServer</admin-server-name>
  <servers>
    <server>
      <name>adminServer</name>

      <listeners>
        <base>base</base>
        <listener>
          <name>base</name>
          <listen-address>0.0.0.0</listen-address>
          <listen-port>9736</listen-port>
          <use-dual-selector>>false</use-dual-selector>
          <backlog>128</backlog>
          <read-timeout>30000</read-timeout>
          <reserved-thread-num>0</reserved-thread-num>
        </listener>
        <listener>
          <name>http-server</name>
          <listen-port>8088</listen-port>
          <use-dual-selector>>false</use-dual-selector>
          <backlog>128</backlog>
          <read-timeout>30000</read-timeout>
          <reserved-thread-num>0</reserved-thread-num>
        </listener>
      </listeners>

      <web-engine>
        <web-connections>
          <webtob-connector>
            <name>webtob</name>
            <output-buffer-size>8192</output-buffer-size>
            <postdata-read-timeout>30000</postdata-read-timeout>
            <max-post-size>-1</max-post-size>
            <max-parameter-count>-1</max-parameter-count>
            <max-header-count>-1</max-header-count>
            <max-header-size>-1</max-header-size>
            <max-querystring-size>8192</max-querystring-size>
            <network-address>
              <port>9999</port>
              <ip-address>localhost</ip-address>
            </network-address>
            <thread-pool>
              <number>10</number>
              <thread-state-notify>
                <max-thread-active-time>0</max-thread-active-time>
                <interrupt-thread>>false</interrupt-thread>
                <active-timeout-notification>>false</active-timeout-notification>
                <notify-threshold-ratio>0.0</notify-threshold-ratio>
                <restart-threshold-ratio>0.0</restart-threshold-ratio>
              </thread-state-notify>
            </thread-pool>
            <hth-count>1</hth-count>
            <registration-id>MyGroup</registration-id>
          </webtob-connector>
        </web-connections>
      </web-engine>
    </server>
  </servers>
</domain>
```

```

        <reconnect-interval>5000</reconnect-interval>
    </webtob-connector>
<!--
    <http-listener>
        <name>http1</name>
        <output-buffer-size>8192</output-buffer-size>
        <postdata-read-timeout>30000</postdata-read-timeout>
        <max-post-size>-1</max-post-size>
        <max-parameter-count>-1</max-parameter-count>
        <max-header-count>-1</max-header-count>
        <max-header-size>-1</max-header-size>
        <max-querystring-size>8192</max-querystring-size>
        <server-listener-ref>http-server</server-listener-ref>
        <thread-pool>
            <min>10</min>
            <max>20</max>
            <max-idle-time>300000</max-idle-time>
            <max-queue>-1</max-queue>
        </thread-pool>
        <keep-alive>true</keep-alive>
        <server-access-control>>false</server-access-control>
    </http-listener>
-->
    </web-connections>
</web-engine>

</server>
</servers>
</domain>

```

Appendix B: CGI Application Example

This appendix describes how CGI applications work and provides bulletin board implementations in C and Perl.

B.1. Overview

CGI is used to process static HTML pages to deliver dynamic content.

A user can interact dynamically with an HTML web page by calling an application program. The application program processes the request and returns the result, which is displayed on the page as if the web page itself processed the request.

An electronic bulletin board system (BBS) will serve as an example to illustrate the use of CGI. A web bulletin board receives input data from a user. The BBS then uses that data to generate an HTML document and sends the document to the browser. The role of CGI in the BBS is to convert the user input into an HTML document. The HTML document is then displayed by the browser. Users can access the bulletin board to post data in real time through CGI. A bulletin board program acts as a "pencil" to write data directly on a web page.

This appendix implements a web service by using CGI. The example bulletin board is written in C while the guest book is written in Perl.

B.2. Implementing BBS in C

This example describes a bulletin application written in C for UNIX. This web page can post user text input and address by calling "board.cgi" from "board.html".

B.2.1. Environment File

The following is a sample environment file.

<sample CGI.m>

```
#Configuration file ( sample CGI.m )

*DOMAIN
webtob

*NODE
webmain      WEBTOBDIR = "$WEBTOBDIR",
              SHMKEY = 72000, HTH=1,
              DOCROOT = "docs/",
              PORT = "8080"

*HTH_THREAD
hworker
  WorkerThreads = 8
```



```

*SVRGROUP
cgig          NODENAME = webmain, SvrType = CGI

*SERVER
cgi          SVGNAME = cgig, MinProc = 1, MaxProc = 5

*URI
uri1         Uri = "/cgi-bin/", SvrName = cgi, Svrtype = CGI

*ALIAS
alias1       URI = "/cgi-bin/",
              RealPath = "${WEBTOBDIR}/cgi-bin/"

```

B.2.2. HTML Page

The following is a sample HTML page code.

<board.html>

