
STON Documentation

Release 2.3.4

WineSOFT Inc.

February 01, 2017

1	0. Overview	3
2	1. Basics	15
3	2. HTTP Caching	35
4	3. Advanced Features	77
5	4. Management & Operation	155
6	Appendix	201

Author WineSOFT Inc.

Version 2.4.3, Jan 20, 2017

0. Overview

1.1 Chapter 1. STON Edge Server

1.1.1 Principles of Service Design

The success of a service depends on availability, speed, and scalability. Kate Matsudaira, author of “Building Scalable Web Architecture and Distributed Systems”, also emphasized these three principles.

Availability

A service must always be available. In the event of failure, ninety percent of users will move on to competitors. While there is no such thing as a perfect system, recovering from failures must be quick.

Speed

In business, time is money, and in e-commerce, high latency will lead to a drop in sales. For every 0.1 second of latency, there is a one percent decrease in revenue. 47 percent of Amazon.com customers want a website loaded on their screen within two seconds.

Scalability

Regardless of the number of users, the service must always be reliable. Scalability is the effort required to increase the system’s capacity to handle more load, and can also refer to how easy it is to add more storage or how many more transactions can be processed. Scalability in maintenance is also important: it should be easy to diagnose and understand problems and implement updates or fixes.

The service is most effective when all the principles can be upheld at the smallest possible cost. Cost is not limited to just money, but also includes time, effort, and training.

A successful service must **grow**, and when it does, it must be able to manage more clients and more content. However, as the system grows, it becomes harder to uphold these principles. What can be done to uphold these principles at the smallest possible cost?

1.1.2 The Growth of the Service

A test or pilot service generally begins with one or two servers, and when it begins to grow, the number of servers will increase accordingly. Content renewal must be meticulously carried out one server at a time. It may be a laborious task, but it is not an impossible one as of yet.

As the service begins to expand with more users and more accumulated data, managing each server one by one becomes even more difficult. At this point, high-cost storage (e.g. NAS, SAN, DAS) is introduced to collect all the data in one system. Expensive but reliable storage systems make content renewal easier, because servers can automatically acquire updated content from storage.

However, as the service sharply grows, the number of servers increases. More servers mean more data from storage, leading to data transfer overload. A new storage system that supports higher bandwidth can be even more expensive, so it's questionable if one would want to invest in it.

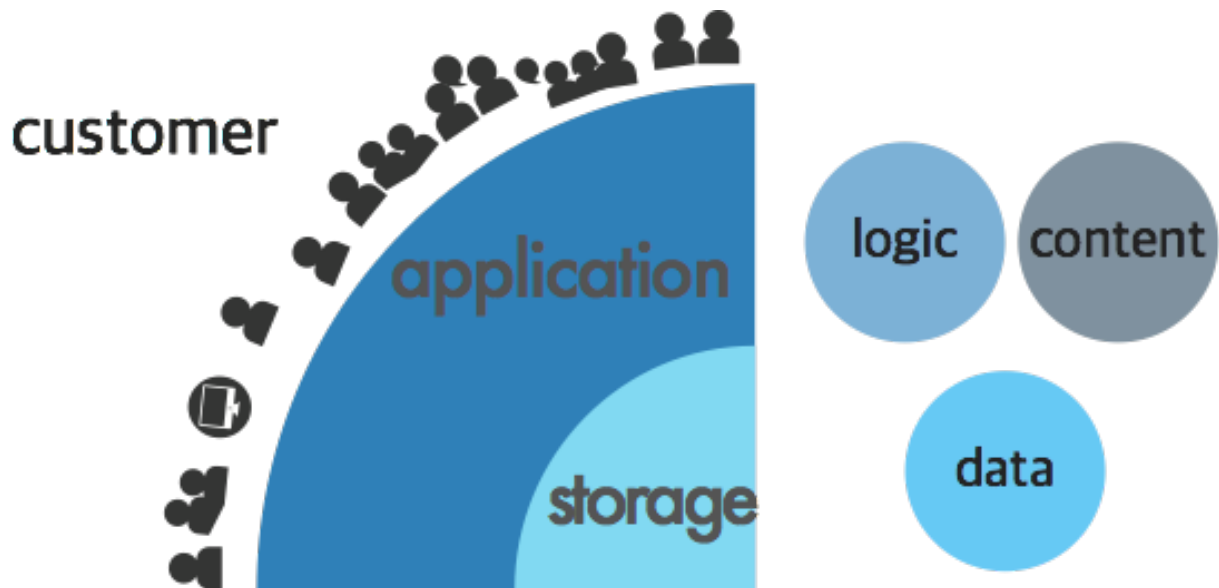
A potential solution to the problem is synchronization. Getting all the data ready is impractical, so the storage system must be able to sort the content. Management is essential to achieve precise content control. Synchronization across a few servers may be easy, but the more servers and files to sync, the harder it becomes. Synchronization can become slower, harder, and even more unstable as the system expands.

Moreover, content is constantly changing. The more files there are to add or delete, the longer the synchronization time becomes. A larger-scale service inevitably requires a bigger and more complicated synchronization managing system. Any problems in the management system may eventually lead to total service failure.

A simpler method to quickly and flexibly deliver content to servers is necessary.

1.1.3 Service Scalability and Content Delivery

As shown in the figure below, a service can be broken down into two layers.

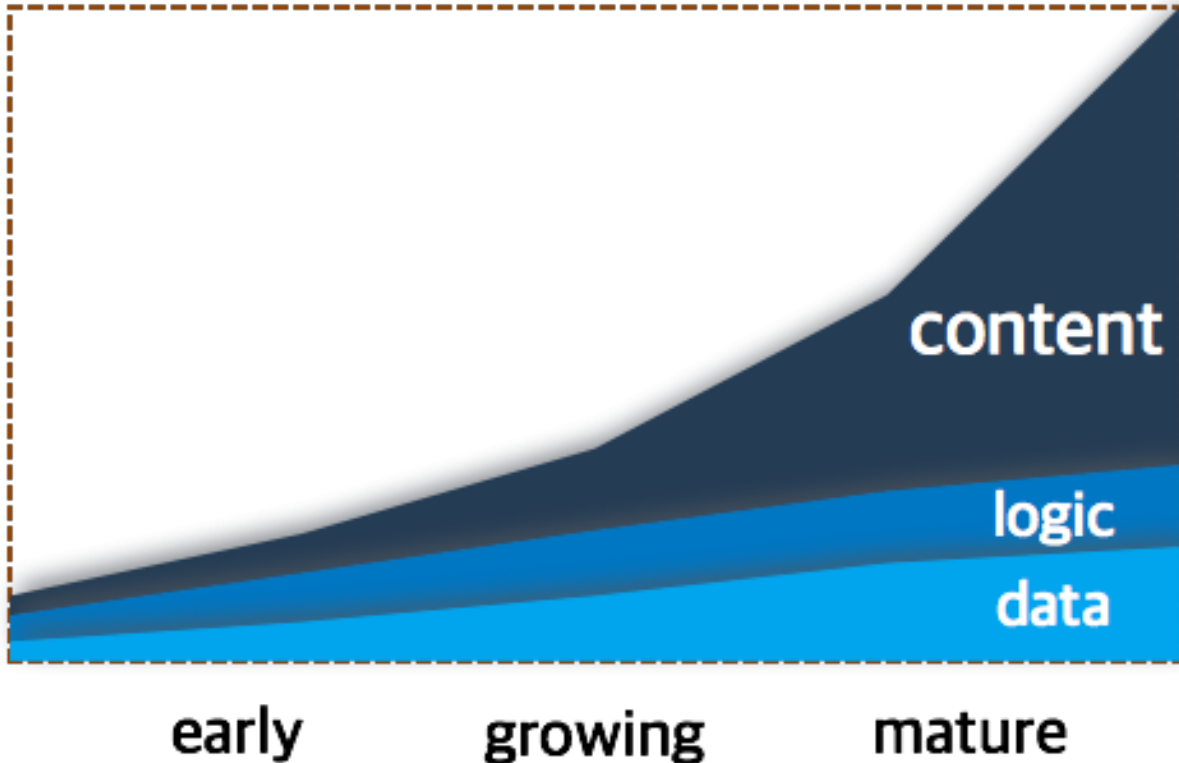


In the center is the storage layer, which manages data. Above it is the application layer, which implements the service logic and can also process content delivery for a small number of customers. In the beginning stages, the service can be set up with only the storage and application layers.

As the service grows, the total cost will change. In the beginning stages, the biggest expense is in logic development, while in the growth stages, the biggest expense is in data management in accordance with the number of users. However, as the service matures, the main concern becomes **content delivery**, making it a huge hurdle for service expansion. How can the exploding bandwidth be taken care of?

1.1.4 The Edge: The Delivery Layer

When the service reaches maturity, the burden of content delivery will increase exponentially. Shopping mall content numbers in the billions, and the video service content has long since begun to use terabytes. To expand a service, the **scalability of content delivery** must definitely be taken into account.



The edge layer is the outermost layer of the service in which clients will be able to experience speed and availability. No matter the circumstances, the content requested by the users must always be delivered to them. Broken images or unavailable pages on the user's screen is fatal to the reputation of the service. By handling content delivery via the edge, there is less of a burden on the application and storage layers.

Having an easily and efficiently expandable edge layer removes the need to expand other high-cost layers. Meanwhile, expanding the storage layer or the application layer is a poor choice due to its high cost and low efficiency.

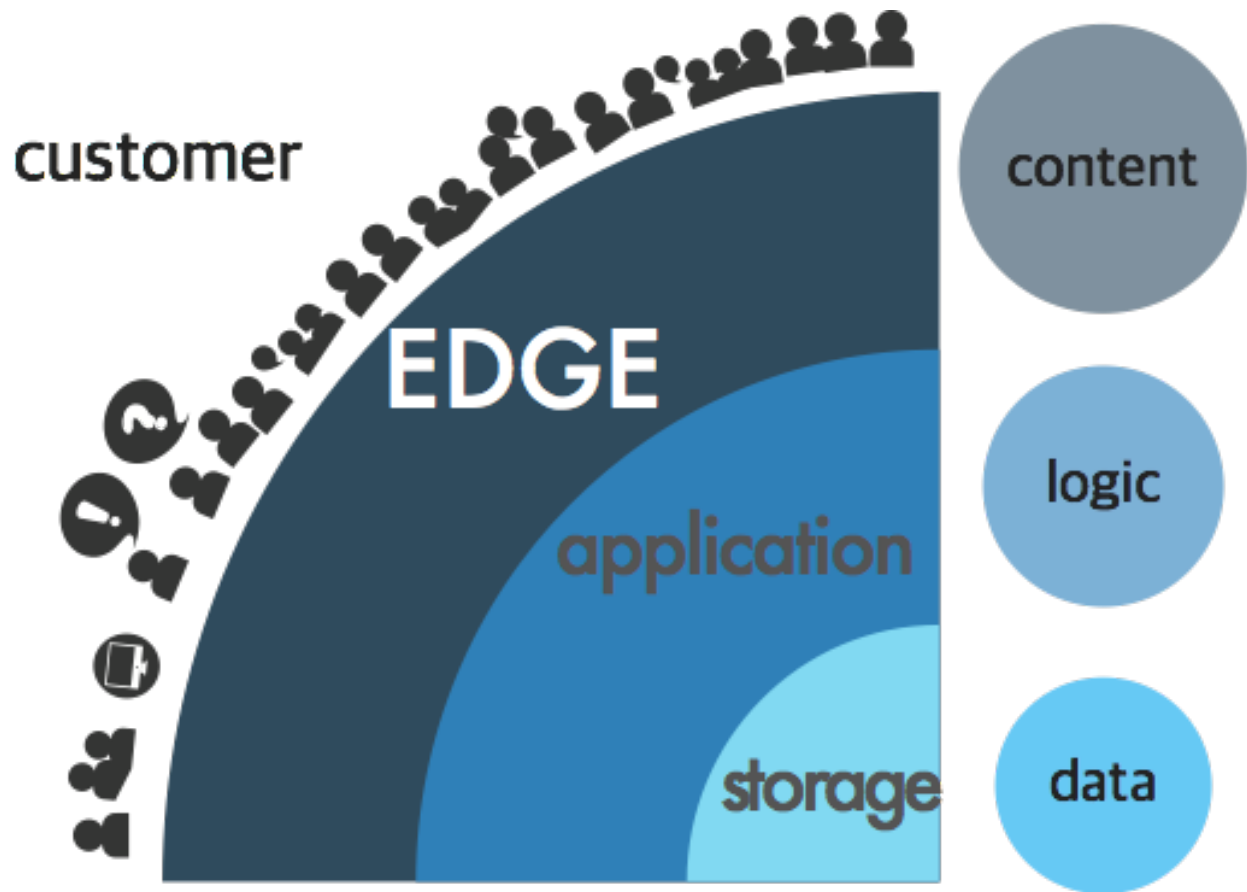
In that case, how does the STON Edge Server make content delivery quicker and easier?

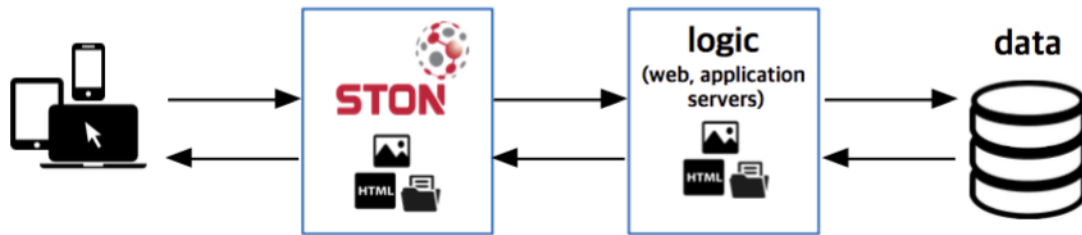
1.1.5 The Behavior of the Edge Server: Caching

The scale of data delivery is proportional to the number of users and the size of content. The service can detect how many users are requesting what content most quickly within the edge layer. Because the bottom-up workflow from edge layer is the most efficient, the edge server implements **caching** behavior that responds on demand to the users' requests, without any need for management systems. The procedure is as follows.

When the edge server receives its first content delivery request, it obtains the content from the storage layer and then transfers the content to the user. This transferred content is also saved in the edge server itself. On future requests, the saved content can immediately be delivered to the users from the edge server itself. The saved content will only be available during the preset Time To Live (TTL) period.

In this way, the edge server can handle a considerable amount of content. It can allow for quick mass distribution of data while minimizing the need to expand the application and storage layers. Any growing service should take the edge server into consideration.



the first request**the second request and on (with a valid TTL)**

The STON Edge Server is software that aims to provide an unrestricted and unconditional environment. The server is designed to provide maximum performance on any type of hardware platform.

CPU: Optimized for multi-core processors. Throughput is proportional to the number of CPU cores.

Memory: Larger memory allows for faster processing and cuts down on Disk I/O.

Disk: I/O is evenly distributed to cache more data.

NIC: Guaranteed bandwidth of either 4 Gbps NIC Bonding or 10 Gbps NIC.

The STON Edge Server supports **powerful live monitoring and logging**. The administrator can check the current service status in real time with statistics updated every second. The real-time statistics are offered in universal formats such as JSON, XML, and SNMP.

STON offers **simple installation** for the sake of administrators, because STON's design principle is to be an edge server made with administrators in mind. An intuitive installation method is provided via the Web Management page. More detailed settings can be configured by editing only two XML files.

1.1.6 Benefits of the Edge Server

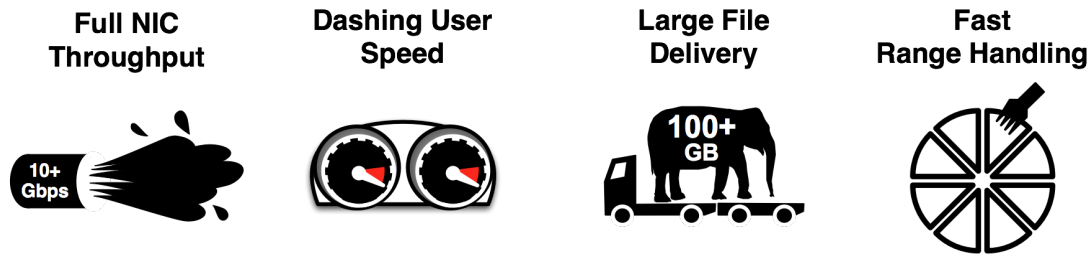
The benefits of the edge server are listed below.

- Provides simple and convenient service acceleration
- Shields the service origin from external access (Origin Shielding)
- Allows the other layers to concentrate on their fundamental roles

The advantages of adopting the edge server can be seen in the following application examples.

Gaming

Gaming services require a large amount of bandwidth. There are a variety of categories in gaming, from “masterpiece” games to casual games. Smartphone games have become especially popular, further diversifying the forms of services.



- **High Bandwidth Throughput**

A universal method to acquire high bandwidth with a single server is bonding 1 Gbps NIC (Network Interface Controller). With this, up to 4 Gbps can be achieved. Recently, 10Gbps NICs have also become common.

STON guarantees full bandwidth for both 4 Gbps NIC Bonding and 10 Gbps NIC.

- **Max User Bandwidth Guaranteed**

Everyone wants to play games as soon as possible, so they will want to download their games as fast as possible. Users with fiber optic LAN may complain if their speed falls under 100 Mbps. As long as a server's bandwidth isn't physically exceeded, it must be able to guarantee maximum speed equally to every user.

STON guarantees maximum transmission speed to all users.

- **Processing Large Files**

Nowadays, a game with a file size of about 4 GB can't even be considered a large game; there exist games with dozens of GB in file size. If the files are too large to cache in the server memory, critical service failure is likely. The worst-case scenario is when every user is each downloading a different part of a massive file from the server.

STON caching has no limit to file size, and will always guarantee high performance by swapping between memory and disk when appropriate.

- **Processing Range Requests**

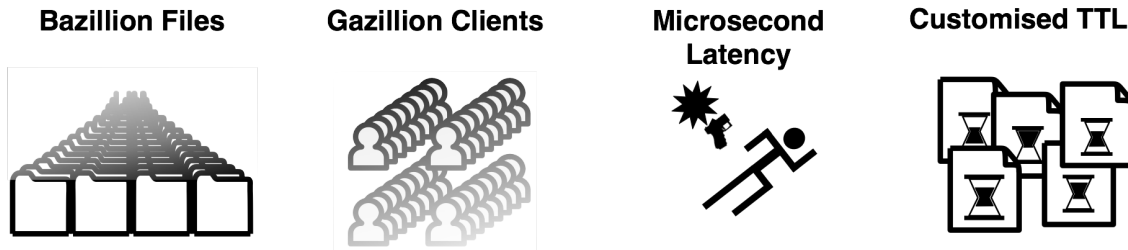
As files to deliver are growing larger, the P2P solution, based on the grid delivery method, has become widely used. This solution shreds a single file into small pieces to send or receive, thus making a huge number of HTTP range requests from the server. Theoretically, ten thousand clients can all request different ranges from a 10 GB file. The service must be able to respond immediately, regardless of what is being requested. However, the size of the transferred data must not exceed the size of the original file.