```

#board.html
<HTML>
<HEAD><TITLE>WebToB Board</TITLE></HEAD>
<BODY>
<H2><big>WebToB Board Upload</big></H2>
<HR WIDTH=500 ALIGN=left><BR>
<FORM METHOD=post ACTION="/cgi-bin/board.cgi">
<TABLE WIDTH=500 BORDER=0>
  <TR>
    <TD>Writer</TD>
    <TD><INPUT NAME=writer SIZE=20></TD>
  </TR>
  <TR>
    <TD>Title</TD>
    <TD><INPUT NAME=title SIZE=50></TD>
  </TR>
  <TR>
    <TD COLSPAN=2><B>Contents</B></TD>
  </TR>
  <TR>
    <TD COLSPAN=2>
      <TEXTAREA NAME=doc COLS=60 ROWS=10></TEXTAREA>
    </TD>
  </TR>
  <TR>
    <TD>E-Mail</TD>
    <TD><INPUT NAME=email SIZE=40></TD>
  </TR>
  <TR>
    <TD>Home Page</TD>
    <TD>
      <INPUT NAME=homepage
        SIZE=40 VALUE="http://">
    </TD>
  </TR>
</TABLE>

```

```

<BR>
<INPUT TYPE=submit VALUE="    Submit    ">
<INPUT TYPE=reset VALUE="    Clear    ">
</FORM>
</BODY>
</HTML>

```

B.2.3. CGI Source Code

The following is a sample CGI source code.

<board.c>

```

#attachment 3: board.c
#include "qDecoder.h"

int strcheck(char *str) {
    /* Check if string is NULL or length is 0 */
    if (str == NULL)
        return 0;
    if (strlen(str) == 0)
        return 0;
    return 1;
}

int main(void) {
    char *name, *title, *doc;
    char *email, *homepage;

    /* Get input vlaues */
    name = qValue("writer");
    title = qValue("title");
    doc = qValue("doc");
    email = qValue("email");
    homepage = qValue("homepage");

    /* Check input values */
    if (!strcheck(name))
        qError("Type Your Name !");
    if (!strcheck(title))
        qError("Type Title !");
    if (!strcheck(doc))
        qError("Write Your Messages !");
    if (strcheck(email))
        if (!qCheckEmail(email))
            qError("Type Your E-Mail !");
    if (strcheck(homepage))
        if (!qCheckURL(homepage))
            qError("Type Your Homepage URL !");

    /* Add to bulletin board here. */
    /* Print the result - Check Input */
    qContentType("text/html");
    printf("<HTML>\n\n");
    printf("<HEAD><TITLE>CGI Board TEST</TITLE></HEAD>\n\n");
    printf("<BODY>\n");

```

```

printf("<H2> CGI Board TEST </H2>\n");
printf("<HR WIDTH=600 ALIGN=left>\n<BR>");
if (strcheck(email))
    printf("<A HREF=\"mailto:%s\">%s</A>",email,name);
else
    printf("%s", name);

if (strcheck(homepage))
    printf( "<SMALL>(<A HREF=\"%s\">%s</A></SMALL>",
           homepage,
           homepage );

printf("wrote<BR><BR>\n");
printf("<TABLE WIDTH=600 BORDER=1>\n");
printf("<TR><TD><B>Title</B> : %s</TD></TR>\n", title);
printf("<TR><TD>%s</TD></TR>\n", doc);
printf("</TABLE>\n\n<BR>Upload Successful !\n");
printf("</BODY>\n\n</HTML>");
qFree();
}

```

B.2.4. Configuration and Checking the Results

The following are the steps for configuration and checking the results.

1. Modify the sample_cgi.m environment file to correspond with the system.
2. Move HTML documents to the DocRoot directory.
3. Modify the ACTION="/cgi-bin/board.cgi" path in board.html to the path that points to board.cgi in its environment.
4. Move board.c, qDecoder.h, and qDecoder.c to the working directory where CGI is executed.
5. Once board.c is compiled, board.cgi is created. (cc or gcc must be installed)

```
$ cc -o board.cgi board.c qDecoder.c
```

or

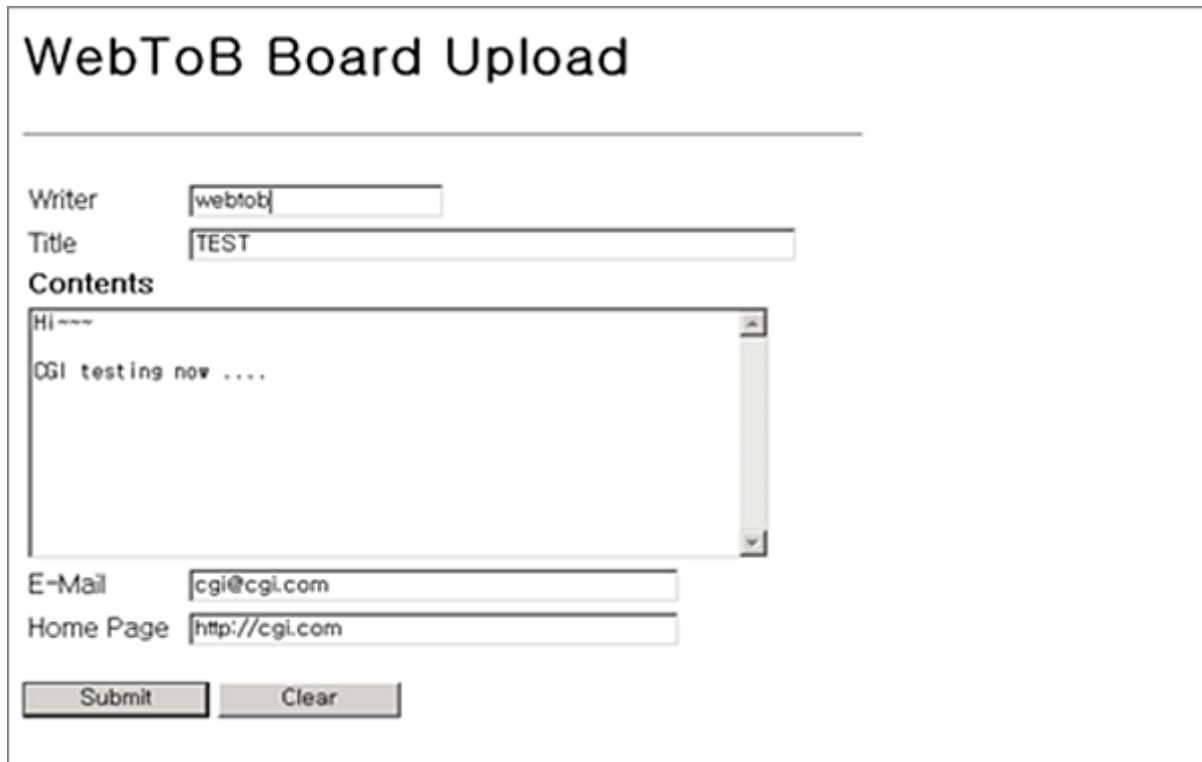
```
$ gcc board.c qDecoder.c -o board.cgi
```

6. Change permission of board.cgi to 755.