STON is loaded with a file system that is optimized for range requests. STON also guarantees faster responses via multi-download. It will not download a single unnecessary byte from the origin server.

E-commerce

In the case of online shopping malls, accessibility of the website is directly related to the amount of total sales. Recently, mobile shopping via smartphones has become just as common as the traditional PC-based online shopping.

A service will face difficulty if it cannot handle not just various shopping environments but also an infinitely growing number of files.



- **Numerous Small Files**

An expensive storage system is necessary when it comes to storing files that seem like they're constantly increasing. However, because being economical is important to the edge server, this solution is not preferred. There could be a service that consists of a billion 1 KB files, and caching all of them is not possible. Therefore, a method that keeps hold of frequently requested files while minimizing load on the origin server is necessary.

STON uses available memory and disk resources to their greatest capacity for caching. It manages the access frequency of all files in real time and removes older files based on the LRU (Least Recently Used) algorithm.

- **Millions of Users**

An online shopping mall must be able to handle the requests of multiple users at once. There are times when bursts of website traffic can occur due to a sudden event. Servers must be able to withstand these bursts and remain stable after them.

STON guarantees CPU scalability, with performance proportional to the number of resources. It can guarantee stability even during bursts using flexible HTTP keep-alive and socket handling.

- **Responsiveness**

Pleasant online shopping experiences occur when pages load quickly and the customers don't have to wait. If a page doesn't load within three seconds, the user will most often move to a different site. Even though the main page is generally made up of about a hundred files, it must still load perfectly in a second.

STON guarantees swift responses through real-time file indexing. Responsiveness is maximized by having seamless file replacement with no dependence on the origin server. Logs and statistics are offered for all HTTP responses (Time to First Byte, transaction completions) to detect declines in performance in real time.

- **Page TTL**

The majority of users follow a route that goes from the main page, to the upper category page, to the lower category page, and then to the product details page. Each page must be different not just in their exposure frequencies but also in their refresh cycles. A smart method of caching and refreshing is necessary.

STON can allocate separate TTLs to each URL. It also offers various refreshing methods such as Purge, Expire, ExpireAfter, and HardPurge to be used to fit the given situation.

Media

Exclusive media protocols are starting to lose their place, while the simple but powerful combination of HTTP and MP4 is gaining influence. Taking into consideration the variable connection statuses of mobile devices, HTTP-based streaming is likely to become the norm.

HTTP Live Streaming



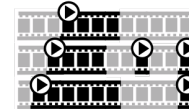
Pseudo-streaming



Bandwidth Control



Range Playback



- **Media Recognition**

Media files should no longer be seen as just one huge chunk of file. Bandwidth can be reduced and various additional functions can be linked only when the format of the media file is correctly recognized. If the server requires the entire file to determine its format, then during the time the server takes to acquire the file, the user will most likely quit waiting.

STON supports the MP4, MP3, M4A, and FLV formats. As soon as the server starts downloading a media file, it prioritizes the sections required for HTTP pseudo-streaming.

- **Media Header Reordering**

If the header is located at the end of the file, HTTP pseudo-streaming is unavailable. An exclusive media player would be necessary for these types of files, but this can make users easily frustrated.

If the header is placed at the end after encoding an MP4 file, an additional action to move it to the front is necessary. STON provides the service of automatically moving the header to the front.

- **Adjustable Bandwidth**

Not many users watch the entire video clip to the end. Therefore, an efficient streaming method would be to use only smallest amount of bandwidth necessary for smooth playback. Though the video may be the same, it can be watched in varying bitrates ranging from 360p to 1080p.

STON uses bandwidth throttling to optimize the bandwidth used during media file delivery.

- **Multi/Single Part Trimming**

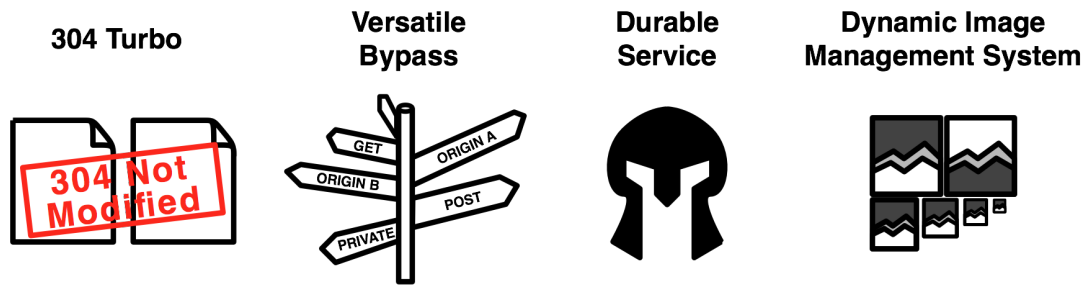
Some preview/highlight/sharing services provide only a specific part of the file instead of the whole. It would be a waste of time and storage space to extract parts for every file. Furthermore, there are cases where the extracted part may be different for each user. Some media players also implement a skip function for segment playback.

STON can trim a media file to extract parts that can be used as complete files themselves.

News / Forums

There are many points of interest for sites that have secured a large loyal user base. As these websites are where people of similar interests gather, users will stay on pages for long periods of time and exchanges will occur with vigor. The

service patterns of these sites vary by their subjects and it is tricky to meet their service requirements.



- **304 Not Modified**

Because these users are loyal to the website, most files will already be cached in local storage. Therefore, checking for updates will be more frequent than actual file transfer.

STON ensures that frequently accessed files are always kept in memory. Checking for updates can be processed immediately without waiting.

- **Bypass**

There are a certain category of pages that cannot be cached, such as user-specific pages or pages with new posts or replies. Even in these cases, a single domain is usually delegated to a reverse proxy instead of separating it into multiple domains.

STON elaborately classifies bypass targets based on various conditions. The server also maintains login sessions using the Origin Affinity and Private functions.

- **Origin Shield**

Websites owned by individuals or small or mid-size businesses cannot afford expensive equipment, infrastructure, or labor force. Server failure can occur relatively frequently, and it is often uneconomical to try and improve server quality.

STON will detect server overload or failure and automatically execute exclusion/recovery of the origin server. It will also extend TTL upon server failure and minimize dependence on the origin server.

- **Image Processing**

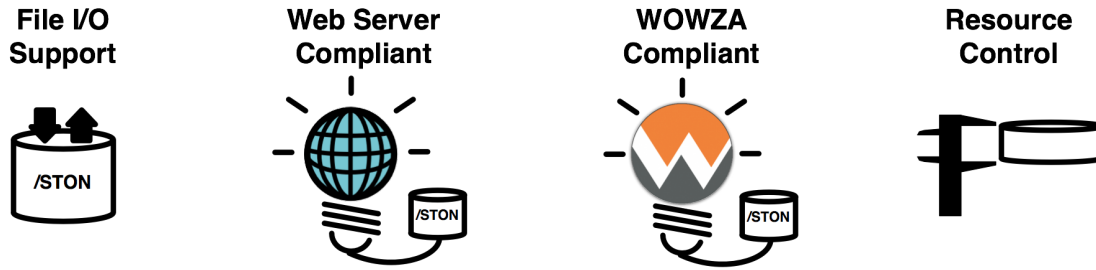
The same image may need to be displayed in different ways depending on the user environment. Search results may display images as thumbnails, while news sites may watermark their images. Processing every image into a specific format is a waste of time, storage, and effort.

STON's *DIMS* function can generate desired image formats from a single image using only URL calls.

File-based Server

The edge server is based on the reverse proxy structure. The fundamental concept of the reverse proxy is to copy/modify/manage files from the remote server to local storage. If STON can integrate with a service's server, it can resolve both storage centralization and synchronization issues. This will also decrease service development time and improve service reliability, killing two birds with one stone.

- **File I/O Support**



If an exclusive protocol is required, the server becomes subordinate to the corresponding module. Even the module was integrated with the server, if performance falls, it is dead weight. The stage between the module and the server must be reduced to a minimum.

STON can adopt standard file I/O. Only a Linux Kernel (VFS) is placed between STON and the exclusive server to guarantee high performance.

- **Web Server Integration**

Standard reverse proxies may be hard to implement if any special expansion modules are installed on standard web servers (Apache, Lighttpd, NginX). For example, it is hard for a file service or a payment service linked with DB/WAS to expand.

If Apache's DocumentRoot is assigned to STON, Apache will recognize STON as a physical disk and nothing will need to be configured.

- **Wowza Integration**

Wowza is considered to be a standard in the media service field. However, Wowza's HTTP caching function is not just inconvenient but also limited. In addition, other "exclusive"; protocols besides HTTP are fading away from the market.

STON can be mounted as a local disk. Moreover, all functions, such as MP4 header conversion and trimming, are available.

- **Resource Management**

A server that acquires back-end files and delivers them to front-end users will always have problems with file synchronization. Exclusive servers such as game or SNS servers have always had these issues during development. Because these servers must stay running for long periods of time without interruption, memory and disk use must be strictly controlled.

STON can easily control memory and disk use. Even when STON is mounted on a disk, all other functions will work in the same manner, so complicated services can be configured with a minimal solution.

The following Korean services are actively making use of the above attributes to grow with STON.



1. Basics

2.1 Chapter 2. Getting Started

This chapter will cover the installation and configuration of the system as well as how to set up a sample virtual host. A simple text editor is all that is necessary.

STON was developed to run on standard Linux servers, and minimize dependence on the hardware, OS, or file system, among others. However, it is still important to choose the most suitable equipment for the job, in order to set up the server to match the service's individual qualities.

2.1.1 Setting Up the Server

Generally, setting up a server means considering the CPU, memory, and disk. For example, if a service requires high performance on the level of 10 Gbps throughput, then each component must meet the requirements to reach the desired performance.

- **CPU** A CPU with at least four cores (quad-core) is recommended. Because STON gains scalability when it comes to multi-core CPUs, the per-second throughput increases with more cores. However, higher throughput does not necessarily mean higher traffic.

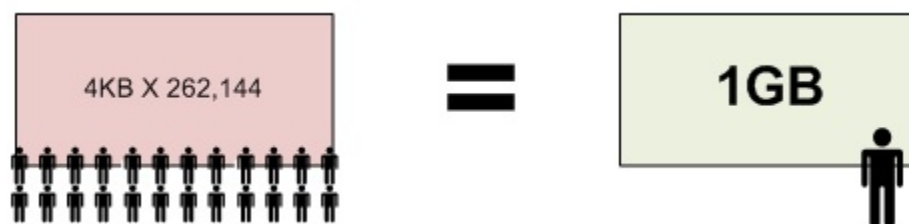


Fig. 2.1: The more clients there are, the more useful it is to have many CPUs.

Transferring a 4 KB file around 260,000 times takes the same amount of bandwidth as transferring a 1 GB file once. The most important criterion in choosing a CPU is the number of parallel connections the service must make.

- **Memory** At least 4GB of memory is recommended to be used in memory indexing (see also *Memory Structure*). Content that is frequently accessed will always be stored in memory, but content that is not will have to be loaded from disk. As such, if there is a lot of content with a large spread (a long-tailed distribution), then the load on the disk will be higher and performance may decline. If disk I/O load is high because of the size of the content, regardless of the amount of content, then memory can simply be added to decrease the load.

- **Disk** At least three disks, including the OS, is recommended. As one would expect, more disks lead to better performance, as I/O load will be dispersed and more content can be cached.

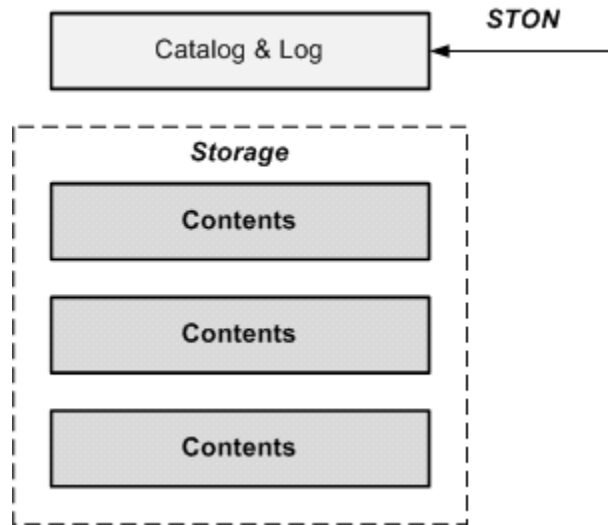


Fig. 2.2: The OS and STON will always be installed on a disk separate from the content.

In general, STON will be installed on the same disk as the OS. The log is also generally installed on the same disk. Because the log must record the service status in real time, it will always create write load.

The STON disks will be used in the RAID 0 setting. The correlation between performance and RAID changes depending on the client service's qualities. However, if file changes are infrequent and content size is much larger than physical memory, increasing read speed with RAID may be effective.

2.1.2 Setting Up the OS

In its most basic form, STON will operate normally on standard 64-bit Linux distributions (CentOS 6.2 or higher, Ubuntu 10.04 or higher), and does not require any particular package.

2.1.3 Installation

1. Download the latest version of STON.

```

[root@localhost ~]# wget http://foobar.com/ston/ston.2.0.0.rhel.2.6.32.x64.tar.gz
--2014-06-17 13:29:14-- http://foobar.com/ston/ston.2.0.0.rhel.2.6.32.x64.tar.gz
Resolving foobar.com... 192.168.0.14
Connecting to foobar.com|192.168.0.14|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 71340645 (68M) [application/x-gzip]
Saving to: "ston.2.0.0.rhel.2.6.32.x64.tar.gz"

100%[=====>] 71,340,645 42.9M/s in 1.6s

2014-06-17 13:29:15 (42.9 MB/s) - "ston.2.0.0.rhel.2.6.32.x64.tar.gz" saved [71340645/71340645]

```

2. Extract the downloaded package.

```
[root@localhost ~]# tar -zxf ston.2.0.0.rhel.2.6.32.x64.tar.gz
```

3. Run the installation script.

```
[root@localhost ~]# ./ston.2.0.0.rhel.2.6.32.x64.sh
```

4. All installation processes are recorded in the install.log file. Any problems that occur during the installation process will be noted there.

```
#DownloadURL: http://foobar.com/ston/ston.2.0.0.rhel.2.6.32.x64.tar.gz
#DownloadTime: 13 sec
#Target: STON 2.0.0
#Date: 2014.03.03 16:48:35
Prepare for STON 2.0.0 install process
    Stopping STON...
    STON stopped
[Copying files]
    './fuse.conf' -> '/etc/fuse.conf'
    './libfuse.so.2' -> '/usr/local/ston/libfuse.so.2'
    './libtbbmalloc_proxy.so' -> '/usr/local/ston/libtbbmalloc_proxy.so'
    './start-stop-daemon' -> '/usr/sbin/start-stop-daemon'
    './libtbbmalloc_proxy.so.2' -> '/usr/local/ston/libtbbmalloc_proxy.so.2'
    './libtbbmalloc.so' -> '/usr/local/ston/libtbbmalloc.so'
    './libtbbmalloc.so.2' -> '/usr/local/ston/libtbbmalloc.so.2'
    './libtbb.so' -> '/usr/local/ston/libtbb.so'
    './libtbb.so.2' -> '/usr/local/ston/libtbb.so.2'
    './stond' -> '/usr/local/ston/stond'
    './stonx' -> '/usr/local/ston/stonx'
    './stonr' -> '/usr/local/ston/stonr'
    './stonu' -> '/usr/local/ston/stonu'
    './stonapi' -> '/usr/local/ston/stonapi'
    './server.xml.default' -> '/usr/local/ston/server.xml.default'
    './vhosts.xml.default' -> '/usr/local/ston/vhosts.xml.default'
    './ston_format.sh' -> '/usr/local/ston/ston_format.sh'
    './ston_diskinfo.sh' -> '/usr/local/ston/ston_diskinfo.sh'
    './wm.sh' -> '/usr/local/ston/wm.sh'
[Exporting config files]
    #Export so directory
    /usr/local/ston/ to ld.so.conf
    #Export sysctl to /etc/sysctl.conf
    vm.swappiness=0
    vm.min_free_kbytes=524288
    #Export sudoers for WM
    Defaults    !requiretty
    winesoft ALL=NOPASSWD: /etc/init.d/ston stop, /etc/init.d/ston start, /bin/ps -ef
[Configuring STON daemon script]
    STON daemon activate in run-level 2345.
[Installing sub-packages]
    curl installed.
    libjpeg installed.
    libgomp installed.
    rrdtool installed.
[Installing WM]
    Stopping WM...
    WM stopped
    './wm.server_default.xml' -> '/usr/local/ston/wm/tmp/conf/server_default.xml'
    './wm.vhost_default.xml' -> '/usr/local/ston/wm/tmp/conf/vhost_default.xml'
    WM configuration found. Current WM port : 8500
    PHP module for Legacy(CentOS 5.5) installed
```

```
`./libphp5.so.5.5' -> `/usr/local/ston/wm/modules/libphp5.so'
WM installation almost complete. Changing WM privileges.
Installation successfully complete
```

2.1.4 Obtaining a License

New clients can obtain a license via this process.

1. Fill out the [application form](#).
2. Email the completed form to license@winesoft.co.kr.
3. The license will be issued after confirmation.

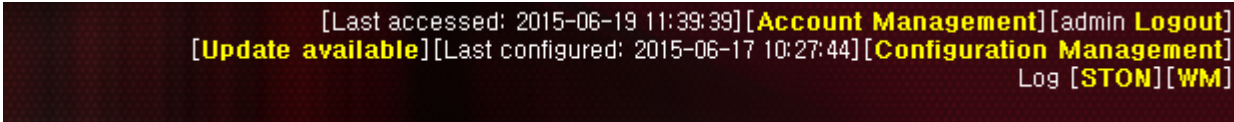
The license file (license.xml) must be in the installation directory for STON to run properly.

2.1.5 Update

Use the “stonu” command to update STON to the latest version.

```
./stonu 2.0.1
```

See also *Update to Latest Version* from *Chapter 13. WM (Web Management)* to easily update STON.



```
[Last accessed: 2015-06-19 11:39:39][Account Management][admin Logout]
[Update available][Last configured: 2015-06-17 10:27:44][Configuration Management]
Log [STON][WM]
```

2.1.6 Run

STON will be installed in the following default directory.

```
/usr/local/ston/
```

If any of the following files is missing or has invalid XML syntax, STON will not run.

- license.xml
- server.xml
- vhosts.xml

After the initial installation, not every XML file will be present. In this case, the distributed license.xml file should be placed in the directory. The server.xml.default and vhosts.xml.default files should also be placed in the directory. The *.default files are always included in the latest package.

2.1.7 Hello World

Open vhosts.xml and edit with the following code.

```
<Vhosts>
  <Vhost Name="www.example.com">
    <Origin>
      <Address>hello.winesoft.co.kr</Address>
    </Origin>
  </Vhost>
</Vhosts>
```

Running STON

1. Copy license.xml to the installation directory.
2. Open server.xml and configure <Storage>.

```
<Server>
  <Cache>
    <Storage>
      <Disk>/cache1</Disk>
      <Disk>/cache2</Disk>
    </Storage>
  </Cache>
</Server>
```

Note: STON normally uses disk storage. A disk must be configured in order to run STON. Disk set-up details can be found in the next chapter.

3. Run STON.

```
[root@localhost ~]# service ston start
```

To stop STON, use the “stop” command.

```
[root@localhost ~]# service ston stop
```

Checking the Virtual Host

(For Windows 7) Add www.example.com in the C:\Windows\System32\drivers\etc\hosts file as shown below.

```
192.168.0.100      www.example.com
```

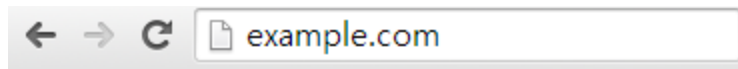
If all settings are correctly configured, the following page will be displayed on the browser when www.example.com is accessed.

If WM is Slow or the Graph Isn't Displayed

RRDtool is dynamically downloaded and installed during installation, and may not be installed properly under a restricted network. Moreover, [Chapter 13. WM \(Web Management\)](#) may run very slowly or [Appendix A: Graph](#) may not work at all. To fix these issues, follow the steps below.

1. Check Installation Status

Checking the installation status of RRDtool can be done as follows.



ston Edge Delivery Server



설치가 성공적으로 완료되었습니다.



WINE SOFT
2014 copyright reserved.

```
[root@localhost ston]# yum install rrdtool
Loaded plugins: fastestmirror, security
Loading mirror speeds from cached hostfile
* base: centos.mirror.cdnetworks.com
* elrepo: ftp.ne.jp
* epel: mirror.premi.st
* extras: centos.mirror.cdnetworks.com
* updates: centos.mirror.cdnetworks.com
Setting up Install Process
Package rrdtool-1.3.8-6.el6.x86_64 already installed and latest version
Nothing to do
```

(For Ubuntu)

```
root@ubuntu:~# apt-get install rrdtool
Reading package lists... Done
Building dependency tree
Reading state information... Done
rrdtool is already the newest version.
The following packages were automatically installed and are no longer required:
  libgraphicsmagick3 libgraphicsmagick++3 libgraphicsmagick1-dev libgraphics-magick-perl libgrap
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 102 not upgraded.
```

2. RRD Manual Install

If yum is unable to install RRDtool, the administrator should [download](#) the right package for the OS and proceed with manual installation.

Name	Last Modified	Size	Description
tcl-rdtool-1.4.7-1.el5.rf.i386.rpm	06-Apr-2012 16:57	36K	RHEL5 and CentOS-5 x86 32bit
tcl-rdtool-1.4.7-1.el5.rf.x86_64.rpm	06-Apr-2012 16:57	37K	RHEL5 and CentOS-5 x86 64bit
tcl-rdtool-1.4.7-1.el6.rf.x86_64.rpm	06-Apr-2012 16:57	35K	RHEL6 and CentOS-6 x86 32bit
tcl-rdtool-1.4.7-1.el6.rf.x86_64.rpm	06-Apr-2012 16:57	35K	RHEL6 and CentOS-6 x86 64bit

2.1.8 Origin Server

The purpose of the virtual host is to provide content in place of the origin server. Various types of origin servers can be accessed in various ways, depending on what is most suitable for the service platform.

```
<Vhosts>
  <Vhost Name="www.example.com">
    <Origin>
      <Address>1.1.1.1</Address>
      <Address>1.1.1.2</Address>
    </Origin>
  </Vhost>
</Vhosts>
```

- <Address>

The address of the origin server from which the virtual host will copy content. There is no limit to the number of addresses that can be added. If there are more than two addresses, selection follows the active/active model (round-robin). If the origin server port is 80, it can be omitted.

For example, if the origin server is using a different port such as 8080, then the port number must be specified as 1.1.1.1:8080. There are eight different ways to format addresses using the {IP|Domain}{Port}{Path} format.

Address	Host Header
1.1.1.1	Virtual host name
1.1.1.1:8080	Virtual host name:8080
1.1.1.1/account/dir	Virtual host name
1.1.1.1:8080/account/dir	Virtual host name:8080
example.com	example.com
example.com:8080	example.com:8080
example.com/account/dir	example.com
example.com:8080/account/dir	example.com:8080

As long as the host header is not specified in the *Origin Request Default Header*, the host header from the table above will be transmitted.:

```
<Vhosts>
  <Vhost Name="www.example.com">
    <Origin>
      <Address>origin.com:8888/account/dir</Address>
    </Origin>
  </Vhost>
</Vhosts>
```

For example, the above configuration will request the following to the origin server.

```
GET / HTTP/1.1
Host: origin.com:8888
```

Note: If a path is added to the origin server's address (e.g. example.com/account/dir), then the requested URL will be placed after the path. If a client requests /img.jpg, the resulting address becomes example.com/account/dir/img.jpg.

Standby Origin Server Address

The standby origin server can be configured as follows.:

```
<Vhosts>
  <Vhost Name="www.example.com">
    <Origin>
      <Address>1.1.1.1</Address>
      <Address>1.1.1.2</Address>
      <Address2>1.1.1.3</Address2>
      <Address2>1.1.1.4</Address2>
    </Origin>
  </Vhost>
</Vhosts>
```

- <Address2>

If all <Address> es are working without problems, <Address2> will not be used. However, if any failures are detected in the active servers, a standby server will be used as a replacement until the failed server recovers. If a failure occurs in a standby server, the server will never be used until it can recover.

2.1.9 API Call

STON provides HTTP-based API. API calls are authorized by *Administrator Settings*. If a call is unauthorized, the connection will be terminated immediately.

The STON version can be checked as follows.

```
http://127.0.0.1:10040/version
```

The same API can be called with Linux shell commands.

```
./stonapi version
```

Note: The ampersand (&) is recognized as a delimiter for QueryStrings in the HTTP API, but it means something different in a Linux console. When running commands or using arguments that contain &s, you must wrap the string in quotes ("...&...").

2.1.10 Hardware Status

The following will look up hardware information.

```
http://127.0.0.1:10040/monitoring/hwinfo
```

The results are returned in JSON format.

```
{
  "version": "1.1.9",
  "method": "hwinfo",
```

```

"status": "OK",
"result":
{
  "OS" : "Linux version 3.3.0 ...(omitted)...",
  "STON" : "1.1.9",
  "CPU" :
  {
    "ProcCount": "4",
    "Model": "Intel(R) Xeon(R) CPU           E5606  @ 2.13GHz",
    "MHz": "1200.000",
    "Cache": "8192 KB"
  },
  "Memory" : "8 GB",
  "NIC" :
  [
    {
      "Dev" : "eth1",
      "Model" : "Intel Corporation 82574L Gigabit Network Connection",
      "IP" : "192.168.0.13",
      "MAC" : "00:25:90:36:f4:cb"
    }
  ],
  "Disk" :
  [
    {
      "Dev" : "sda",
      "Model" : "HP DG0146FAMWL (scsi)",
      "Total" : "238787584",
      "Usage" : "40181760"
    },
    {
      "Dev" : "sdb",
      "Model" : "HITACHI HUC103014CSS600 (scsi)",
      "Total" : "144706478080",
      "Usage" : "2101075968"
    },
    {
      "Dev" : "sdc",
      "Model" : "HITACHI HUC103014CSS600 (scsi)",
      "Total" : "144706478080",
      "Usage" : "2012160000"
    }
  ]
}
}

```

2.1.11 Restart/Quit

The following commands restart or quit STON. To avoid unintended results, STON asks for confirmation for restart/quit commands on the web page.

```

http://127.0.0.1:10040/command/restart
http://127.0.0.1:10040/command/restart?key=JUSTDOIT           // Immediately executes command
http://127.0.0.1:10040/command/terminate
http://127.0.0.1:10040/command/terminate?key=JUSTDOIT        // Immediately executes command

```

2.1.12 Caching Reset

The following commands stop the service and discard all cached content. The commands will format all disks and resume the service when completed.

```
http://127.0.0.1:10040/command/cache-clear
http://127.0.0.1:10040/command/cache-clear?key=JUSTDOIT // Immediately executes command
```

In the console window, the following commands will reset all or one of the virtual hosts.

```
./stonapi reset
./stonapi reset/ston.winesoft.co.kr
```

2.2 Chapter 3. Configuration

This chapter will explain the configuration structure of the STON Edge Server and how to apply the settings. Understanding the configuration structure is important in not only setting up the server quickly but also troubleshooting any problems that occur.

Configuration consists of global (server.xml) and virtual host (vhosts.xml) configuration.

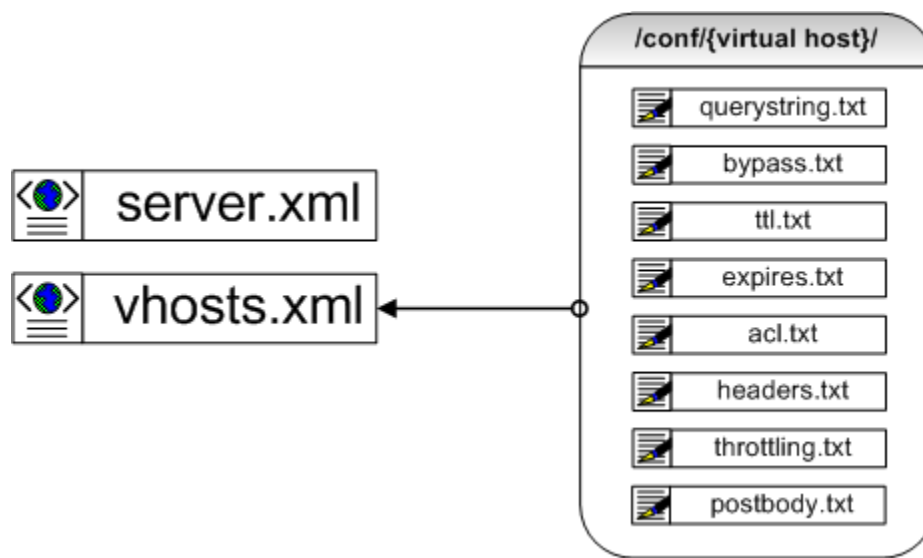


Fig. 2.3: Two XML files are all that is needed.

The two XML files are what comprise most of the service. Various TXT files are used for exception handling for each virtual host and for listing certain functions. The following illustrates the complete form of the XML.

```
<Server>
  <VHostDefault>
    <Options>
      <CaseSensitive>ON</CaseSensitive>
    </Options>
  </VHostDefault>
</Server>
```

However, explaining functions using this format is rather inconvenient, so it will instead be explained in this simplified format.

```
# server.xml - <Server><VHostDefault><Options>

<CaseSensitive>ON</CaseSensitive>
```

Note: The license file (license.xml) is not a configuration file.

2.2.1 Reload Settings

After changing the settings, the administrator must make a call to the API. Aside from system or performance settings, most settings will immediately be applied without interrupting the service.

```
http://127.0.0.1:10040/conf/reload
```

Any changes from settings will be recorded in the *Info Log*.

2.2.2 Global Settings (server.xml)

The server.xml file is the global settings file and can be found in the same path as the execution file. It is an XML format text file.

```
# server.xml

<Server>
  <Host> ... </Host>
  <Cache> ... </Cache>
  <VHostDefault> ... </VHostDefault>
</Server>
```

We will first cover the configuration structure and basic functions. While *Chapter 14. Access Control* and *Chapter 11. SNMP* are also part of global settings, they will be explained in their respective chapters.

Administrator Settings

The following configures the administrator settings of the system.

```
# server.xml - <Server>

<Host>
  <Name>stream_07</Name>
  <Admin>admin@example.com</Admin>
  <Manager Port="10040" HttpMethod="ON" Role="Admin" UploadMultipartName="confile">
    <Allow>192.168.1.1</Allow>
    <Allow Role="Admin">192.168.2.1-255</Allow>
    <Allow Role="User">192.168.3.0/24</Allow>
    <Allow Role="Looker">192.168.4.0/255.255.255.0</Allow>
  </Manager>
</Host>
```

- **<Name>** This configures the server name. If left blank, it will use the system name.

- **<Admin>** This configures the administrator's information (name or email address). This item is only used for the SNMP inquiry.
- **<Manager>** This configures the manager port and the ACL (Access Control List) for administrative purposes. The ACL supports the four forms of IP, IP range, BitMask, and Subnet. If the IP address of the connected session is not authorized by the <Allow> list, the server will block the connection. An IP that calls an API must be configured in the <Allow> list.

After the access conditions, the access authorization (Role) can also be configured. Any requests without authorization will be responded to with a **401 Unauthorized** message. If Role properties are not declared in <Allow> conditions, the Role property from the <Manager> tag will be applied.

- Admin All API calls are allowed.
- User Only *Chapter 10. Monitoring & Statistics* and *Appendix A: Graph* API calls are allowed.
- Looker Only *Appendix A: Graph* API calls are allowed.

In addition, there are other minor administrative properties.

- HttpMethod
 - * ON (default) *HTTP Method* will check ACL when called.
 - * OFF *HTTP Method* will not check ACL when called.
- UploadMultipartName Configures variable names in *Configuration Upload*.

Storage Settings

This section will explain the setup of storage that will store cached content.

```
# server.xml - <Server>

<Cache>
  <Storage DiskFailSec="60" DiskFailCount="10" OnCrash="hang">
    <Disk>/user/cache1</Disk>
    <Disk>/user/cache2</Disk>
    <Disk Quota="100">/user/cache3</Disk>
  </Storage>
</Cache>
```

- **<Storage>** This configures the disk that will store the content. There is no limit to the number of subordinate <Disk> s.

Because the disk is where problems are most likely to happen, it is recommended to set specific fail conditions. If a disk operation fails more than DiskFailCount (default: 10) times within DiskFailSec (default: 60) seconds, then that disk will be excluded from the service. The status of that disk will be shown as “invalid”.

If all disks are excluded, then the server will operate according to the OnCrash property.

- hang (default) Will put all the excluded disks back to work. This behavior is more likely to protect the origin server rather than return to normal service.
- bypass All requests will be passed to the origin server. If the disk recovers, STON will start taking care of the service as soon as it can.
- selfkill STON will be shut down.

The maximum caching capacity for each disk can be configured with the Quota (unit: GB) property. Even when not specifically configured, the LRU (Least Recently Used) algorithm is used to automatically delete old content

to ensure that there is always space on the disk. Therefore, there is no large effect on performance regardless of the file system the administrator chooses to use.

Memory Restriction

This configures the maximum available memory and the ratio of loaded content.

```
# server.xml - <Server>

<Cache>
  <SystemMemoryRatio>100</SystemMemoryRatio>
  <ContentMemoryRatio>50</ContentMemoryRatio>
</Cache>
```

- **<SystemMemoryRatio> (default: 100%)** This ratio will configure the maximum amount of system memory that STON will use. For example, if this property is set to 50% with 16 GB of memory, the system will operate as if there were only 8 GB of memory. This option is especially useful when used together with other processes such as Chapter 17 File System.
- **<ContentMemoryRatio> (default: 50%)** STON improves service quality by caching as much of the Body data as possible from disk to memory. This ratio can be adjusted to optimize the quality of the service according to its type.

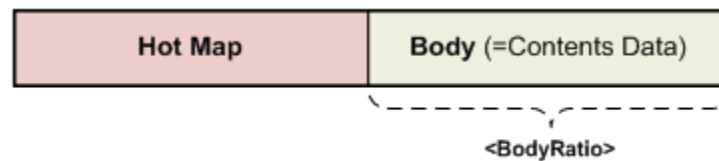


Fig. 2.4: The ratio of memory is configured using the ContentMemoryRatio property.

Using the gaming example, if there are not many files but there is a lot of content, it places a burden on File I/O. In this case, raising the <ContentMemoryRatio> value allows more data to reside in memory, thus improving service quality.



Fig. 2.5: If ContentMemoryRatio is raised, I/O load decreases.

Other Caching Settings

This section covers other caching service functions.

```
# server.xml - <Server>

<Cache>
  <Cleanup>
    <Time>02:00</Time>
    <Age>0</Age>
```

```
</Cleanup>
<Listen>0.0.0.0</Listen>
<ConfigHistory>30</ConfigHistory>
</Cache>
```

- **<Cleanup>** The server carries out system optimization once a day. The optimization procedure mainly consists of disk cleanup, which creates I/O load. In order to prevent a drop in service quality, optimization is performed gradually.
 - **<Time>** (default: 2 AM) Configures when cleanup is performed. A 24-hour clock is used, so for example, 11:10 PM should be written as 23:10.
 - **<Age>** (default: 0, unit: days) If set to a value greater than zero, content that has not been accessed in the specified amount of time will be deleted. This is for the sake of securing available space on the disk in advance to lower the possibility that there will not be enough space during service time.
- **<Listen>** Assigns a list of IP addresses for all virtual hosts to listen to. The default Listen setting of *:80 for all virtual hosts stands for 0.0.0.0:80. The following is an example of enabling specific IP addresses.

```
# server.xml - <Server>

<Cache>
  <Listen>10.10.10.10</Listen>
  <Listen>10.10.10.11</Listen>
  <Listen>127.0.0.2</Listen>
</Cache>
```

- **<ConfigHistory>** (default: 30 days) STON backs up the configuration settings when changes are made. The configuration files will be compressed into one file and saved at `./conf/`. The file will be named in a “DATE_TIME_HASH.tgz” format, as follows.

```
20130910_174843_D62CA26F16FE7C66F81D215D8C52266AB70AA5C8.tgz
```

If two files have identical hash values, it means they have identical settings. Even if *Restore Configuration* is called, it will be saved as a new configuration. A backup file will only be stored for the set amount of time after Cleanup is performed. There is no limit to the amount of time that backups can be stored.

Forced Cleanup

Cleanup is executed with an API call. An **<Age>** parameter can be attached.

```
http://127.0.0.1:10040/command/cleanup
http://127.0.0.1:10040/command/cleanup?age=10
```

If **<Age>** is zero, cleanup will be performed only when there is insufficient disk space. If **<Age>** is greater than 0, then content that has not been accessed for that amount of days will be deleted.

Virtual Host Default Settings

Administrators can configure each virtual host with different settings. However, it can be exhausting to set identical settings for new virtual hosts. All virtual hosts will inherit **<VHostDefault>**.

In the figure above, `www.example.com` does not override any value and thus has the values `A=1` and `B=2`. Meanwhile, `img.example.com` has overridden the value of `B` and thus has the values `A=1` and `B=3`. Administrators will normally keep services with similar attributes on the same server, making inheritance extremely effective.