```
$ chmod 755 board.cgi
```

7. Start WebtoB and load the bulletin board in a web browser. (Use the IP address of the machine where WebtoB is currently running.)

8. The following screen is displayed.



The screenshot shows a web form titled "WebToB Board Upload". It contains several input fields: "Writer" with the value "webtob", "Title" with the value "TEST", "Contents" with a text area containing "Hi~~~" and "CGI testing now", "E-Mail" with the value "cgi@cgi.com", and "Home Page" with the value "http://cgi.com". At the bottom, there are two buttons: "Submit" and "Clear".

Screen shot of C based Bulletin Board System

Fill out the form and click on **[submit]**. board.cgi is called and the following screen that shows the input is entered to the board is displayed.



The screenshot shows a web page titled "CGI Board TEST". It displays the following information: "webtob(http://cgi.com) wrote:", a box containing "Title : TEST" and "Hi~~~ CGI testing", and the message "Upload Successful !".

Article Posted on C based BBS

B.3. Implementing BBS in Pearl

Perl is a scripting language, which is not compiled. Perl allows for fast development because it can be executed and debugged as soon as it is written. This is the main reason why scripting languages are often preferred. Perl is also free for download. Introduction to Perl itself will not be covered here.

This section describes the environment file required to operate Perl in WebtoB. A sample guest book implemented in Perl is also shown.

B.3.1. Environment File

The following is an example of an environment configuration file.

<sample_Perl.m>

```
#Configuration file ( sample_perl.m )

*DOMAIN
webtob1

*NODE
webmain      WEBTOBDIR = "/usr/local/webtob",
              SHMKEY = 72000, HTH=1,
              DOCROOT = "/usr/local/webtob/docs",
              PORT = "9989"

*HTH_THREAD
hworker
  WorkerThreads = 16

*SVRGROUP
cgig         NODENAME = webmain,
              SvrType = CGI

*SERVER
cgi         SVGNAME = cgig,
              MinProc = 1,
              MaxProc = 5

*URI
uri1        Uri = "/cgi-bin/",
              SvrName = cgi,
              Svrtype = CGI

*ALIAS
alias1      URI = "/cgi-bin/",
              RealPath = "/usr/local/webtob/docs/cgi-bin/"
```

B.3.2. Perl Script

The following example is a guestbook written in PERL.

<guestbook.cgi>

```

#guestbook.cgi

#!/usr/local/bin/perl

# Base URL (Usually Homepage address)
$Base_href      = 'http://webmain.tmax.co.kr:9081/';

# Path to the file based on $base_href
$Cgi_file       = 'cgi-bin/guestbook.cgi';

# lock directory path
$Lock_dir       = 'lock/guestbook';

# File containing BBS data
$Bbs_file       = 'book_data';

# Title to link
$Link_name      = 'Back to Home';
# URL to link (based on $base_href)
$Link_url       = './';

# Title
$Bbs_title      = 'Guest Book';

# Administrator name
$Bbsmaster_name = 'webtob';

# Administrator email address
$Bbsmaster_email = 'webotb@tmax.co.kr';

# Maximum size of a message (byte) (To prevent undesired emails)
$Max_msg_size   = 8000;
$Titlecolor     = '#800080';
$Bgcolor        = '#fafff8';
$Textcolor      = '#000000';
$Linkcolor      = '#401080';

$Bbs_write_title = 'WRITE';
$Bbs_read_title  = 'READ';

$Rem_name       = '';
$Rem_email      = '';
$Rem_msg        = '' .
                '' .
                '';

$Btn_write      = 'Send';
$Btn_read_old   = 'Old Message';
$Btn_read_new   = 'New Message';

$Rem_bbs_end    = 'End of Messages';
$Rem_bbs_back   = 'Go Top';

$Bbs_logo       = '';
$Bbs_logo_addr  = 'http://';

$Delimiter      = chr(30);
$Prev_num_skip  = 0;

```

```

##### main routine #####

&read_file;
&read_form;

if ($ENV{'REQUEST_METHOD'} eq "POST" && $Name ne "" && $Msg ne "" ) {
    &remove_html_tags;    # remove HTML tags
    &new_msg;              # make new BBS msg, use &clock
    &save_msg;            # save BBS msg
}

&show_header;          # show header
&show_entry_form;      # show BBS entry form
&show_bbs;             # show BBS msgs
&show_read_form;       # show BBS read form
&show_footer;          # show footer
exit;

##### sub routines #####
sub read_file {
    open (IN, "$Bbs_file") || &return_error
        (500, "File open error", "Cannot access BBS file");
    @Bbs = <IN>;
    close (IN);
}

sub read_form {
    %Form = &read_input;

    $Name    = $Form{'name'};
    $Email   = $Form{'email'};
    $Msg     = $Form{'msg'};
    $Num_each = $Form{'num_each'};
    $Num_skip = $Form{'num_skip'};
}

sub remove_html_tags {
    $Name =~ s/\&/\&amp;/g;
    $Name =~ s/</\&lt;/g;
    $Name =~ s/>/\&gt;/g;

    $Email =~ s/\&/\&amp;/g;
    $Email =~ s/</\&lt;/g;
    $Email =~ s/>/\&gt;/g;

    $Msg =~ s/\&/\&amp;/g;
    $Msg =~ s/</\&lt;/g;
    $Msg =~ s/>/\&gt;/g;

    $Msg =~ s/\r\n/\n/g;      # Windows(CR,LF)    -> LF
    $Msg =~ s/\r/\n/g;       # Mac (CR)         -> LF
    $Msg =~ s/\n/<BR>/g;     # LF                 -> <BR>
}

sub new_msg {
    my ($access_time) = &clock; # get date and time
    my ($new_msg);
    my ($site) = ($ENV{'REMOTE_HOST'} || $ENV{'REMOTE_ADDR'});
}

```

```

# delete msg larger than 4000 byte
if (length($Msg) >= 4000) {
    $Msg = substr($Msg, 0, 4000);
    if (($Msg =~ tr/[\xA1-\xFE]//)%2 != 0) {
        chop $Msg;
    }
}

if ($Email !~ /([\w\.-]+\@([\w\.-]+\.[\w\.-]+))/) {
    $Email = ""; # check valid email address format
}

$new_msg = "$Name"      . $Delimiter . # name
           "$Email"     . $Delimiter . # email address
           "$site"      . $Delimiter . # ip_address
           "$access_time" . $Delimiter . # access time
           "$Msg"       . "\n";      # message

@Bbs = ($new_msg, @Bbs);
}

sub save_msg {
    (&filelock ($Lock_dir) eq 'OK') || &return_error
    ("500", "Lock error", "Too many access or cannot create lock file");

    open (OUT, ">$Bbs_file") || &return_error
    (500, "File write error", "Cannot access BBS file");
    print OUT @Bbs;
    close (OUT);

    (&fileunlock ($Lock_dir) eq 'OK') || &return_error
    ("500", "Unlock error", "No lock file exist");
}

sub show_header {
    print "Content-type: text/html\n\n";
    print <<END_OF_HTML;

<HTML>
<HEAD>
    <BASE HREF="$Base_href">
    <TITLE>$bbs_title</TITLE>
</HEAD>
<BODY BGCOLOR="$Bgcolor" TEXT="$Textcolor" LINK="$Linkcolor">
    <P ALIGN="CENTER"><FONT COLOR="$Titlecolor" SIZE="+2">
<B>$Bbs_title</B></FONT>
<br>
<div align="left"><A HREF="$Link_url">$Link_name</A></div>
<P ALIGN="RIGHT">
<HR>
END_OF_HTML
;
}

sub show_entry_form {
    print <<END_OF_HTML;
<a name="write"> </a>

<FONT COLOR="#800080" SIZE="+1">
<B>$Bbs_write_title</B></FONT>

```



```

<P>
<FORM ACTION="\$Cgi_file\" METHOD="\POST\">
  <B>Name</B> $Rem_name<BR>
<INPUT NAME="name" TYPE="text" SIZE = "40" MAXLENGTH="64">
<P>
<B>Email address</B> (optional) $Rem_email<BR>
<INPUT NAME="email" TYPE="text" SIZE = "40" MAXLENGTH="72">
<P>
<B>Message</B> $Rem_msg<BR>
<TEXTAREA COLS="64" ROWS="10" WRAP="VIRTUAL" NAME="msg">
</TEXTAREA><BR>
<INPUT TYPE="submit" VALUE="$Btn_write ">
</FORM>
<HR>
END_OF_HTML
;
}

sub show_read_form {
    print <<END_OF_HTML;
<font color="#800080" size="+1"><b>$Bbs_read_title</b></font>
<br><br>