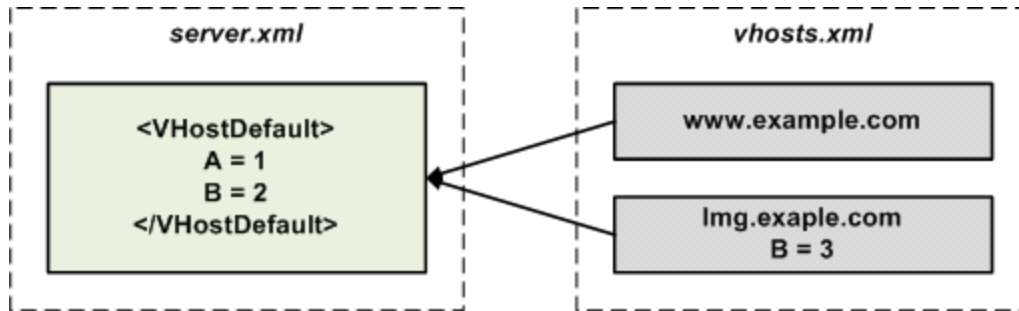


Fig. 2.6: A simple inheritance.

`<VHostDefault>` consists of five function-based subordinate tags.

```
# server.xml - <Server>

<VHostDefault>
  <Options> ... </Options>
  <OriginOptions> ... </OriginOptions>
  <Media> ... </Media>
  <Stats> ... </Stats>
  <Log> ... </Log>
</VHostDefault>
```

For example, the function for [Chapter 16. Media](#) is configured in the `<Media>` tag.

2.2.3 Virtual Host Settings (vhosts.xml)

The `vhosts.xml` file is recognized as the virtual host settings file and can be found in the same path as the execution file. There is no limit to the amount of virtual hosts that are allowed.

```
# vhosts.xml

<Vhosts>
  <Vhost Status="Active" Name="www.example.com"> ... </Vhost>
  <Vhost Status="Active" Name="img.example.com"> ... </Vhost>
  <Vhost Status="Active" Name="vod.example.com"> ... </Vhost>
</Vhosts>
```

Create/Remove Virtual Host

Virtual hosts are set up with the `<Vhost>` tag within the `<Vhosts>` tag.

```
# vhosts.xml - <Vhosts>

<Vhost Status="Active" Name="www.example.com">
  <Origin>
    <Address>10.10.10.10</Address>
  </Origin>
</Vhost>
```

- `<Vhost>` configures the virtual host.
 - `Status` (default: `Active`) An inactive status means the virtual host does not run. Cached content is still stored.

- Name The name of the virtual host. Identical names cannot be used.

If a `<Vhost>` tag is erased then the corresponding virtual host is deleted, along with all of its stored content. Even if the virtual host is readed, the deleted content cannot be restored.

Discovering a Virtual Host

The following is the simplest form of an HTTP request.

```
GET / HTTP/1.1
Host: www.example.com
```

Most web servers will discover virtual hosts with a Host header. If a virtual host wants to operate under different names, the `<Alias>` tag can be used.

```
# vhosts.xml - <Vhosts>

<Vhost Name="example.com">
  <Alias>another.com</Alias>
  <Alias>*.sub.example.com</Alias>
</Vhost>
```

- **<Alias>** This option configures the alias of a virtual host. There is no limit to the amount of aliases that can be assigned. Aliases can be assigned using both specific domain names (another.com) or patterned domain names (*.sub.example.com). For patterned domain names, only a simple format with an asterisk as a prefix is supported.

When discovering a virtual host, follow the procedures below.

1. Does the Name of the `<Vhost>` match?
2. Does the specific `<Alias>` match?
3. Does the patterned `<Alias>` match?

Facade Virtual Host

Because `<Alias>` is just a nickname for the virtual host, it will not provide separate statistics and logs. If you want to use the same virtual host but obtain different *Client Statistics* and *Access Log* depending on the domain, a Facade Virtual Host can be configured.

```
# vhosts.xml - <Vhosts>

<Vhost Name="example.com">
  ...
</Vhost>

<Vhost Name="another.com" Status="facade:example.com">
  ...
</Vhost>
```

This can be done by inputting `facade: + virtual host` into the `Status` property. In the previous example, the *Client Statistics* and *Access Log* will be recorded for clients that request another.com, not example.com.

Sub-Path

A single virtual host can have different sub-paths. These sub-paths can be configured to be handled by separate virtual hosts.

```
# vhosts.xml - <Vhosts>

<Vhost Name="sports.com">
  <Sub Status="Active">
    <Path Vhost="baseball.com">/baseball/<Path>
    <Path Vhost="football.com">/football/<Path>
    <Path Vhost="photo.com">/*.jpg<Path>
  </Sub>
</Vhost>

<Vhost Name="baseball.com" />
<Vhost Name="football.com" />
<Vhost Name="photo.com" />
```

- If the page path or pattern matches the <Sub> input, then it will be sent to the corresponding virtual host. If they do not match, then the page will be handled by the current virtual host.
 - Status (default: Active) Sub-paths are ignored when inactive.
 - <Path> If the URI requested by the client and the path match, the request will be sent to Vhost. Only paths or patterns are allowed.

```
<Path Vhost="baseball.com">baseball<Path>
<Path Vhost="photo.com">*.jpg<Path>
```

If input as above, they will be parsed as /baseball/ and /*.jpg, respectively.

For example, if the client requests the following, the request will be sent to the football.com virtual host.

```
GET /football/rank.html HTTP/1.1
Host: sports.com
```

Default Virtual Host

A default virtual host can be assigned for cases when a virtual host cannot be found for a request. If a default virtual host is not assigned, the request will be abandoned.

```
# vhosts.xml

<Vhosts>
  <Vhost Status="Active" Name="www.example.com"> ... </Vhost>
  <Vhost Status="Active" Name="img.example.com"> ... </Vhost>
  <Default>www.example.com</Default>
</Vhosts>
```

- **<Default>** Configures the name of the default virtual host. It must use a string identical to the Name property from a <Vhost> tag.

Service Address

This section explains how to configure the service address.

```
# vhosts.xml - <Vhosts>

<Vhost Name="www.example.com">
  <Listen>*:80</Listen>
</Vhost>
```

- **<Listen> (default: *:80)** The service address is configured in an {IP}:{Port} format. If written as *:80, for example, then all requests that arrive at port 80 from the NIC will be handled. If a service is supposed to process only requests from a specific address (1.1.1.1) and port (90), then the following setting will do.

```
# vhosts.xml - <Vhosts>

<Vhost Name="www.example.com">
  <Listen>1.1.1.1:90</Listen>
</Vhost>
```

Note: If you do not wish to open the service port, you can configure with the OFF setting.

```
# vhosts.xml - <Vhosts>

<Vhost Name="www.example.com">
  <Listen>OFF</Listen>
</Vhost>
```

Virtual Host - Exceptions (.txt)

There are some cases during the service when the following exceptions should be allowed.

- POST requests are not allowed in general, but a POST request from a specific URL should be allowed.
- STON responds to all GET requests in general, but requests from a specific IP band may want to be bypassed to the origin server.
- A limit should be placed on transmission speeds for specific countries.

These exceptions are not configured in the XML file; rather, settings are saved as TXT files under the `./svc/virtualhost/` directory. Each virtual host has its own independent exception settings. Exceptions will be explained in more detail in the relevant section.

2.2.4 Checking the Virtual Host List

This command queries the virtual host list.

```
http://127.0.0.1:10040/monitoring/vhostslist
```

The result is returned in JSON format.

```
{
  "version": "1.1.9",
  "method": "vhostslist",
  "status": "OK",
  "result": [ "www.example.com", "www.foobar.com", "site1.com" ]
}
```

2.2.5 Confirm Configuration

The next step is to confirm the configuration files. Each TXT file must be clearly assigned to a specific virtual host.

```

http://127.0.0.1:10040/conf/server.xml
http://127.0.0.1:10040/conf/vhosts.xml
http://127.0.0.1:10040/conf/querystring.txt?vhost=www.example.com
http://127.0.0.1:10040/conf/bypass.txt?vhost=www.example.com
http://127.0.0.1:10040/conf/ttl.txt?vhost=www.example.com
http://127.0.0.1:10040/conf/expires.txt?vhost=www.example.com
http://127.0.0.1:10040/conf/acl.txt?vhost=www.example.com
http://127.0.0.1:10040/conf/headers.txt?vhost=www.example.com
http://127.0.0.1:10040/conf/throttling.txt?vhost=www.example.com
http://127.0.0.1:10040/conf/postbody.txt?vhost=www.example.com

```

2.2.6 Configuration History

The following command allows you to browse backup configuration histories.

```

http://127.0.0.1:10040/conf/latest
http://127.0.0.1:10040/conf/history

```

The result is returned in JSON format. Checking only the latest configuration is fastest using `/conf/latest`.

```

{
  "history" :
  [
    {
      "id" : "5",
      "conf-date" : "2013-11-06",
      "conf-time" : "15:26:37",
      "type" : "loaded",
      "size" : "16368",
      "hash" : "D62CA26F16FE7C66F81D215D8C52266AB70AA5C8",
      "ver": "1.2.8"
    },
    {
      "id" : "6",
      "conf-date" : "2013-11-07",
      "conf-time" : "07:02:21",
      "type" : "modified",
      "size" : "27544",
      "hash" : "F81D215D8C52266AB70AA5C8D62CA26F16FE7C66",
      "ver": "1.2.8"
    }
  ]
}

```

- `id` The unique identification number (+1 per reload).
- `conf-date` Configuration modified date.
- `conf-time` Configuration modified time.
- `type` When settings take effect.
 - `loaded` When STON is loaded.
 - `modified` When a configuration is modified (by administrator or WM).
 - `uploaded` When a configuration file is uploaded via API.
 - `restored` When a configuration file is restored via API.

- `size` The size of a configuration file.
- `hash` The hash value of the configuration file using the SHA-1 algorithm.

2.2.7 Restore Configuration

This command restores the configuration to when a certain hash value or id was created. If both the hash value and id are stated in the command, the hash value takes precedence. If rollback occurs successfully, the result will be “200 OK”, while a failure will result in “500 Internal Error”.

```
http://127.0.0.1:10040/conf/restore?hash=...
http://127.0.0.1:10040/conf/restore?id=...
```

2.2.8 Configuration Download

This command will download a configuration of a time when a hash value or id was created. If both the hash value and id are stated in the command, the hash value takes precedence. The Content-Type will be displayed as “application/x-compressed”. If a hash value is not stated and the id cannot be found, the command will return “404 NOT FOUND”.

```
http://127.0.0.1:10040/conf/download?hash=...
http://127.0.0.1:10040/conf/download?id=...
```

2.2.9 Configuration Upload

This command will upload the configuration file using the HTTP Post method (Multipart supported).

```
http://127.0.0.1:10040/conf/upload
```

The address, Content-Length, and Content-Type (“multipart/form-data”) must be clearly stated in the command as shown below.

```
POST /conf/upload
Content-Length: 16455
Content-Type: multipart/form-data; boundary=.....
```

When the upload is completed, the configuration file will be extracted and applied to the system immediately.

In the multipart method, “confile” is used as a default name. This name can be changed in the `UploadMultipartName` property of the `<Manager>` tag.

```
<form enctype="multipart/form-data" action="http://127.0.0.1:10040/conf/upload" method="POST">
  <input name="confile" type="file" />
  <input type="submit" value="Upload" />
</form>
```

2. HTTP Caching

3.1 Chapter 4. Caching Policy

This chapter will cover the Time To Live (TTL), the Caching Key, and the expiration policy, which are fundamental to the service. Stored content is only available for the amount of time given by the TTL. Standard HTTP protocol specifies that Cache-Control can be used to set the TTL. Service quality can be improved through the use of various TTL policies and *Chapter 5. Content Purge*.

HTTP has various standards to classify content. As such, various Caching Keys exist as well. Not only will there be less load on the origin server with less content changes, but the service will be easier to scale up as well. In this chapter, we will discuss various ways to set up optimized expiration policies for a service.

If you want to apply the upcoming configurations to all virtual hosts as a default configuration, you can do so under the <VHostDefault> To do the opposite and apply them to specific virtual hosts, use the <Vhost> tag.

The **Caching-Key** is a distinct value that classifies content. It is similar to how a file system classifies files using a distinct path (e.g. ./user/conf.txt). It is easy to confuse Caching Keys with URLs. However, depending on the various functions of HTTP, identical URLs could return different content.

3.1.1 Time To Live (TTL)

The TTL is the amount of time stored content stays available. A longer TTL setting will reduce load on the origin server, but modifications will take longer to be applied because they must wait until the TTL expires. Conversely, a shorter TTL will mean higher load on the origin server due to more frequent requests for modification checks. The service can run smoothly when an appropriate TTL setting is found and the origin server load is decreased. When a TTL is set, it will not change until it expires. A new TTL is only applied to a file when the old TTL expires. Administrators can use API functions such as *Purge*, *Expire*, *ExpireAfter*, and *HardPurge* to change the TTL.

Default TTL

By default, the TTL is set based on the response of the origin server. The stored content will be provided until the TTL expires. When the TTL expires, a check request for modified content (**If-Modified-Since** or **If-None-Match**) will be sent to the origin server. If the origin server returns a 304 Not Modified response, the TTL is extended.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<TTL>
  <Res2xx Ratio="20" Max="86400">1800</Res2xx>
  <NoCache Ratio="0" Max="5" MaxAge="0">5</NoCache>
```

```

<Res3xx>300</Res3xx>
<Res4xx>30</Res4xx>
<Res5xx>30</Res5xx>
<ConnectTimeout>3</ConnectTimeout>
<ReceiveTimeout>3</ReceiveTimeout>
<OriginBusy>3</OriginBusy>
</TTL>

```

Except for `Ratio` (0~100), all units are in seconds.

- `<Res2xx>` (default: 1800 sec, Ratio: 20, Max: 86400) Sets the TTL when the origin server responds with 200 OK. When content is first stored, it is set to expire after `<Res2xx>` seconds. After the TTL expires, if the content on the origin server has not changed (304 Not Modified), then the TTL is extended according to the `Ratio` value (0~100). The TTL can be increased up to `Max` seconds.
- `<NoCache>` (default: 5 sec, Ratio: 0, Max: 5, MaxAge: 0) This function works identically to `<Res2xx>`, but is only used when the origin server responds with “no-cache”.

```
cache-control: no-cache or private or must-revalidate
```

If `MaxAge` is greater than 0, max-age can be applied.

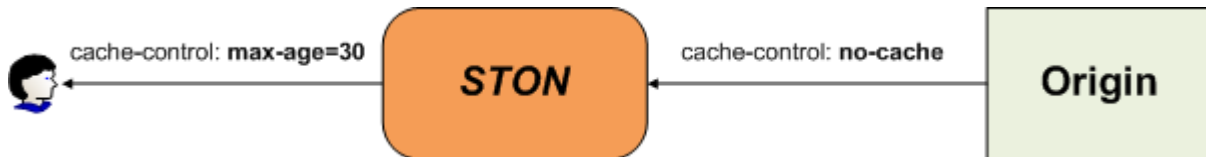


Fig. 3.1: Files are cached for Max-Age seconds.

- `<Res3xx>` (default: 300 sec) Sets the TTL when the origin server responds with “3xx”. This function is frequently used for redirects.
- `<Res4xx>` (default: 30 sec) Sets the TTL when the origin server responds with “4xx”. Responses are often **404 Not Found**.
- `<Res5xx>` (default: 30 sec) Sets the TTL when the origin server responds with “5xx”. This usually occurs due to an internal error in the origin server.
- `<ConnectTimeout>` (default: 3 sec) Sets the TTL when the origin server cannot be reached. If the content is already saved, then the TTL is extended for `<ConnectTimeout>` seconds. If the content is not saved, then an error status will be returned for `<ConnectTimeout>` seconds. The intention is to lessen the burden on the origin server for a TTL amount of time, not to provide an error status to the service.
- `<ReceiveTimeout>` (default: 3 sec) Sets the TTL when the connection is successful but data could not be acquired. The intention is the same as `<ConnectTimeout>`.
- `<OriginBusy>` (default: 3 sec) If the *Overload Detection* condition is satisfied, then the TTL of expired content will be extended for the set amount of time without making requests to the origin server. This is so that additional load is not placed on the origin server.

Note: If the TTL is set to zero, content will expire as soon as it is provided. If you want to have the origin server respond to all requests, a bypass is recommended.

Custom TTL

Separate TTLs can be set for each URL. Fixed TTLs can be set for content that match specific URLs or patterned URLs. This can be configured in `/svc/{virtual host name}/ttl.txt`.

```
# /svc/www.example.com/ttl.txt
# Commas (,) are used as delimiters and the unit of time is seconds.

*.jsp, 10
/, 5
/index.html, 5
/script/*.js, 300
/image/ad.jpg, 1800
```

Even if you add `*.html` to set separate TTLs for all pages (e.g. `html`, `php`, `jsp`), this will not set the TTL for the first page (`/`). The HTTP protocol cannot identify what page is set as the first page (e.g. `index.php`, `default.jsp`) for the origin server. Therefore, in order to set a TTL for every page, a `/` should always be added.

TTL Priority

The order in which TTL is applied can be configured.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<TTL Priority="cc_nocache, custom, cc_maxage, rescode">
... (omitted) ...
</TTL>
```

This can be configured using the `Priority` (default: `cc_nocache`, `custom`, `cc_maxage`, `rescode`) item in the `<TTL>` tag.

- `cc_nocache` When the origin server responds with “Cache-Control: no-cache”.
- `custom` *Custom TTL*.
- `cc_maxage` When the origin server displays maxage in Cache-Control.
- `rescode` The default TTL for response codes from the origin server.

Abnormal TTL Extension

It’s obvious that an error has occurred if there is no response from the origin server, but there are cases when an error will have occurred while the server continues to respond normally at times. For example, it may lose connection with the storage that has the content, or it may decide that regular service is unavailable. The response will usually be a 4xx response (usually **404 Not Found**) for the former or a 5xx response (usually **500 Internal Error**) for the latter.

However, if the content is already stored, then it is more effective to extend the TTL to prevent total service failure rather than rely the origin server’s responses.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<TTLExtensionBy4xx>OFF</TTLExtensionBy4xx>
<TTLExtensionBy5xx>ON</TTLExtensionBy5xx>
```

- `<TTLExtensionBy4xx>`
 - `OFF` (default) Updates content with a 4xx response.

- ON Acts as if the response was a **304 Not Modified**.

You should also check to see if the 4xx response was intentional.

- <TTLExtensionBy5xx>
 - ON (default) Acts as if the response was a **304 Not Modified**.
 - OFF Updates content with a 5xx response.

A normal server will not return a 5xx response, as it is used to reduce the load on the origin server by invalidating content from a temporary server error.

3.1.2 Update Policy

Content will be updated after the TTL expires and the update check is confirmed at the origin server.

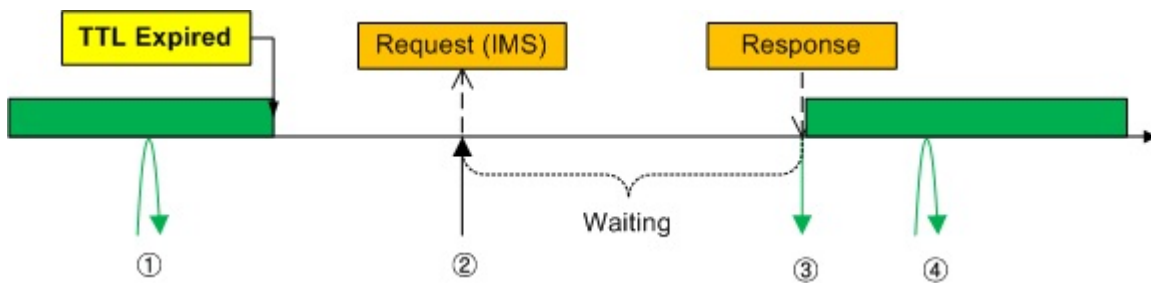


Fig. 3.2: A response after checking for modifications.

1. The TTL is valid and an immediate response is given.
2. The TTL has expired, so a modification check (If-Modified-Since) is requested to the origin server. There is no response to the client until this check is performed.
3. When a response is returned by the origin server, either the TTL is extended or the content is swapped. With confirmation from the origin server, a response will be made to the client.
4. An immediate response is given for the checked content until its TTL expires.

For services like HD videos or games where transfer speed is more important than the response rate, this process is not very meaningful. With bulk data, it doesn't matter that the origin server can respond within ten seconds because the transfer time takes much longer. Rather, since it is content that isn't accessed frequently, it will need more renewal checks.

Meanwhile, online shopping malls are a different story. Web pages loading quickly is more important than anything else. The client's screen must load in 1~2 seconds. In other words, the transfer time is more important than the response rate.

At this point, if the TTL expires and a update check must be performed, it could cause a huge delay. Considering the fact that most shopping malls must handle millions of items of content simultaneously, you must assume that update checks are constantly occurring on the origin server.

What we want is to stably transfer cached content regardless of any origin server error or delay.

These different requirements have led to the development of the background content renewal function.

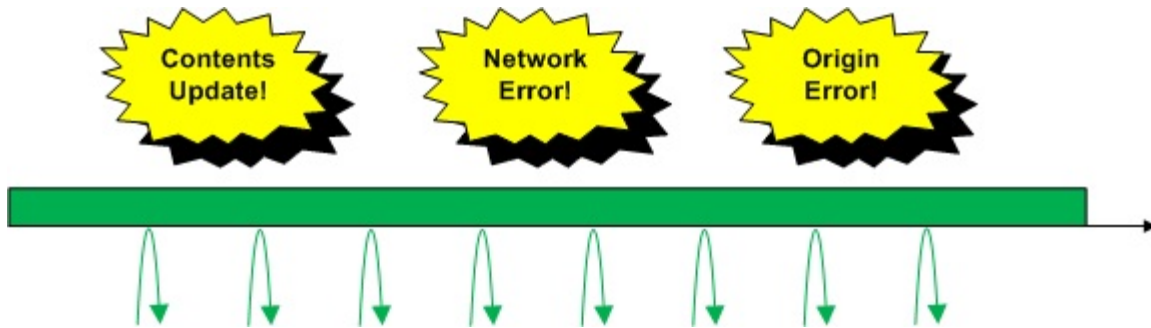


Fig. 3.3: We're not afraid of errors!

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<RefreshExpired>ON</RefreshExpired>
```

- <RefreshExpired>

- ON (default) Responds after the modification check.
- OFF Responds without waiting for the modification check response. Content will be swapped when new content is downloaded.

OFF The OFF setting is generally used because content is not changed very often.

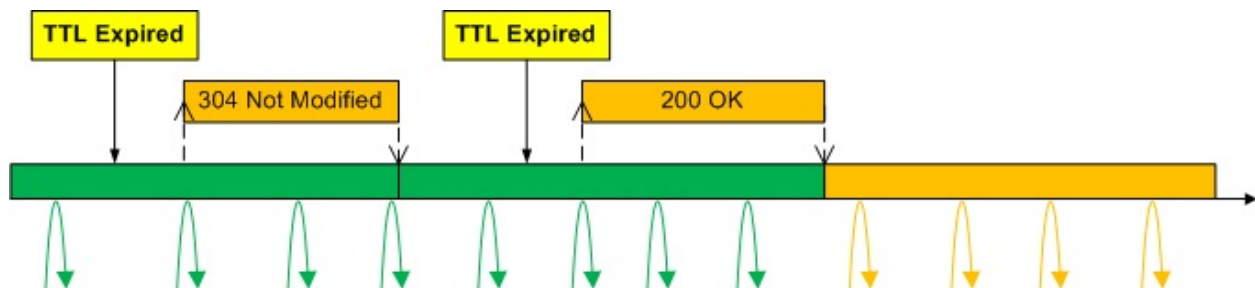


Fig. 3.4: There is no need to wait if content is not sensitive to changes.

As seen in the above image, because updating from the origin server takes place in the background, the cached content can be immediately sent to the client without waiting. If the origin server responds with **304 Not Modified**, the TTL is extended. When the file is updated and the origin server responds with **200 OK**, the file is smoothly replaced after it is fully downloaded. After the file is updated, users will receive the new file (colored yellow). Regardless of variables such as network failures or server failures, content updating will take place in the background and result in no service delays.

TTL Expiration when Clients Request no-cache

If there is at least one no-cache setting in the client's HTTP request, then content can be made to expire right away.

```
GET /logo.jpg HTTP/1.1
...
cache-control: no-cache or cache-control:max-age=0
```

```
pragma: no-cache
...
```

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<NoCacheRequestExpire>OFF</NoCacheRequestExpire>
```

- `<NoCacheRequestExpire>`
 - OFF (default) The request is ignored.
 - ON The TTL is made to expire right away.

The expired content follows the [Update Policy](#).

3.1.3 Accept-Encoding Header

Even though HTTP requests may be for the same URL, depending on the existence of an Accept-Encoding header, different content may be cached. When STON sends a request to the origin server, it has no idea of knowing if the file is compressed or not, and it can't check for compression every time, either.

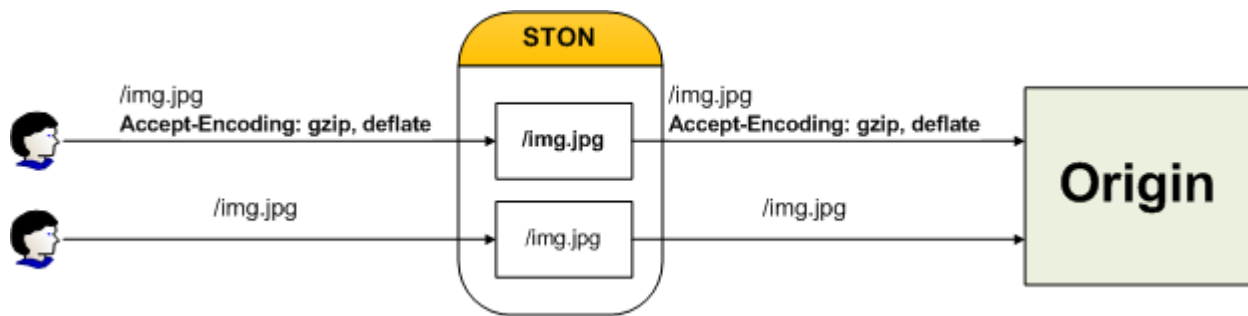


Fig. 3.5: It's impossible to know what kind of response the origin server will give.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<AcceptEncoding>ON</AcceptEncoding>
```

- `<AcceptEncoding>`
 - ON (default) Will recognize the Accept-Encoding header sent by the HTTP client.
 - OFF Will ignore the Accept-Encoding header sent by the HTTP client.

If the origin server does not support compression, or if the bulk file does not require compression, then it is recommended to set `<AcceptEncoding>` to OFF.

3.1.4 Case Sensitivity

STON is unable to tell on its own if the origin server can differentiate between upper and lower case letters.

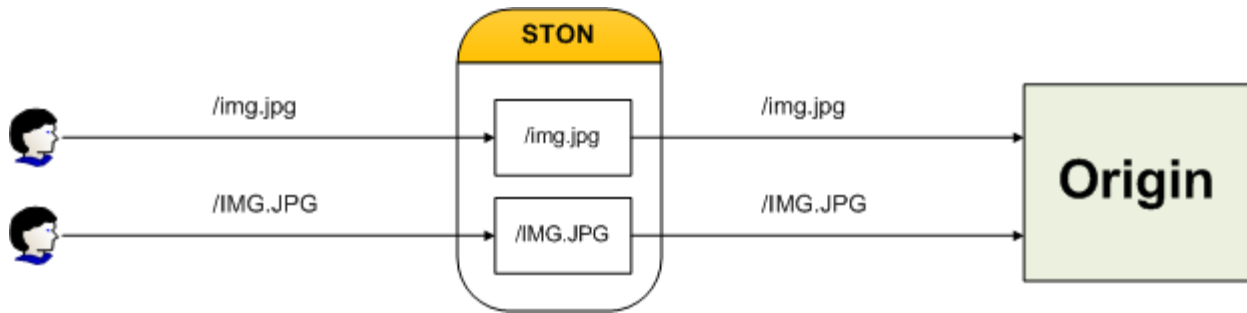


Fig. 3.6: Either the content is the same or a 404 will occur.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<CaseSensitive>ON</CaseSensitive>
```

- <CaseSensitive>
 - ON (default) Differentiates between upper and lower case letters.
 - OFF Does not differentiate. All letters are processed as lower case.

3.1.5 QueryString Differentiation

It is not necessary to identify a query string unless the content is dynamically created by the query string. If a URL contains a meaningless random value or a constantly changing time value, then it can create a lot of load on the origin server.

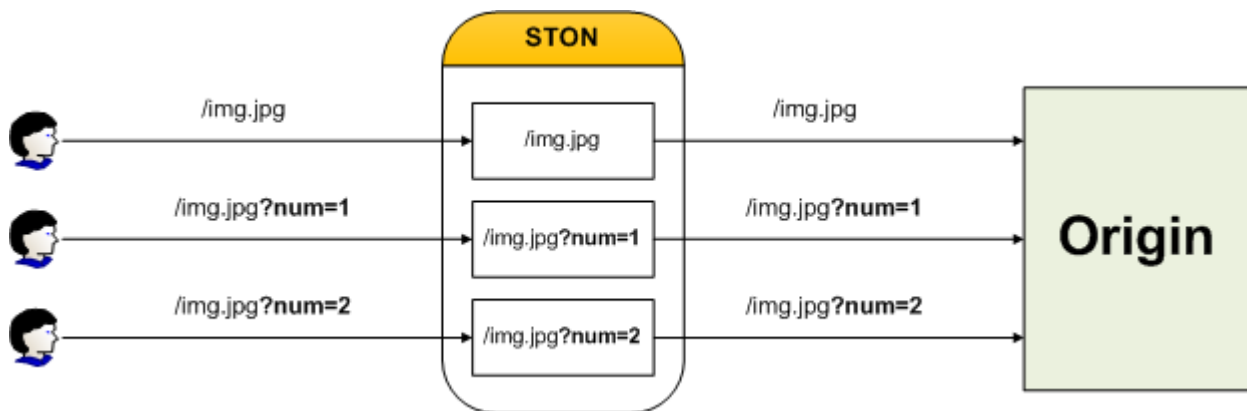


Fig. 3.7: If the content is not dynamic, it is more likely to be identical.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<ApplyQueryString Collective="OFF">ON</ApplyQueryString>
```

- <ApplyQueryString>
 - ON (default) Identifies query strings. If one of the exception cases is met, the query string is ignored.

- OFF Ignores query strings. If one of the exception cases is met, the query string is identified.

Query string exception cases are saved at `/svc/{virtual host name}/querystring.txt`.

```
# ./svc/www.example.com/querystring.txt

/private/personal.jsp?login=ok*
/image/ad.jpg
```

Note that the exception case changes in meaning depending on the setting of `<ApplyQueryString>`. Specific or patterned URLs (only * patterns are allowed) can be used in the configuration.

The `Collective` property comes into play when the *Chapter 5. Content Purge* API is called.

- `Collective`
 - OFF (default) Only the URL parameter will be targeted.
 - ON All content with URLs containing query strings will be targeted, not just the URL parameter.

If the `Collective` property is set to ON and there are many files, CPU load will become higher. It may take longer to search for the correct files, and unforeseen problems may occur. It is recommended to call the *Chapter 5. Content Purge* API using clearly defined URLs with query strings as much as possible.

3.1.6 Vary Header

Content can be classified through the use of Vary headers. Generally, Vary headers are the primary cause of sudden drops in performance on the cache server.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<VaryHeader />
```

- `<VaryHeader>`

Configures the list of Vary headers to be supported among the ones returned by the origin server. Commas (,) are used as delimiters.

For example, if the origin server returns the following as the Vary header, it will be ignored because it is not set in `<VaryHeader>`.

```
Vary: Accept-Encoding, Accept, User-Agent
```

To exclude User-Agent and only recognize Accept-Encoding and Accept headers, do the following.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<VaryHeader>Accept-Encoding, Accept</VaryHeader>
```

To recognize all Vary headers sent by the origin server, do the following.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<VaryHeader>*</VaryHeader>
```

3.1.7 POST Request Caching

POST requests can be configured so that they are cached. POST requests have the same characteristics as URLs, but may differ in Body data.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<PostRequest MaxContentLength="102400" BodySensitive="ON">OFF</PostRequest>
```

- <PostRequest>
 - OFF (default) If a POST request arrives, the session ends.
 - ON POST requests are cached.

Most POST request processing cases use the Body data as Caching Keys. Detailed configuration can be made using the BodySensitive property and exception cases.

- BodySensitive
 - ON (default) Body data is recognized as a Caching Key. Maximum length is set by the MaxContentLength (default: 102400 bytes) property. If one of the exception cases is met, the Body data is ignored.
 - OFF Body data is ignored. If one of the exception cases is met, the Body data is recognized.

POST request exceptions can be set in the file /svc/{virtual host name}/postbody.txt.

```
# /svc/www.example.com/postbody.txt

/bigsale/*.php?nocache=*
/goods/search.php
```

Note that the exception case changes in meaning depending on the setting of BodySensitive. Specific or patterned URLs (only * patterns are allowed) can be used in the configuration.

It is possible to mix up this setting with *GET/POST Bypass*. POST requests may not be cached at all depending on the <BypassPostRequest> (default: ON) setting. As such, to cache POST requests, either <BypassPostRequest> must be set to OFF or an exception case must be set. In order of priority:

- Bypasses the origin server if bypass conditions (*GET/POST Bypass*) are met.
- Terminates the connection if there is no Content-Length header.
- Caches files if PostRequest is set to ON and Content-Length does not exceed MaxContentLength.
- Terminates the request if none of the above scenarios are encountered.

Note: If MaxContentLength is set to too high of a value, a lot of memory will be needed to manage the Caching Key. It is best to set it as small as possible.

3.2 Chapter 5. Content Purge

This chapter will explain how to purge cached content. Because there are many different conditions and environments, many different parts of the API are necessary.

Content cached from the origin server have update cycles based on the *Time To Live (TTL)*. However, if the administrator wishes to immediately have the changes be effective, there is no need to wait until the *Time To Live (TTL)* expires. By using *Purge/Expire/HardPurge*, content can immediately be purged.

The purge API can be called by the browser, but in most cases it is automated. For example, when an FTP file upload is completed, *Purge* is called immediately. Administrators can configure behaviors in several ways as shown below.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<Purge2Expire>NONE</Purge2Expire>
<RootPurgeExpire>ON</RootPurgeExpire>
<ResCodeNoCtrlTarget>200</ResCodeNoCtrlTarget>
```

- `<Purge2Expire>` (default: NONE)

Purge requests will be processed as *Expire* depending on the setting. For example, if a pattern (*.jpg) used alongside *Purge*, the deletion of an unexpectedly large amount of content can cause immense load on the origin server. In this case, if all the requests are processed as *Expire* requests, that server load can be prevented.

- NONE Does not process requests as *Expire*.
- ROOT Processes requests for only the root directory (/) as *Expire*.
- PATTERN Processes requests with patterns as *Expire*.
- ALL Processes all requests as *Expire*.

- `<RootPurgeExpire>` (default: ON)

An unintentional *Purge/Expire* request on the root directory can also cause load on the origin server. This setting can intercept *Purge/Expire* requests and prevent them. This setting takes precedence over `<Purge2Expire>`.

- ON *Purge/Expire* is allowed.
- PURGE Only *Purge* is allowed.
- EXPIRE Only *Expire* is allowed.
- OFF All *Purge/Expire* requests are prevented.

- `<ResCodeNoCtrlTarget>` (default: 200)

Sets the HTTP response code for when *Purge*, *Expire*, *HardPurge*, and *ExpireAfter* have no target object.

Targets can be either URLs or patterns.

```
example.com/logo.jpg      // URL
example.com/img/           // URL
example.com/img/*.jpg      // pattern
example.com/img/*          // pattern
```

While patterned URLs can be called, the amount of actual content being targeted cannot be known until the command is executed. Therefore, the administrator may underestimate the amount of content and try to purge too many targets, consuming more CPU than expected and causing strain on the system.

As such, it is strongly recommended to use only specific URLs. Patterned representations should only be used for the sake of administrative purposes when the service is not running.

Note: For security reasons, accessing specific directories (e.g. example.com/files/) is forbidden and returns 403 FORBIDDEN. However, the root directory is exempt: that is, if a user accesses example.com, their browser will request the root directory (/).

```
GET / HTTP/1.1
Host: example.com
```

The web server will respond with the default page set by the administrator (e.g. index.html). Most web services will return a page, not a directory, for the root directory (/).

However, when the cache server accesses the root directory, it will think it has received a 200 OK page, and won't even be able to know what page was returned. In other words, to the cache server, a directory is just another URL.

example.com/img/	// The resulting page from accessing /img/ on the example.com virtual host
example.com/	// The default page (/) for the example.com virtual host
example.com/img/*	// The /img/ directory and all pages below it on the example.com virtual host
example.com/*	// All content on the example.com virtual host

3.2.1 Purge

Purges the target in order to have it be redownloaded from the origin server. Content will be cached again when it is first accessed after a purge. If the content is not available on the origin server due to an error, the purged content will be restored to keep the service running. This restored content will be updated after the time set by ConnectTimeout.

```
http://127.0.0.1:10040/command/purge?url=...
```

Target content can be designated with URLs and patterns, and can also be designated with vertical bars ("|") to indicate multiple domains and multiple targets. If the domain name is omitted, the most recently used domain name is used.

```
http://127.0.0.1:10040/command/purge?url=http://www.site1.com/image.jpg
http://127.0.0.1:10040/command/purge?url=www.site1.com/image.jpg
http://127.0.0.1:10040/command/purge?url=www.site1.com/image/bmp/
http://127.0.0.1:10040/command/purge?url=www.site1.com/image/*.bmp
http://127.0.0.1:10040/command/purge?url=www.site1.com/image1.jpg|css/style.css|script.js
http://127.0.0.1:10040/command/purge?url=www.site1.com/image1.jpg|www.site2.com/page/*.html
```

The results are returned in JSON format. The number and size of purged items, as well as the elapsed time (units: ms) will be displayed. Content that has already been purged will not be purged again.