<table border="0" cellspacing="10">
<tr valign="top">
  <td>
    <FORM ACTION="\$Cgi_file\" METHOD="\POST\">
      <INPUT TYPE="submit" VALUE="$Btn_read_old"><br>
    <INPUT TYPE="radio"
      NAME="num_each"
      VALUE="5" CHECKED>5 more
      <INPUT TYPE="radio"
        NAME="num_each"
        VALUE="20">20 more
      <INPUT TYPE="hidden"
        NAME="num_skip"
        VALUE="$Prev_num_skip">
    </FORM>
  </td>
  <td>
    <FORM ACTION="\$Cgi_file\" METHOD="\POST\">
      <INPUT TYPE="submit" VALUE="$Btn_read_new"><br>
      <INPUT TYPE="radio"
        NAME="num_each"
        VALUE="-5" CHECKED>5 more
      <INPUT TYPE="radio"
        NAME="num_each"
        VALUE="-20">20 more
      <INPUT TYPE="hidden"
        NAME="num_skip"
        VALUE="$Prev_num_skip">
    </FORM>
  </td>
</tr>
<tr></tr>
<tr></tr>
  <td>
    <FORM ACTION="\$Cgi_file\" METHOD="\POST\">
      <INPUT TYPE="hidden" NAME="num_each" VALUE="top">
      <INPUT TYPE="submit" VALUE="Go Top">
  </td>

```

```

        </FORM>
</td>
<td></td>
<td>
    <FORM ACTION="\$Cgi_file\" METHOD="\POST\">
    <INPUT TYPE="hidden" NAME="num_each" VALUE="all">
    <INPUT TYPE="submit" VALUE="Show All"><br>
    (Warning: very long!)
    </FORM>
</td>
</tr>
</table>
<hr>
END_OF_HTML
;
}

sub show_bbs {
    local( $bbs_no,
           $serial_no,
           @show_bbs,
           @data,
           $temp_email,
           $temp_msg );

    $bbs_no = $#Bbs + 1;

    unless (defined ($Num_each)) {$Num_each = 5;}
    unless (defined ($Num_skip)) {$Num_skip = 0;}

    if ($Num_each eq "all") {
        $Num_each = $bbs_no - $Num_skip;
    }
    if ($Num_each eq "top") {
        $Num_each = 5;
        $Num_skip = 0;
    }
    if ($Num_skip >= $bbs_no) {
        &show_end_of_bbs;
    }
    if ($Num_skip + $Num_each > $bbs_no) {
        $Num_each = $bbs_no - $Num_skip;
    }
    if ($Num_each < 0 ) {
        $Num_skip = $Num_skip + $Num_each + $Num_each;
        $Num_each = 0 - $Num_each;
        if ($Num_skip < $Num_each) {
            $Num_skip = 0;
        }
    } else {
        $Prev_num_skip = $Num_skip + $Num_each;
    }

    # skip data at next loading
    }

    $serial_no = $bbs_no - $Num_skip;
    # first serial No.

    @show_bbs = splice(@Bbs, $Num_skip, $Num_each);

```

```

# displayed msgs
    while (@show_bbs) {
        @data = split (/$Delimiter/, shift(@show_bbs));
# use slice of message file

        print "[ $serial_no ] <B>";
        print shift(@data); # $Name
        print '</B>';

        if ($temp_email = shift(@data)) { # $Email
            print "<<a href=\"mailto:$temp_email\">$temp_email</a>>";
        }
        print ' from ';
        print shift(@data); # $ip_address
        print ' at ';
        print shift(@data); # $access_time
        print ' <P>';
        $temp_msg = shift(@data);
        $temp_msg =~ s/(http:\\\\)([\\w+\\-\\/=\\?\\.\\~\\:\\&\\;\\#]+)/
            <a href=\"$1$2\" target=\"_new\">$1$2</a>/g;
        $temp_msg =~ s/([\\w\\-]+@[\\w\\-+\\.]+[\\w\\-]+)/
            <a href=\"mailto:$1\">$1</a>/g;
        print $temp_msg; # $msg
        print "\\n<HR>\\n";

        $serial_no --; # prepare next serial No.
    }
}

sub show_end_of_bbs {
    print "$Rem_bbs_end";
    print '<P>';
    print "<A HREF=\"$Cgi_file\">$Rem_bbs_back</A>";
    print "<HR>\\n";
}

sub show_footer {
    print <<END_OF_HTML;
<A HREF=\"$Link_url\">$Link_name</A>
<HR>
<I>BBS Master : $Bbsmaster_name
 / <A HREF=\"mailto:$Bbsmaster_email\">$Bbsmaster_email</A></I>

<div align="right">
<B><I><A HREF=\"$Bbs_logo_addr\">$Bbs_logo</A></I></B>
</div>

</body>
</html>
END_OF_HTML
;
}

#### other subroutines (shared with other perl cgi) #####
sub read_input {
    local ($buffer, @pairs, $name, $value, %FORM);
    if ($ENV{'REQUEST_METHOD'} =~ /^post$/i) {
        read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
    }
}

```

```

    } else {
        $buffer = $ENV{'QUERY_STRING'};
    }
    @pairs = split(/&/, $buffer);
    foreach (@pairs) {
        ($name, $value) = split(/=/, $_);
        $value =~ tr/+// /;
        $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
        $FORM{$name} = $value;
    }
    return %FORM;
}

sub clock {
    my ($date);