```
{
  "version": "2.0.0",
  "method": "purge",
  "status": "OK",
  "result": { "Count": 24, "Size": 3747491, "Time": 12 }
}
```

Using the <Purge2Expire> tag, Purge can be set to Expire under certain conditions. For a response with no results, the HTTP response code can be set with <ResCodeNoCtrlTarget>.

Note: If all origin servers are down due to errors, Purge will not work, as content is unable to be updated.

3.2.2 Expire

The TTL of the target content is set to expire immediately. A check for modification is made when the content is first accessed after expiring. If there is no change, there is no redownload; only the TTL is extended.

```
http://127.0.0.1:10040/command/expire?url=...
```

Everything else is identical to *Purge*.

3.2.3 ExpireAfter

The TTL of the target content is set so that the content expires the input number of seconds after the API is called. ExpireAfter can make the expiration time earlier and make content update faster, or it can make the expiration time later and reduce load on the origin server.

```
http://127.0.0.1:10040/command/expireafter?sec=86400&url=...
```

Though the function call format resembles *Purge* and *Expire*, the sec parameter (in seconds) can also set the expiration date. If the sec parameter is omitted, the default value of 1 day (86400 s) is applied, and setting it to 0 is not allowed. For a response with no results, the HTTP response code can be set with <ResCodeNoCtrlTarget>.

Note: ExpireAfter only sets the current expiration time, and does not affect custom TTLs or default TTLs. There is no change to cached content after an ExpireAfter call.

If the url parameter is entered first, the sec parameter may be recognized as a query string of the url parameter. Therefore, it is recommended to set enter the sec parameter first.

3.2.4 HardPurge

If there is an error in the origin server, *Purge/Expire/ExpireAfter* will retain the content and continue normally. In contrast, HardPurge means the content is permanently deleted. As HardPurge is the most powerful deletion method, deleted content cannot be restored if there is an error. For a response with no results, the HTTP response code can be set with <ResCodeNoCtrlTarget>.

```
http://127.0.0.1:10040/command/hardpurge?url=...
```

3.2.5 Default Purge Behavior

The behavior of content restoration after a Purge API call can be configured.

```
# server.xml - <Server><Cache>

<Purge>Normal</Purge>
```

- <Purge>
 - Normal (default) Behaves as if it was *Purge*. (Content is restored if there is an error.)
 - Hard Behaves as if it was *HardPurge*. (Content is not restored if there is an error.)

3.2.6 HTTP Method

The purge API can be called with an extended HTTP Method.

```
PURGE /sample.dat HTTP/1.1
host: ston.winesoft.co.kr
```

HTTP methods fundamentally work under the manager port and the service port (80). HTTP Method requests sent to the service port can be configured in *Administrator Settings*.

3.2.7 POST Standard

The purge API can be called with POST, as shown below.

```
POST /command/purge HTTP/1.1
Content-Length: 37

url=http://ston.winesoft.co.kr/sample.dat
```

3.3 Chapter 6. Handling HTTP Requests

This chapter will explain the HTTP client session and methods of handling HTTP requests. Parts of this chapter may be difficult to follow without some understanding of HTTP. However, as these functions are not critical to the service, you can simply use the default settings without affecting the quality of service at all.

3.3.1 Session Management

An HTTP session is created when an HTTP client connects to the STON server. Content saved on the server is delivered to the client through the HTTP session. The process from the request to the response is called an **HTTP transaction**. An HTTP session handles multiple HTTP transactions in succession.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<ConnectionHeader>keep-alive</ConnectionHeader>
<ClientKeepAliveSec>10</ClientKeepAliveSec>
<KeepAliveHeader Max="0">ON</KeepAliveHeader>
```

- `<ConnectionHeader>` (default: keep-alive) Configures the Connection header (keep-alive or close) of the HTTP response sent to the client.
- `<ClientKeepAliveSec>` (default: 10 sec) Terminates a session when there is no transaction with the client session for the given amount of time. If the time is set to too large a value, then the number of sessions that are not transacting can grow unexpectedly. Maintaining a large number of sessions can cause load on the system.
- `<KeepAliveHeader>`
 - ON (default) Specifies the Keep-Alive header in the HTTP response. If Max (default: 0) is set to greater than zero, then the Max value will be used for the Keep-Alive header. Each HTTP transaction will reduce the value by one.
 - OFF Omits the Keep-Alive header in the HTTP response.

HTTP Session Maintenance Policies

STON follows Apache policies as much as possible. Specifically, there are many variables in the session maintenance policies based on the value of the HTTP header. The following is a list of items that can influence HTTP session maintenance policies.

- The Connection header specified in the HTTP response (“Keep-Alive” or “Close”)

- Virtual host <Connection> setting
- Virtual host session Keep-Alive time setting
- Virtual host <Keep-Alive> setting

1. When “Connection: Close” is specified in the client HTTP request:

```
GET / HTTP/1.1
...(omitted)...
Connection: Close
```

For an HTTP request like this, a “Connection: Close” response will be returned regardless of the virtual host settings. The Keep-Alive header will not be specified.

```
HTTP/1.1 200 OK
...(omitted)...
Connection: Close
```

When this HTTP transaction is completed, the connection is terminated.

2. When <ConnectionHeader> is set to Close:

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<ConnectionHeader>Close</ConnectionHeader>
```

A “Connection: Close” response will be returned regardless of the client’s HTTP requests. The Keep-Alive header will not be specified.

```
HTTP/1.1 200 OK
...(omitted)...
Connection: Close
```

3. When <KeepAliveHeader> is set to OFF:

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<ConnectionHeader>Keep-Alive</ConnectionHeader>
<KeepAliveHeader>OFF</KeepAliveHeader>
```

The Keep-Alive header will not be specified. The HTTP session can be continuously reused.

```
HTTP/1.1 200 OK
...(omitted)...
Connection: Keep-Alive
```

4. When <KeepAliveHeader> is set to ON:

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<ConnectionHeader>Keep-Alive</ConnectionHeader>
<ClientKeepAliveSec>10</ClientKeepAliveSec>
<KeepAliveHeader>ON</KeepAliveHeader>
```

The Keep-Alive header will be specified. The Keep-Alive time setting of the session is used for the timeout value.

```
HTTP/1.1 200 OK
...(omitted)...
Connection: Keep-Alive
Keep-Alive: timeout=10
```

Note: The Relationship between <Keep-Alive> and <ClientKeepAliveSec>

The <Keep-Alive> setting references the <ClientKeepAliveSec> setting, but <ClientKeepAliveSec> is related to a more fundamental problem. The most important issue in terms of performance or resources is the issue of when to terminate idle sessions, or sessions where HTTP transactions are no longer occurring. HTTP header settings can be changed dynamically and can occasionally be omitted, but the termination of idle sessions is a more complicated problem. Because of this, <ClientKeepAliveSec> is not unified with <KeepAliveHeader> and exists separately.

5. When the Max property of <KeepAliveHeader> is set:

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<ConnectionHeader>Keep-Alive</ConnectionHeader>
<ClientKeepAliveSec>10</ClientKeepAliveSec>
<KeepAliveHeader Max="50">ON</KeepAliveHeader>
```

The max value will be specified in the Keep-Alive header. The session can be used for the number set by the Max property, and each HTTP transaction will decrease the value by one.

```
HTTP/1.1 200 OK
...(omitted)...
Connection: Keep-Alive
Keep-Alive: timeout=10, max=50
```

6. When the max value of Keep-Alive runs out:

As mentioned above, if the max value is set, it will gradually decrease until it hits one.

```
HTTP/1.1 200 OK
...(omitted)...
Connection: Keep-Alive
Keep-Alive: timeout=10, max=1
```

This means that only one more HTTP transaction is possible in the current session. After one more HTTP request, the response will be “Connection: Close” as shown below.

```
HTTP/1.1 200 OK
...(omitted)...
Connection: Close
```

3.3.2 Client Cache-Control

This section covers the settings related to client cache-control.

Age Header

The age header stands for the elapsed time (in seconds) from the moment something is cached, and is calculated by RFC2616 - 13.2.3 Age Calculations.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<AgeHeader>OFF</AgeHeader>
```

- `<AgeHeader>`
 - OFF (default) Omits Age header.
 - ON Specifies Age header.

Expires Header

The following refreshes the Expires header.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<RefreshExpiresHeader Base="Access">OFF</RefreshExpiresHeader>
```

- `<RefreshExpiresHeader>`
 - OFF (default) Specifies the Expires header returned by the origin server to the client. If the Expires header is omitted in the origin server, it will also be omitted in the response to the client.
 - ON The Expires conditions will be reflected in the Expires header. The OFF setting will be applied for content that does not satisfy the conditions.

The Expires condition behaves identically to the `mod_expires` of Apache. You can also configure the Expires header and Cache-Control values of content that matches special conditions (such as URL or MIME Type). The max-age value of the Cache-Control is the difference between the given Expires time and the requested time.

The Expires conditions can be set in `/svc/{virtual host name}/expires.txt`.

```
# /svc/www.exmaple.com/expires.txt
# The delimiter is a comma (,), and the format is {condition},{time},{reference}.

$URL[/test.jpg], 86400
/test.jpg, 86400
*, 86400, access
/test/1.gif, 60 sec
/test/*.dat, 30 min, modification
$MIME[application/shockwave], 1 years
$MIME[application/octet-stream], 7 weeks, modification
$MIME[image/gif], 3600, modification
```

- **Condition** The condition can be set to either a URL or a MIME Type. `$URL[...]` is used for URL, and `$MIME[...]` is used for MIME Type. Patterned expressions can also be used, and if the `$` format is not used, the condition will be recognized as a URL.
- **Time** Sets the Expires expiration time. Common units of time are supported, and if the units are not specified, seconds will be used.
- **Reference** Configures the reference point for the Expires expiration time. If a separate reference point is not specified, then it will use the Access as a reference. Access uses the current time as a reference. The following example indicates that for files that have a MIME Type of image/gif, the Expires header value will be set to 1 day and 12 hours after the access time.

```
$MIME[image/gif], 1 day 12 hours, access
```

Modification uses the Last-Modified time sent by the origin server as a reference. The following example indicates that for all JPG files, the Expires value will be set to 30 minutes after the Last-Modified time.

```
*.jpg, 30 min, modification
```

For Modification, if the calculated time turns out to be in the past relative to the current time, then the current time is used. If the origin server does not provide a Last-Modified header, then an Expires header will not be sent.

ETag Header

The ETag header in the HTTP response sent to the client can be configured.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<ETagHeader>ON</ETagHeader>
```

- <ETagHeader>
 - ON (default) Specified ETag header.
 - OFF Omits ETag header.

3.3.3 Response Headers

Origin Nonstandard Header

For the sake of performance and security, out of the headers sent by the origin server, only the standard headers will be recognized.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<OriginalHeader>OFF</OriginalHeader>
```

- <OriginalHeader>
 - OFF (default) Ignores nonstandard headers.
 - ON Saves all headers (with the exception of cookie, set-cookie, and set-cookie2) and sends them to the client. However, this option can consume more memory.

Via Header

The Via header in the HTTP response sent to the client can be configured.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<ViaHeader>ON</ViaHeader>
```

- <ViaHeader>
 - ON (default) Specifies the Via header as follows.

```
Via: STON/2.0.0
```

- OFF Omits the Via header.

Server Header

The Server header in the HTTP response sent to the client can be configured.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<ServerHeader>ON</ServerHeader>
```

- `<ServerHeader>`
 - ON (default) Specifies the Server header of the origin server.
 - OFF Omits the Server header.

3.3.4 Client Request/Response Header Modification

The client's requests and responses can be modified based on certain conditions.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<ModifyHeader FirstOnly="OFF">OFF</ModifyHeader>
```

- `<ModifyHeader>`
 - OFF (default) Does not modify.
 - ON Modifies the header based on the header modification conditions.

The following explains the points when the header is modified.

- **HTTP Request Header Modification Point** The header is modified when the client's HTTP request is first recognized. If the header gets modified, then it will be handled in its modified state by the cache module. However, the Host header cannot have its URI modified.
- **HTTP Response Header Modification Point** The header is modified just before the response to the client. However, the Content-Length cannot be changed.

Header modification conditions can be set in `/svc/{virtual host name}/headers.txt`. Multiple conditions can be set, and if header meets all the conditions, then all modifications will be applied in order.

If only the first condition should be applied, then the `FirstOnly` property should be set to ON. If different conditions attempt to modify the same header, then the result will be Last-Win from set or specified by put or append.

```
# /svc/www.example.com/headers.txt
# The delimiter is a comma (,).

# Request Modification
# The format is {Match}, {$REQ}, {Action(set|put|append|unset)}.
$IP[192.168.1.1], $REQ[SOAPAction], unset
$IP[192.168.2.1-255], $REQ[accept-encoding: gzip], set
$IP[192.168.3.0/24], $REQ[cache-control: no-cache], append
$IP[192.168.4.0/255.255.255.0], $REQ[x-custom-header], unset
$IP[AP], $REQ[X-Forwarded-For], unset
$HEADER[user-agent: *IE6*], $REQ[accept-encoding], unset
```

```

$HEADER[via], $REQ[via], unset
$URL[/source/*.zip], $REQ[accept-encoding: deflate], set

# Response Modification
# The format is {Match}, {$RES}, {Action(set|put|append|unset)}, {condition}.
# {condition} can modify the header based on special response codes, but is not mandatory.
$IP[192.168.1.1], $RES[via: STON for CDN], set
$IP[192.168.2.1-255], $RES[X-Cache], unset, 200
$IP[192.168.3.0/24], $RES[cache-control: no-cache, private], append, 3xx
$IP[192.168.4.0/255.255.255.0], $RES[x-custom-header], unset
$HEADER[user-agent: *IE6*], $RES[vary], unset
$HEADER[x-custom-header], $RES[cache-control: no-cache, private], append, 5xx
$URL[/source/*], $RES[cache-control: no-cache], set, 404
/secure/*.dat, $RES[x-custom], unset, 200
/*.mp4, $RES[Access-Control-Allow-Origin: example1.com], set
/*.mp4, $RES[Access-Control-Allow-Origin: example2.com], put

```

{Match} can be set to IP, GeoIP, Header, and URL forms.

- **IP** The format is \$IP[...] and supports the formats of IP, IP Range, Bitmask, and Subnet.
- **GeoIP** The format is \$IP[...] and *GeoIP* must be configured in advance. The country codes [ISO 3166-1 alpha-2](#) and [ISO 3166-1 alpha-3](#) are permitted.
- **Header** The format \$HEADER[Key : Value]. The Value can be either a specific expression or a pattern. If the Value is omitted, the condition will be the existence of a header corresponding to the Key.
- **URL** The format is \$URL[...] and can be omitted. It can be either a specific expression or a pattern.

{REQ} and {RES} configure how to modify the header. *set*, *put*, and *append* configure the header to {Key: Value}, and if the Value is omitted, an empty value ("") will be input. *unset* will only input the {Key}.

{Action} can be set to one of the four settings: *set* , *put* , *append* , *unset*.

- *set* The Key and Value defined in the request/response header is added to the header. If the same Key is used, the new Value overwrites the old.
- *put* (resembles *set*) If the same Key is used, the new Value is added in a new line instead of overwriting the old Value.
- *append* (resembles *set*) If the same Key is used, the old Value and the new Value are attached with a comma (,).
- *unset* The Key defined in the request/response header is deleted from the header.

{Condition} can be set to a specific response code like 200 or 304 or can be set to a group of codes like 2xx, 3xx, 4xx, or 5xx. If {Match} matches but {Condition} does not, then the modification does not take place. If {Condition} is omitted, the response code is not checked.

3.3.5 URL Preprocessing

Regular expressions are used to modify the requested URLs. If URL preprocessing is defined, all client requests (HTTP or File I/O) must pass through the URL Rewriter.



Fig. 3.8: The request can only reach the virtual host by passing through the URL Rewriter.

If an approaching Host name is modified by the URL Rewriter, then it will consider it as if the Host header was modified by the client's HTTP request. URL preprocessing is configured in the virtual host settings (vhosts.xml). While most settings are under the virtual host, URL preprocessing can change the name of the Host requested by the client, so the settings must be on the same level as the virtual host.

```
# vhosts.xml

<Vhosts>
  <Vhost ...> ... </Vhost>
  <Vhost ...> ... </Vhost>
  <URLRewrite ...> ... </URLRewrite>
  <URLRewrite ...> ... </URLRewrite>
</Vhosts>
```

Multiple configurations are allowed, and the regular expressions will be checked in order.

```
# vhosts.xml - <Vhosts>

<URLRewrite AccessLog="Replace">
  <Pattern>www.example.com/([^/]+)/(.*)</Pattern>
  <Replace>#1.example.com/#2</Replace>
</URLRewrite>
```

- `<URLRewrite>`

Configures URL preprocessing. `AccessLog` (default: `Replace`) Configures URLs that will be recorded in the Access log. `Replace` records URLs after processing (`/logo.jpg`), while `Pattern` records URLs after processing (`/baseball/logo.jpg`).

3.3.6 Compression

STON can compress and deliver content in place of the origin server. Content must be categorized by the *Accept-Encoding Header*.

```
Accept-Encoding: gzip, deflate
```

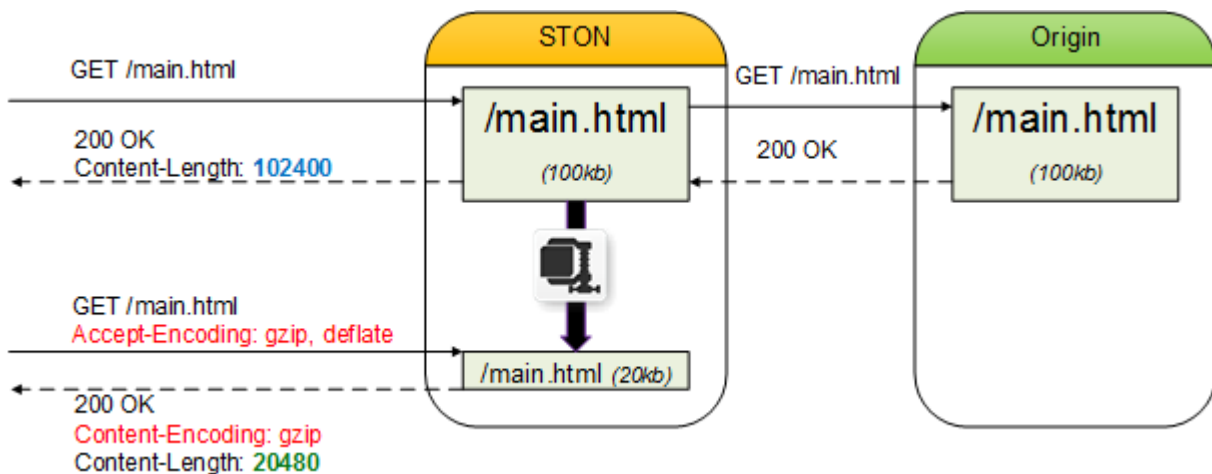


Fig. 3.9: Files are compressed and delivered in real time.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>
```

```
<Compression Method="gzip" Level="6" SourceSize="2-2048">OFF</Compression>
```

- `<Compression>`
 - OFF (default) The compression function is not used.
 - ON The compression function is used with the following properties.
 - * Method (default: gzip) Assigns the compression method. For now, only gzip is supported.
 - * Level (default: 6) Assigns the compression level. This value varies based on the Method used. Only 1~9 are available for gzip. A lower number means compression is faster but worse, while a higher number means compression is slower but better.
 - * SourceSize (default: 2-2048, unit: KB) Assigns a range for the source size. If files are too small, then files might be hardly compressed, while if files are too big, too much CPU might be consumed.

Compressed content is recognized and stored separately from the original content, and requests for the same content will not cause the content to be compressed again. The files to be compressed can be configured in `/svc/{virtual host name}/compression.txt`. The files will be compressed in that order.

```
# /svc/www.example.com/compression.txt
# The delimiter is a comma (,).
# The format is {URL condition}, {Method}, {Level}.

/sample.css, no           // No compression
*.css                    // Compress *.css with default method and level
*.htm, gzip              // Compress *.html with gzip (default level)
*.xml, , 9               // Compress *.xml with level 9 (default method)
*.js, gzip, 5            // Compress *.js with gzip level 5.
```

Compression is a function that consumes a large amount of CPU. The following is a performance test done on gzip (level 9) with files of different sizes.

- OS CentOS 6.3 (Linux version 2.6.32-279.el6.x86_64 (mockbuild@c6b9.bsys.dev.centos.org) (gcc version 4.4.6 20120305(Red Hat 4.4.6-4) (GCC)) #1 SMP Fri Jun 22 12:19:21 UTC 2012)
- CPU Intel(R) Xeon(R) CPU E5-2603 0 @ 1.80GHz (8 processors)
- RAM 8GB
- HDD SAS 275GB X 5EA

Size	Comp. Ratio(%)	Files	La- tency(ms)	Client Traffic(Mbps)	Origin Traffic(Mbps)
1KB	26.25	5288	6.72	40.58	55.02
2KB	57.45	5238	7.20	41.52	97.58
4KB	76.94	5236	7.18	42.44	184.04
8KB	87.61	5021	7.53	41.87	337.80
16KB	93.32	4608	8.30	41.19	616.83
32KB	96.26	3495	13.55	34.53	924.22
64KB	97.79	1783	24.50	20.71	938.83
bootstrap.css(20KB)	86.87	3944	9.67	83.79	638.25
boot- strap.min.js(36KB)	73.00	1791	51.50	139.00	514.86

If `<Compression>` is turned on, only uncompressed files will be requested from the origin server. In other words, the responses from the origin server will be to requests that have omitted the Accept-Encoding header. If the origin

server specifies a Content-Encoding header to a request for uncompressed content, STON will recognize the content as already compressed and will not compress it again.

Note: Content that is already compressed on the origin server that matches <Compression> conditions may be compressed again. Since this can cause problems, this policy is followed.

1. New content will be compressed.
 2. If the content is compressed on the origin server, it will not be compressed again.
 3. If the content is not compressed on the origin server, the corresponding content will be purged and compressed again.
-

3.4 Chapter 7. Origin Server

This chapter will explain the relationship between STON and the origin server. The origin server generally refers to the web server that abides by the HTTP standard. For the sake of protecting the origin server, administrators should have a thorough understanding of the contents of this chapter. Doing so will enable you to establish a service that's flexible and resistant to origin server errors.

The origin server must be protected. With a variety of ways errors can occur, there are a variety of countermeasures to deal with them. Having a proper protection policy for the origin server will make it easier during inspection.

3.4.1 Error Detection and Recovery

If an error occurs in the origin server during caching, the server is automatically excluded. When the server is judged to be stable, it will be brought back into the service.

```
# server.xml - <Server><VHostDefault><OriginOptions>
# vhosts.xml - <Vhosts><Vhost><OriginOptions>

<ConnectTimeout>3</ConnectTimeout>
<ReceiveTimeout>10</ReceiveTimeout>
<Exclusion>3</Exclusion>
<Recovery Cycle="10" Uri="/" ResCode="0" Log="ON">5</Recovery>
```

- <ConnectTimeout> (default: 3 sec)

If a connection to the origin server cannot be made within the set amount of time, it will be considered a connection failure.

- <ReceiveTimeout> (default: 10 sec)

If the origin server does not return an HTTP response for a normal HTTP request within the set amount of time, it will be considered a transaction failure.

- <Exclusion> (default: 3 times)

If an error occurs (<ConnectTimeout> or <ReceiveTimeout>) consecutively for the set number of times, the corresponding server will be excluded from the available server list. The value will be reset to 0 if a successful communication occurs before exclusion.

- <Recovery> (default: 5 times)

If the origin server responds with ResCode when Uri is requested every Cycle consecutively for the set number of times, the corresponding server will be restored. If this value is set to zero, the server will not be restored.

- Cycle (default: 10 sec) Makes a new request after the set amount of seconds.
- Uri (default: /) The Uri to be sent in the request.
- ResCode (default: 0) The response code to be identified as normal. If set to 0, any response will be considered a success regardless of the response code. If set to 200, the response code must be 200 for the response to be identified as normal. Commas (,) can be used to set multiple response codes. For example, if set to "200, 206, 404", then any one of those response codes will be identified as normal.
- Log (default: ON) Records the HTTP transaction that was used for recovery to the *Origin Log*.

3.4.2 Health-Checker

Error Detection and Recovery responds to errors that occur during the caching process. <Recovery> will terminate an HTTP transaction as soon as a response code is received. However, Health-Checker checks for a successful HTTP transaction.

```
# vhosts.xml - <Vhosts><Vhost>

<Origin>
  <Address> ... </Address>
  <HealthChecker ResCode="0" Timeout="10" Cycle="10"
    Exclusion="3" Recovery="5" Log="ON"></HealthChecker>
  <HealthChecker ResCode="200, 404" Timeout="3" Cycle="5"
    Exclusion="5" Recovery="20" Log="ON">/alive.html</HealthChecker>
</Origin>
```

- <HealthChecker> (default: /)

Configures Health-Checker. Multiple configurations are allowed. Uri is used as the input, and CDATA is used for invalid XML characters.

- ResCode (default: 0) The correct response code (multiple codes can be assigned with commas).
- Timeout (default: 10 sec) The available time from the socket connection until the HTTP transaction is completed.
- Cycle (default: 10 sec) The execution period.
- Exclusion (default: 3 times) The number of consecutive failures before excluding the server.
- Recovery (default: 5 times) The number of consecutive successes before reintroducing the server.
- Log (default: ON) Records the HTTP Transaction to the *Origin Log*.

Health-Checker can be configured in multiple ways and can be executed independently of client requests. It does not share information with *Error Detection and Recovery* or other Health-Checkers and uses only its own information to decide exclusion and recovery.

3.4.3 Origin Address Use Policy

The following factors are considered in deciding how to use the origin address (IP).

- *Origin Server* address format (IP or domain) and standby address
- *Error Detection and Recovery*
- *Health-Checker*

As a service is run, the origin address being excluded and recovered will occur frequently. STON uses IP Table-based origin addresses and provides information via the *Origin Status Monitoring* API.

It is simpler to set the origin address with an IP instead of a domain.

- Nothing will be able to change the IP list except for configuration changes.
- The IP address will not expire based on the TTL.
- Exclusion/recovery will work based on the IP address.

If the origin address is set with a domain, it must be resolved in order to obtain the IP. (This will be saved in the *DNS Log*.) The IP list will be able to be changed dynamically, and IPs will only be valid during the TTL.

- The domain will be resolved periodically (1~10 s).
- The IP Table to be used will be organized based on the resolving results.
- All IPs will be valid during the TTL and will not be used when the TTL expires.
- If an identical IP is resolved, the TTL will be refreshed.
- The IP Table cannot be empty. Even if the TTL is expired, the last IP will never be deleted.

Even if the origin address is set to a domain, error/recovery will work based on the IP address. Here there is something to keep in mind. The DNS client (STON) is unable to know the exact IP list for a domain. If a domain consists of only unavailable IP addresses, then it may constantly be in a state of error.

The domain address error/recovery policy is as follows.

- If all known IP addresses for a domain are excluded (Inactive), then the corresponding domain will also be excluded.
- Even if a new IP is resolved, if the domain is excluded then the IP address will also be excluded.
- Even if the TTLs of all IPs expire, this will not change the state of the excluded domain.
- At least one IP of an excluded domain must be recovered for that domain to also be recovered.

It is recommended to improve your understanding of service behavior through the *Origin Status Monitoring* API.

3.4.4 Origin Status Monitoring

An API is used to monitor the state of a virtual host's origin server.

```
http://127.0.0.1:10040/monitoring/origin          // All virtual hosts
http://127.0.0.1:10040/monitoring/origin?vhost=www.example.com
```

The results are given in JSON format.

```
{
  "origin" :
  [
    {
      "VirtualHost" : "example.com",
      "Address" :
      [
        { "1.1.1.1" : "Active" },
        { "1.1.1.2" : "Active" }
      ],
      "Address2" : [  ],
      "ActiveIP" :
      [
        { "1.1.1.1" : 0 },

```

```

        { "1.1.1.2" : 0 }
    ],
    "InactiveIP" : [ ]
},
{
    "VirtualHost" : "foobar.com",
    "Address" :
    [
        { "origin.foobar.com" : "Active" }
    ],
    "Address2" : [ ],
    "ActiveIP" :
    [
        { "5.5.5.5" : 21 },
        { "5.5.5.6" : 60 },
        { "5.5.5.7" : 37 }
    ],
    "InactiveIP" :
    [
        { "5.5.5.8" : 10 },
        { "5.5.5.9" : 184 }
    ]
}
]
}

```

- **VirtualHost** The virtual host name.
- **Address** *Origin Server*. It will return *Active* if the address is being used, and *Inactive* if not being used (due to an error).
- **Address2** *Standby Origin Server Address*. It will return *Active* if the address is being used, and *Inactive* if not being used.
- **ActiveIP** The list of IPs in use and their TTLs. If the origin server is set with an IP address, an identical IP will be shown in *Address* with a TTL of 0. If set with a domain, the values depend on the resolving results. Various IPs and TTLs are used.
- **InactiveIP** The list of IPs not in use and their TTLs. Even though they are not in use, they can still be in a recovery status or being managed by Health-Checker. If the address is not recovered within the TTL, it will be removed.

3.4.5 Origin Status Reset

An API is used to reset the exclusion/recovery of origin servers of a virtual host. The current session will not be reused, and a new connection will be created instead.

```

http://127.0.0.1:10040/command/resetorigin          // All virtual hosts
http://127.0.0.1:10040/command/resetorigin?vhost=www.example.com

```

3.4.6 Overload Detection

Content requested for the first time must always be retrieved from the origin server. However, content already cached can be taken care of more flexibly. If STON detects that the origin server is overloaded, renewal of content can be postponed so as to not increase server load.

```
# server.xml - <Server><VHostDefault><OriginOptions>
# vhosts.xml - <Vhosts><Vhost><OriginOptions>

<BusySessionCount>100</BusySessionCount>
```

- `<BusySessionCount>` (default: 100) If the number of HTTP transactions taking place on the origin server exceeds the set value, it will be considered an overload. So that the origin server isn't further accessed to renew expired content, the TTL is extended by the value of `<OriginBusy>` in *Time To Live (TTL)*. This value can be set to a very large number if you want all requests to go to the origin server.

3.4.7 Origin Selection

This configures the origin server selection policy in the case when the origin server consists of multiple addresses (two or more).

```
# server.xml - <Server><VHostDefault><OriginOptions>
# vhosts.xml - <Vhosts><Vhost><OriginOptions>

<BalanceMode>RoundRobin</BalanceMode>
```

- `<BalanceMode>` (default: RoundRobin)
 - RoundRobin (default) The server will be chosen via round-robin so that all origin servers receive requests uniformly. Connected idle sessions are only used when a request to the corresponding server is necessary.
 - Session A session will be used if it can be reused. If a new session is necessary, the next server will be chosen via round-robin.
 - Hash Content will be requested in a dispersed way following the [consistent hashing](#) algorithm. If a server is chosen, the current session will be reused; if there is no session, a new one will be created.

/	RoundRobin	Session
Load (Requests)	Load is divided equally across servers	Higher load on servers with better responsiveness and reusability
Connection cost	High (Finds a connection or attempts a new one for each server)	Low (Only connects when the session can't be reused)
Reusability	Low (Server division is prioritized)	High (Already-connected sessions are prioritized)
Session count	Many (Sum of simultaneous HTTP transactions for each server)	Few (Only as many sessions as there are HTTP transactions)

3.4.8 Session Recycle

If the origin server supports the Keep-Alive setting, then the connected session will always be reused. However, the origin server can unilaterally terminate connections to recycled sessions. As a result, the connection will need to be recovered, which can potentially lower user responsiveness. This is especially the case for sessions that haven't been used in a while. To prevent this, sessions that aren't reused for the set number of seconds will have their connection be automatically terminated.

```
# server.xml - <Server><VHostDefault><OriginOptions>
# vhosts.xml - <Vhosts><Vhost><OriginOptions>

<ReuseTimeout>60</ReuseTimeout>
```

- `<ReuseTimeout>` (default: 60 s) Terminates sessions that have not been used within the set amount of time. If set to zero, origin server sessions will not be reused.

3.4.9 Range Request

Configures how much content is downloaded at a time. If the content is generally viewed from the front, such as videos, then setting a limit to the download size can reduce unnecessary origin traffic.

```
# server.xml - <Server><VHostDefault><OriginOptions>
# vhosts.xml - <Vhosts><Vhost><OriginOptions>

<PartSize>0</PartSize>
```

- `<PartSize>` (default: 0 MB) If set to greater than zero, a range request will be used to download the set value (MB) starting from the point requested by the client.

Another reason to use `<PartSize>` is to save disk space. Under default settings, STON generates a file with the same size as the original on the disk. However, as long as `<PartSize>` is not zero, the file will be partitioned to the given size and saved.

For example, if a client watches the first minute (10 MB) of a one-hour (600 MB) video, only 10 MB of the disk space will be used. There is some benefit in saving disk space, but because the file is saved in parts, the disk load increases a little.

Note: When content is downloaded for the first time, the content length is unknown and a range request cannot be made. Therefore, if `<PartSize>` is configured, the connection will be closed after the set size is downloaded.

3.4.10 Initializing the Entire Range

Generally, whether a file is downloaded for the first time or is being checked for renewal on the origin server, the same simple GET request is sent.

```
GET /file.dat HTTP/1.1
```

However, origin servers configured to always modify files for general GET requests may have issues because the original file cannot be cached in its original form.

One of the most common examples is when the Apache web server embeds external modules such as `mod_h.264_streaming`. The Apache web server will always respond using the `mod_h.264_streaming` module. As such, the client (in this case, STON) will not receive the file as it originally was but a file modified by the module.

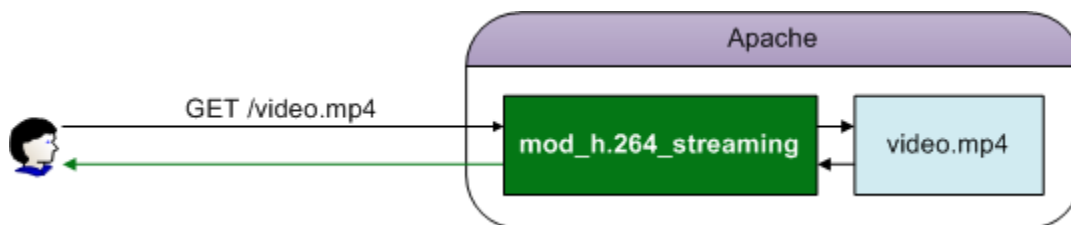


Fig. 3.10: The `mod_h.264_streaming` module always modifies the original file.

A range request can be used to bypass the module and download the original.

```
# server.xml - <Server><VHostDefault><OriginOptions>
# vhosts.xml - <Vhosts><Vhost><OriginOptions>

<FullRangeInit>OFF</FullRangeInit>
```

- `<FullRangeInit>`
 - OFF (default) A normal HTTP request is sent.
 - ON A range request that begins with 0 is sent. In Apache, if the range header is specified, the module is bypassed.

```
GET /file.dat HTTP/1.1
Range: bytes=0-
```

Because the Range can't be known when a file is cached for the first time, Full-Range (starting with 0) is requested. You must verify that a normal response (206 OK) is returned for range requests.

If content is being renewed, the **If-Modified-Since** header will also be specified as below. The origin server must properly respond with **304 Not Modified**.

```
GET /file.dat HTTP/1.1
Range: bytes=0-
If-Modified-Since: Sat, 29 Oct 1994 19:43:31 GMT
```

Note: The following is a list of web servers where `<FullRangeInit>` is confirmed to work properly.

- Microsoft-IIS/7.5
 - nginx/1.4.2
 - lighttpd/1.4.32
 - Apache/2.2.22
-

3.4.11 Keeping Client Requests

You can configure whether client requests are kept or changed via the Caching-Key.

```
# server.xml - <Server><VHostDefault><OriginOptions>
# vhosts.xml - <Vhosts><Vhost><OriginOptions>

<WholeClientRequest>OFF</WholeClientRequest>
```

- `<WholeClientRequest>`
 - OFF (default) The Caching-Key is used as the URL requested to the origin server.
 - ON The URL requested by the client is requested to the origin server.

To raise the Hit Ratio, the following settings are used to select the Caching-Key.

- *Case Sensitivity*
- *QueryString Differentiation*
- *POST Request Caching*

Therefore, the URL and Caching-Key requested to the origin server is determined in the following way.

Setting	Client Requested URL	Origin Requested URL / Caching-Key
<i>Case Sensitivity</i> OFF	/Image/LOGO.png	/image/logo.png
<i>Case Sensitivity</i> ON	/Image/LOGO.png	/Image/LOGO.png
<i>QueryString Differentiation</i> OFF	/view/list.php?type=A	/view/list.php
<i>QueryString Differentiation</i> ON	/view/list.php?type=A	/view/list.php?type=A

If <WholeClientRequest> is set to ON, the URL sent by the client will be sent as is to the origin, regardless of the Caching-Key.

Setting	Client/Origin Requested URL	Caching-Key
<i>Case Sensitivity</i> OFF	/Image/LOGO.png	/image/logo.png
<i>Case Sensitivity</i> ON	/Image/LOGO.png	/Image/LOGO.png
<i>QueryString Differentiation</i> OFF	/view/list.php?type=A	/view/list.php
<i>QueryString Differentiation</i> ON	/view/list.php?type=A	/view/list.php?type=A

When POST requests are cached and requested to the origin server, the body data of the POST request sent by the client is transmitted without modification.

Note: Because URLs sent by the client are not modified in anyway, QueryStrings added using functions such as *Trimming* will also be sent without modification.