    @days = ('Sun','Mon','Tue','Wed','Thr','Fri','Sat');
    ($sec,$min,$hour,$mday,$mon,$year,$yday,$yday,$isdst) = localtime(time);
    $mon ++;
    if ($hour < 10) { $hour = "0".$hour; }
    if ($min < 10) { $min = "0".$min; }
    if ($sec < 10) { $sec = "0".$sec; }
    $date = "19$year\".$mon\".$mday\".($days[$yday]) $hour\":$min\":$sec";
}

sub return_error {
    my ($status, $keyword, $msg) = @_;
    print "Content-type: text/html\n";
    print "Status: ", $status, " ", $keyword, "\n\n";
    print "<TITLE>CGI program Error</TITLE>\n";
    print "<H1>", $keyword, "</H1>\n";
    print $msg, "\n";
    print "<P>Please contact the webmaster for more information.\n";
    print "<HR>";
    print "<A HREF=\"http://heartkorea.com/\"><I>Guest Book</I></A>";
    exit(1);
}

sub filelock {
    # parameter is filename for lock
    my ($lockdir) = @_;
    mkdir ($lockdir, 0777) && return 'OK';
    -M $lockdir > 3/(24*60) && do {
        rmdir ($lockdir) || return 'FAIL';
    };
    for (1 .. 10) {
        mkdir ($lockdir, 0777) && return 'OK';
        sleep (1);
    }
    return 'BUSY';
}

sub fileunlock {
    my ($lockdir) = @_;
    -d $lockdir || return 'FAIL';
    rmdir ($lockdir) && return 'OK';
    return 'FAIL';
}

#### end of program #####

```

B.3.3. Configuration and Checking the Results

The following are the steps for configuration and checking the results.

1. Modify the `sample_perl.m` environment file to correspond with the system
2. Get the Perl path by using the `$which perl` command.
3. Modify the `guestbook.cgi` file to correspond with the system.
4. Move `guestbook.cgi` to the directory where CGI is executed and change the directory permission to 755.

```
$chmod 755 guestbook.cgi
```

5. Create `book_data` to store `guestbook.cgi`. Change the file permission to 666.

```
$touch book_data  
$chmod 666 book_data
```

6. Create a lock directory and change the permission of the directory to 777.

```
$mkdir lock  
$chmod 777 lock
```

7. Start WebtoB and request the CGI file in the browser. (Use the IP address of the machine where WebtoB is currently running.)

```
http://IP address:port/cgi-bin/guestbook.cgi
```

8. After execution, the following screen is displayed.

Guest Book

[Back to Home](#)

WRITE

Name

Email address (optional)

Message

```
GUEST BOOK CGI testing ~~~~~  
It's working !!
```

[2] **guest 2** <b@bbb.com> from 143.248.150.192 at 19101.2.16.(Fri) 15:57:12

Hi~~~ your page is good.

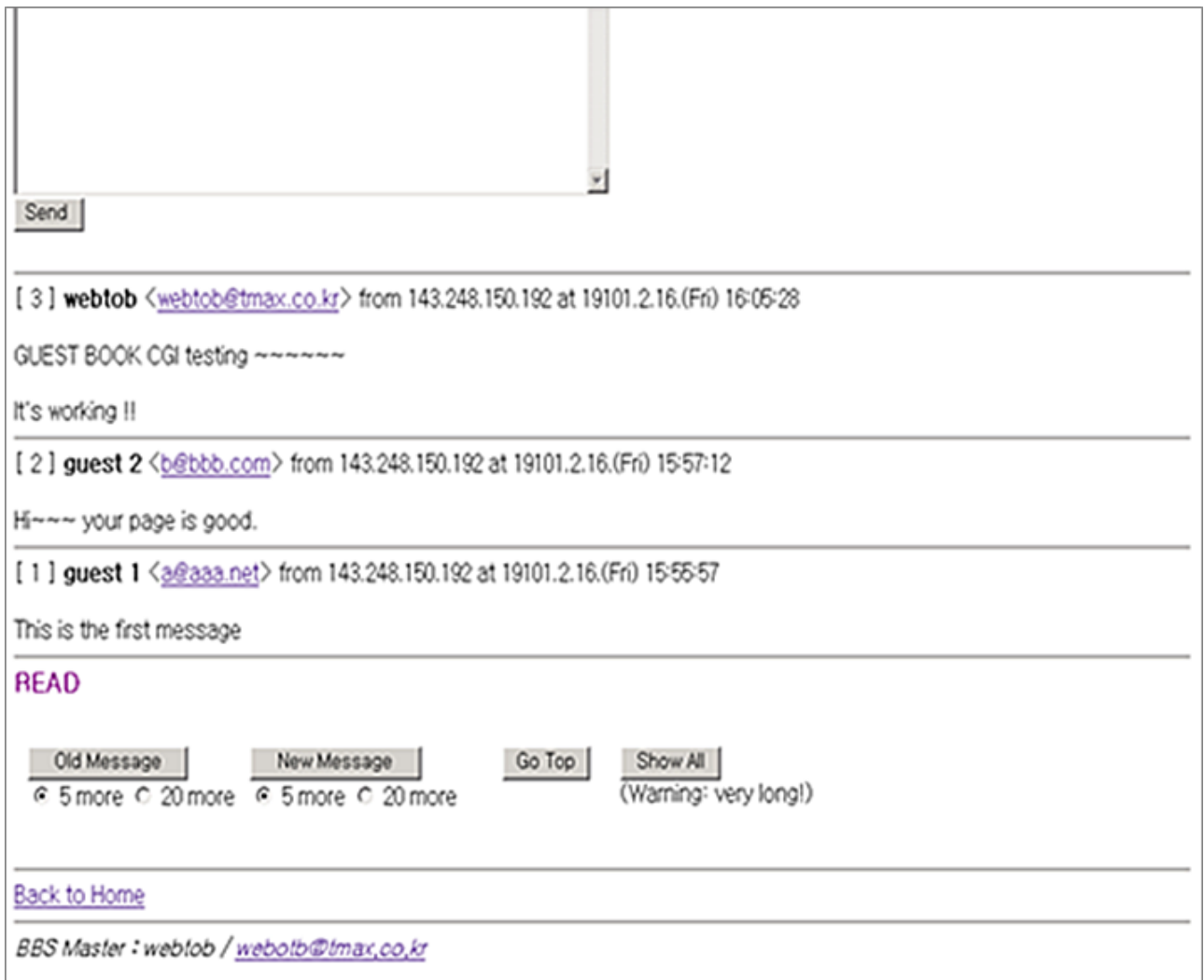
[1] **guest 1** <a@aaa.net> from 143.248.150.192 at 19101.2.16.(Fri) 15:55:57

This is the first message

Input Form of Perl based BBS

Fill out the form and click **[Send]** to write to the guest book. In the example, the message from 'webtob' will be posted above the message from 'guest 2'. Messages from visitors are left in the book_data file.

The following is the result screen.



Result Screen of Perl based BBS.

Appendix C: Integration with Tmax

This appendix describes how to create, compile, and execute a client program for WebtoB and Tmax integration.

C.1. Client Program for Integrating with Tmax

The following is the client program for apsl.m file described in [Tmax Integration Configuration](#).

This client program uses the TOUPPER service of svr2, a Tmax sample server program. TOUPPER is mainly composed of wbsession.c and wbquery.c. The path to tmax.env must be specified in wbquery.c.



Tmax must be installed separately. For more information on instructions to Tmax and its sample server, refer to the Tmax manual.

The following is wbsession.c.

<wbsession.c>

```
#include <stdio.h>
#include <string.h>
#include "../usrinc/wbapi.h"

wbsession(WBSVCINFO *rqst) {
    int len;
    char *value, *value2;
    char *name;
    SESSION *session;

    name = wbGetData( rqst, "name" );
    session= wbGetSession( rqst );
    session = wbSessionSetValue( session, "name", name, strlen(name) );
    value = wbSessionGetValue( session , "name", &len );

    wbSendRedirect( rqst, "/svct/query" );
    wbReturn( rqst, WBSUCCESS );
}
```

The following is wbquery.c.

<wbquery.c>

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include <usrinc/wbapi.h>

int flag=0;

wbSvrInit( int argc, char *argv[] ) {
```



```

fprintf( stdout, "query SERVER Start!\n" );

if( test_tpstart() == -1 ) {
    fprintf( stdout,"tpstart failed!!! %s \n",tpsterror(tperrno) );
}
return 0;
}

wbSvrDone() {
    fprintf( stdout, "query SVR DONE!\n" );
    tpend();
}

query(WBSVCINFO *rqst) {
    int len;
    char *name;
    SESSION *session;
    session= wbGetSession( rqst );

    name = wbSessionGetValue( session , "name", &len );
    wbPutHdr(rqst,"Content-type","text/html; charset=euc-kr");
    wbPrint( rqst, "<HTML><BODY>" );

    if( query != NULL ) {
        wbPrint(rqst, "<H1>send Data: %s <br></H1>", name);
    }
    else {
        wbPrint( rqst, "<H1>QUERY is null</H1>" );
    }

    if ( test_tpcall(name) == -1 ) {
        wbPrint( rqst, "<H1>tpcall failed : %s</H1>", tpsterror(tperrno) );
    }
    else {
        wbPrint( rqst, "<H1>recieved Data : %s</H1>",name );
    }

    wbPrint( rqst, "</body></html>" );
    wbReturn( rqst, WBSUCCESS );
}

int test_tpstart() {
    int rcode;

    rcode = tmaxreadenv( "/data1/gloria/webtob/ap/tmax.env", "TMAX" );

    if ( rcode == -1 ) {
        printf( "tmax readenv failed %s \n", tpsterror(tperrno));
        return -1;
    }
    rcode = tpstart( (TPSTART_T *)NULL );

    if ( rcode == -1 ){
        printf( "tpstart failed %s \n",tpsterror(tperrno) );
        return -1;
    }
    return 1;
}

```

```

int test_tpcall(char *name) {
    int rcode;
    char *sndbuf, *rcvbuf;
    long rcvlen, sndlen;

    if ((sndbuf = (char *)tpalloc("STRING", NULL, 0)) == NULL){
        printf( "sndbuf alloc failed ! %s \n", tpstrerror(tperrno) );
        tpend();
        return -1;
    }
    if ((rcvbuf = (char *)tpalloc("STRING", NULL, 0)) == NULL){
        printf( "rcvbuf alloc failed ! %s \n", tpstrerror(tperrno) );
        tpfree( (char *)sndbuf );
        tpend();
        return -1;
    }
    strcpy(sndbuf, name);

    if(tpcall("TOUPPER", sndbuf, 0, &rcvbuf, &rcvlen, 0)==-1){
        printf("Can't send request to service TOUPPER %s \n", tpstrerror(tperrno));
        tpfree( (char *)sndbuf );
        tpfree( (char *)rcvbuf );
        tpend();
        return -1;
    }
    printf( "[rcvbuf:%s]\n", rcvbuf );
    strcpy( name, rcvbuf );
    tpfree( (char *)sndbuf );
    tpfree( (char *)rcvbuf );
    return 1;
}

```

From the previous source code, modify tmaxreadenv to the path of the Tmax environment file.

The following is the tmax.env file.

<tmax.env>

```

[TMAX]
TMAXDIR=/data1/gloria/tmax
TMAX_HOST_ADDR=192.168.1.43
TMAX_HOST_PORT=7979
SDLFILE=/data1/gloria/tmax/sample/sdl/tmax.sdl
FDLFILE=/data1/gloria/tmax/sample/fdl/tmax.fdl
TMAX_CONNECT_TIMEOUT=2 [TMAX]
TMAXDIR=/data1/gloria/tmax
TMAX_HOST_ADDR=192.168.1.43
TMAX_HOST_PORT=7979
SDLFILE=/data1/gloria/tmax/sample/sdl/tmax.sdl
FDLFILE=/data1/gloria/tmax/sample/fdl/tmax.fdl
TMAX_CONNECT_TIMEOUT=2

```

C.2. Compiling Client Program

The following is a makefile used to compile client programs.

```
#Makefile.c
TARGET = $(COMP_TARGET)
APOBJS = $(TARGET).o

WEBTOB_INCDIR = $(WEBTOBDIR)/usrinc
WEBTOB_BINDIR = $(WEBTOBDIR)/bin
WEBTOB_LIBDIR = $(WEBTOBDIR)/lib

#####
# TMAX_LIBDIR #
#####
#32bit Tmax library
TMAX_LIBDIR = $(TMAXDIR)/lib

#64bit Tmax library
#TMAX_LIBDIR = $(TMAXDIR)/lib64

#SDLFILE = demo.s
#SDLDIR = $(WEBTOBDIR)/sdl

#####
# CFLAGS #
#####

#hp 32bit
CFLAGS = -Ae +DA1.1 +DD32 +DS2.0 -O -I$(WEBTOBDIR)
        -L/data1/gloria/tmax/lib

#hp 64bit
#CFLAGS = -Ae +DA2.0W +DD64 +DS2.0 -O -I$(WEBTOBDIR)
        -L/data1/gloria/tmax/lib

#sun 64bit
#CFLAGS = -xarch=v9 -O -I$(WEBTOBDIR)

#Linux
#CFLAGS = -O -I$(WEBTOBDIR)

#ibm 32bit
#CFLAGS = -q32 -O -I$(WEBTOBDIR) -brtl

#ibm 64bit
#CFLAGS = -q64 -O -I$(WEBTOBDIR) -bnoquiet -brtl

#####
# LIBS #
#####

#hp, sun, linux
LIBS = -laps -lcli

#ibm
#LIBS = -laps -lcli -lz
```

```

OBJS = $(APOBJS) $(SVCTOBJ) $(SDLOBJ)
SVCTOBJ = $(TARGET)_svctab.o
#SDLOBJ = ${SDLFILE:.s=_sdl.o}
#SDLC = ${SDLFILE:.s=_sdl.c}

.SUFFIXES : .v

.c.o:
$(CC) $(CFLAGS) -c $<

# Server
$(TARGET): $(APOBJS) $(SVCTOBJ)
$(CC) $(CFLAGS) -L$(WEBTOB_LIBDIR) -L$(TMAX_LIBDIR) -o $(TARGET) $(OBJS) $(LIBS)
@rm -f *.o
@rm -f *_svctab.c

$(SVCTOBJ):
cp $(WEBTOBDIR)/svct/$(TARGET)_svctab.c .
$(CC) $(CFLAGS) -I$(WEBTOB_INCDIR) -c $(TARGET)_svctab.c

#
clean:
-rm -f *.o core

```

The following shows how to use Makefile.c.

```

#!/bin/ksh
# program compile
#
#main
    Param=$1
    case "$Param" in
        c)      export COMP_TARGET=$2
                make -f Makefile.c;;
        api)    export COMP_TARGET=$2
                make -f Makefile.api;;
        pc)     export COMP_TARGET=$2
                make -f Makefile.pc all;;
        clean)  make -f Makefile.pc clean;;
        *)      echo "Usage: $0 argument";;
    esac

```

The following is an example of compilation.

```

$ compile c wbsession
$ compile c wbquery

```

C.3. Integrating with Tmax

Integrate Tmax by using the previous examples as follows:

1. Compile the WebtoB environment file.

```
$ wscfl -i apsl.m
```

2. Create a service table.

```
$ wsgst
```

3. Compile the client program.

```
$ compile c wbsession  
$ compile c wbquery
```

4. Compile the Tmax server program.

```
$ compile c svr2
```

5. Boot Tmax.

```
$ tmbot
```

6. Boot WebtoB.

```
$ wsboot
```

Use an HTML page, such as toupper.html, to view the execution result.

```
<html>  
<head>  
  <title>wbGetQueryString</title>  
</head>  
<body>  
<form method=post action="/svct/wbsession">  
<table width=370>  
<br>  
<tr>  
  <td> input send Data</td>  
  <td><input type=input name=name></td>  
  <td><input type=submit value="submit"></td>  
</tr>  
</table>  
</form>  
</body>  
</html>
```

input send Data

toupper.html

send Data : aaaa
recieved Data : AAAA

TOUPPER Result Screen