3.4.12 Origin Request Default Header

Host Header

Configures the Host header of the HTTP request sent to the origin server. If not specified, the virtual host name will be used.

```
# server.xml - <Server><VHostDefault><OriginOptions>
# vhosts.xml - <Vhosts><Vhost><OriginOptions>

<Host />
```

- <Host> Configures the Host header sent to the origin server. If the origin server uses a port other than 80, the port must be specified.

```
# server.xml - <Server><VHostDefault><OriginOptions>
# vhosts.xml - <Vhosts><Vhost><OriginOptions>

<Host>www.example2.com:8080</Host>
```

If you want to send the Host header from the client to the origin, * is used.

User-Agent Header

Configures the User-Agent header of the HTTP request sent to the origin server.

```
# server.xml - <Server><VHostDefault><OriginOptions>
# vhosts.xml - <Vhosts><Vhost><OriginOptions>

<UserAgent>STON</UserAgent>
```

- <UserAgent> (default: STON) Configures the User-Agent header sent to the origin server.

If you want to send the User-Agent header from the client to the origin, * is used.

XFF (X-Forwarded-For) Header

If STON is placed between the client and the origin server, the origin server will not be able to obtain the client's IP. Therefore, all HTTP requests sent by STON to the origin server have an X-Forwarded-For header.

```
# server.xml - <Server><VHostDefault><OriginOptions>
# vhosts.xml - <Vhosts><Vhost><OriginOptions>

<XFFClientIPOnly>OFF</XFFClientIPOnly>
```

- `<XFFClientIPOnly>`
 - OFF (default) Appends the client's IP to the XFF header sent by the client (IP: 128.134.9.1). If the client did not send an XFF header, only the client IP is displayed.

```
X-Forwarded-For: 220.61.7.150, 61.1.9.100, 128.134.9.1
```

- ON Sends the first address of the XFF header to the origin server.

```
X-Forwarded-For: 220.61.7.150
```

ETag Header Recognition

Configures the recognition of the ETag header returned by the origin server.

```
# server.xml - <Server><VHostDefault><OriginOptions>
# vhosts.xml - <Vhosts><Vhost><OriginOptions>

<OriginalETag>OFF</OriginalETag>
```

- `<OriginalETag>`
 - OFF (default) The ETag header is ignored.
 - ON The ETag header is recognized and an If-None-Match header is appended on content renewal.

3.4.13 Origin Request Header Modification

The HTTP header can be changed when sending HTTP requests to the origin server based on certain conditions.

```
# server.xml - <Server><VHostDefault><OriginOptions>
# vhosts.xml - <Vhosts><Vhost><OriginOptions>

<ModifyHeader FirstOnly="OFF">OFF</ModifyHeader>
```

- `<ModifyHeader>`
 - OFF (default) Keeps the original header.
 - ON The header changes based on conditions.

The point in time the header is changed is when the HTTP request packet is completed, just before it is sent to the origin server. However, range requests cannot be changed.

This function is a sub-function of *Client Request/Response Header Modification*. The \$ORGREQ keyword is used for header changes.

```
# /svc/www.example.com/headers.txt

$URL[/*.mp4], $ORGREQ[x-media-type: video/mp4], set
$IP[1.1.1.1], $ORGREQ[user-agent: media_probe], put
*, $ORGREQ[If-Modified-Since], unset
*, $ORGREQ[If-None-Match], unset
```

Note: If the If-Modified-Since and If-None-Match headers are set to unset, content will always be downloaded when their TTL expires.

3.4.14 Redirect Tracking

If the origin server returns responses from the Redirect category (301, 302, 303, 307), the Location header is tracked to request content.

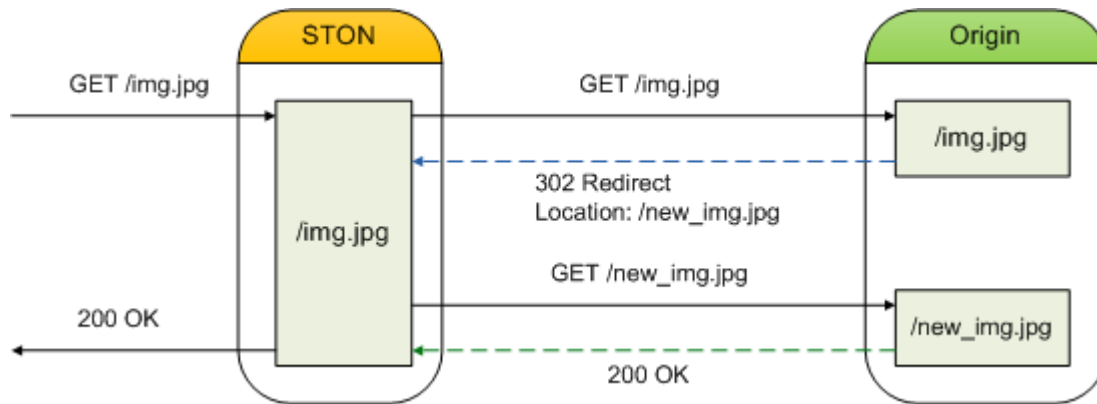


Fig. 3.11: Clients will not know if they are redirected or not.

```
# server.xml - <Server><VHostDefault><OriginOptions>
# vhosts.xml - <Vhosts><Vhost><OriginOptions>

<RedirectionTrace>OFF</RedirectionTrace>
```

- <RedirectionTrace>
 - OFF (default) Saves as a 3xx response.
 - ON Downloads content from the address given in the Location header. If the format of the header is incorrect or there is no header, then tracking will fail. In order to prevent infinite redirection, STON will only redirect once.

3.5 Chapter 8. Bypass (Pass-through)

This chapter will explain how to set a bypass that delegates client request handling to the origin server. Bypasses can be divided into conditions and behaviors.

Bypass takes priority over the caching policy. If a service did not consider the edge server during the design state, it most likely cannot distinguish between static and dynamic resources. In this case, it can be configured so that all

client requests are bypassed, caching only content that is frequently requested according to the log. In general, even a few hours of logging can dramatically decrease the load on the origin server. The real-time information provided by *Chapter 10. Monitoring & Statistics* is there to allow you to tune the service in real time.

Bypasses are not only fast, but they also work on the level of HTTP transactions. No matter how personalized a site is, it will generally be composed of a main page (.html) that changes dynamically, with the remaining 99% being made up of static resources. A bypass version of *Origin Request Default Header* exists separately to match up with the actions of the origin server.

3.5.1 No-Cache Request Bypass

If the client sends a no-cache request, it will be bypassed.

```
GET / HTTP/1.1
cache-control: no-cache or cache-control:max-age=0
pragma: no-cache
```

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<BypassNoCacheRequest>OFF</BypassNoCacheRequest>
```

- `<BypassNoCacheRequest>`
 - OFF (default) The request is handled by the cache module.
 - ON The request is bypassed to the origin server.

Note: This setting is judged by the client's action (likely Ctrl+F5). As a result, a large number of bypasses can cause strain on the origin server.

3.5.2 GET/POST Bypass

A bypass can be set to be the default action of GET/POST requests. It is important to keep in mind that because GET and POST are used differently, their actions will be different as well.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<BypassPostRequest>ON</BypassPostRequest>
<BypassGetRequest>OFF</BypassGetRequest>
```

- `<BypassPostRequest>`
 - ON (default) The POST request is bypassed to the origin server.
 - OFF The POST request is handled by STON.
- `<BypassGetRequest>`
 - OFF (default) The GET request is handled by STON.
 - ON The GET request is bypassed to the origin server.

Bypasses support the same conditions as *Virtual Host ACL*. Exception cases for bypasses can be set in `/svc/{virtual host name}/bypass.txt`.

```
# /svc/www.example.com/bypass.txt
$IP[192.168.2.1-255]
/index.html
```

If cache or bypass conditions are not specified, the opposite of the default setting will be applied. For example, if `<BypassGetRequest>` is set to ON, the exception cases become the caching list. There is a lot of room for confusion, but the second parameter can be set to explicitly state the conditions.

```
# /svc/www.winesoft.co.kr/bypass.txt

$HEADER[cookie: *ILLEGAL*], cache           // Always cache
!HEADER[referer:]                          // Depends on the default setting
!HEADER[referer] & !HEADER[user-agent], bypass // Always bypass
$URL[/source/public.zip]                    // Depends on the default setting
```

The priority of actions is as follows.

1. No-Cache bypass
2. Bypass is specified in bypass.txt
3. Default setting of bypass.txt

3.5.3 Fixed Origin Servers

Some transactions, such as login status, require a one-to-one communication between the origin server and the client. Properties of *GET/POST Bypass* can be used to fix the origin server to the client.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<BypassPostRequest OriginAffinity="ON">...</BypassPostRequest>
<BypassGetRequest OriginAffinity="ON">...</BypassGetRequest>
```

- OriginAffinity
 - ON (default) Guarantees that the client's requests will always be bypassed to the same origin server. However, it is not guaranteed to be the same socket.

There is always the possibility that all the sockets of an origin server will lose connection. However, if this occurs, a new socket connection will simply be requested from the corresponding server.

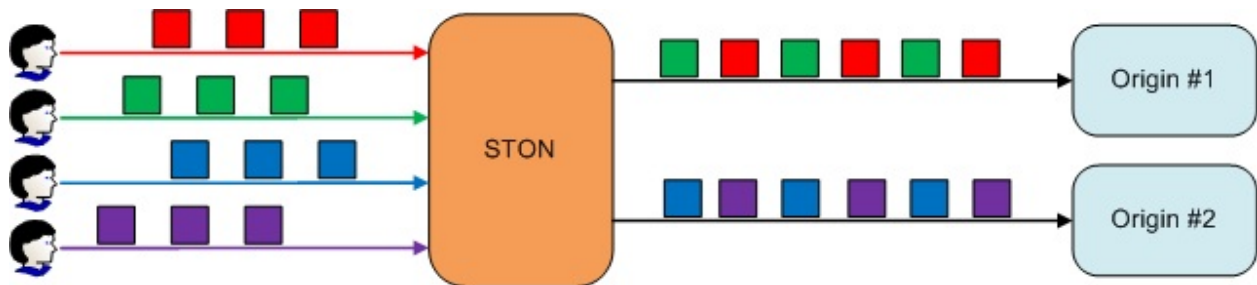
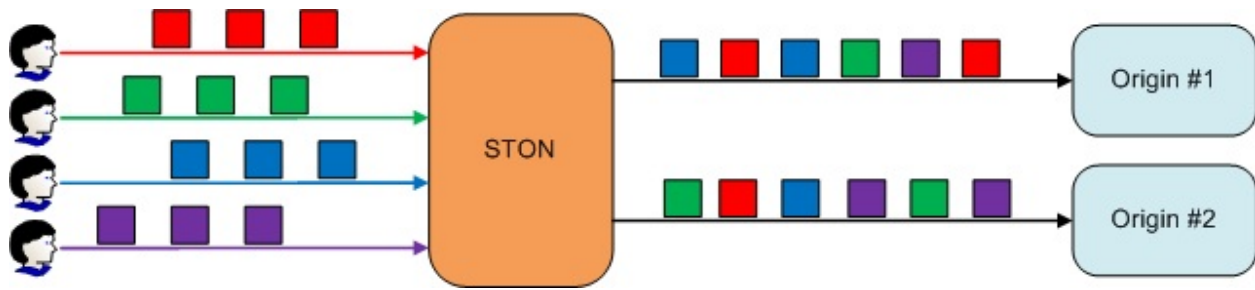


Fig. 3.12: Requests will always be bypassed to the same server.

If the origin server being bypassed to ends up being excluded due to errors or dropped from DNS, requests will be bypassed to a new server instead.

- OFF Will not guarantee which server the client requests will bypass to.

Fig. 3.13: Requests will follow *Origin Selection*.

3.5.4 Fixed Origin Sessions

Each client socket will use a one-to-one bypass session with the origin server.

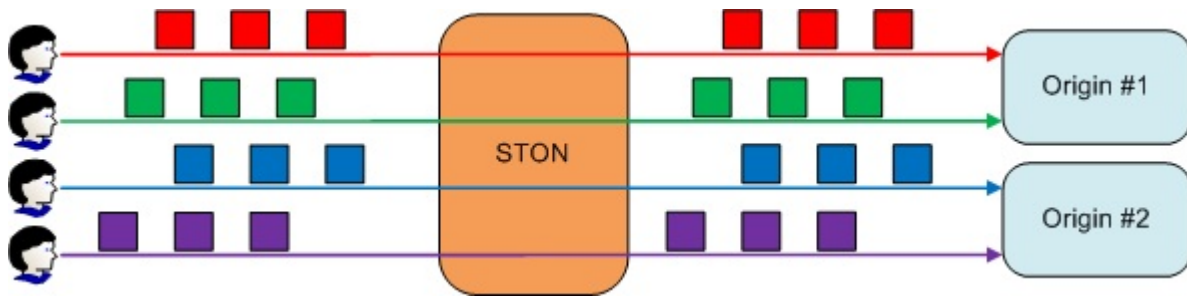


Fig. 3.14: The client will have their own origin session.

Properties of *GET/POST Bypass* can be used to fix the origin session to the client.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<BypassPostRequest Private="OFF">...</BypassPostRequest>
<BypassGetRequest Private="OFF">...</BypassGetRequest>
```

- Private

- ON The client session will have their own private session on the origin server. Requests will always be bypassed to the same server. Either the client or the origin server can terminate the session, at which point the session will be terminated on the other end as well.
- OFF (default) Private sessions are not used.

Just as origin servers hold on to the user's login information within a session, it is helpful if the client handles requests within the same socket as well.

Note: If too many requests are bypassed with `Private`, there will be as many origin server connections as there are clients, creating an immense amount of load. Also, because origin sessions connected in this way are owned by the clients, it could endanger the server to malicious attacks.

Timeout

There are many cases when a bypass responds with results dynamically processed in the origin server. As such, there are many cases when processing speed is slower than static content. Setting a timeout specifically for bypasses is recommended to avoid the system prematurely assuming an error situation.

```
# server.xml - <Server><VHostDefault><OriginOptions>
# vhosts.xml - <Vhosts><Vhost><OriginOptions>

<BypassConnectTimeout>5</BypassConnectTimeout>
<BypassReceiveTimeout>300</BypassReceiveTimeout>
```

- `<BypassConnectTimeout>` (default: 5 sec) If the bypass is unable to connect to the origin server within the set time, it will be considered a connection timeout.
- `<BypassReceiveTimeout>` (default: 5 sec) If there is no response from the origin server within the set time during a bypass, it will be considered a reception timeout.

3.5.5 Bypass Header

A bypass header configures whether or not to apply the bypass setting of *Origin Request Default Header*.

```
# server.xml - <Server><VHostDefault><OriginOptions>
# vhosts.xml - <Vhosts><Vhost><OriginOptions>

<UserAgent Bypass="OFF">...</UserAgent>
<Host Bypass="ON"/>
<XFFClientIPOnly Bypass="ON">...</XFFClientIPOnly>
```

- Bypass Property
 - ON Specifies the configured header.
 - OFF Specifies the relative headers sent by the clients.

3.5.6 Port Bypass

With a port bypass, the packets from a specific TCP port will all be bypassed to the origin server. This setting is exclusive to virtual hosts.

```
# vhosts.xml - <Vhosts>

<Vhost Name="www.example">
  <PortBypass>443</PortBypass>
  <PortBypass Dest="1935">1935</PortBypass>
</Vhost>
```

- `<PortBypass>` Bypasses all packets from the designated port to the same port on the origin server. The `Dest` property configures the destination port on the origin server.

For example, bypassing port 443 will have an effect similar to creating a direct SSL connection with the origin server. Ports being bypassed can never have multiple redundant settings.

Note: Structurally, port bypasses take place in the TCP, a layer beneath HTTP. The reason for setting up a port bypass under a specific virtual host is that the virtual host is needed to collect statistics.

3.6 Chapter 9. HTTPS

This chapter will explain how to configure HTTPS. STON supports up to TLS 1.2, but allows SSL 2.0 only for upgrades due to security reasons. HTTPS is used only between the client and STON. STON does not communicate with the origin server using HTTPS, because it wouldn't be suitable for security and performance reasons if STON used HTTPS as a relay. If the origin server must use HTTPS to communicate, using *Port Bypass* is recommended.

3.6.1 Service Configuration

As long as a specific IP or port is not designated, the default binding service address is “*.443”. This is configured in the global configuration (server.xml).

```
# server.xml - <Server>

<Https>
  <Cert>/usr/ssl/cert.pem</Cert>
  <Key>/usr/ssl/certkey.pem</Key>
  <CA>/usr/ssl/CA.pem</CA>
</Https>

<Https Listen="1.1.1.1:443">
  <Cert>/usr/ssl_ip_port/cert.pem</Cert>
  <Key>/usr/ssl_ip_port/certkey.pem</Key>
  <CA>/usr/ssl_ip_port/CA.pem</CA>
</Https>

<Https Listen="*:886">
  <Cert>/usr/ssl_port/cert.pem</Cert>
  <Key>/usr/ssl_port/certkey.pem</Key>
  <CA>/usr/ssl_port/CA.pem</CA>
</Https>
```

- <Https> Configures HTTPS.
 - <Cert> Server certificate.
 - <Key> The private key for server certification. Encrypted formatting not supported.
 - <CA> Certificate authority (CA) chain certificate.

Even if the same port is used, more specific expressions will take priority.

For example, if there are multiple NICs as in the example above, a client that connects using 1.1.1.1:443 will be given service using the second and more specific certificate (1.1.1.1:443), while a client that connects to 1.1.1.4:443 will be given service with the first and more general certificate (omitted, or *:443). If the certificate is overwritten with a file of the same name, the changes will be reflected upon reload.

Note: Only the PEM (Privacy Enhanced Mail) format is supported for certificates, and RSA for the asymmetric key algorithm.

3.6.2 SSL/TLS Acceleration

SSL/TLS can be accelerated using CPU (AES-NI). A CPU that supports AES-NI will prioritize the AES algorithm for SSL/TLS. If AES-NI is recognized, the following will be recorded in the Info.log file.

```
AES-NI : ON (SSL/TLS accelerated)
```

Administrators can select whether to use AES-NI or not.

```
# server.xml - <Server><Cache>

<AES-NI>ON</AES-NI>
```

- `<AES-NI>` (default: `ON`) Decides whether or not AES-NI is used.

3.6.3 CipherSuite Selection

The following CipherSuites are supported.

Cipher Suite	TLS1.2	TLS1.1/1.0	SSL3.0
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)	O		
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)	O		
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)	O	O	
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)	O	O	
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)	O		
TLS_RSA_WITH_AES_256_CBC_SHA256 (0x003d)	O		
TLS_RSA_WITH_AES_128_CBC_SHA256 (0x003c)	O		
TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)	O	O	
TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)	O	O	
TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)	O	O	
TLS_RSA_WITH_RC4_128_SHA (0x0005)			O
TLS_RSA_WITH_RC4_128_MD5 (0x0004)			O

The CipherSuite to be used can be configured in the CipherSuite property of `<Https>`.

```
# server.xml - <Server>

<Https CipherSuite="ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP">
  <Cert>/usr/ssl/cert.pem</Cert>
  <Key>/usr/ssl/certkey.pem</Key>
  <CA>/usr/ssl/CA.pem</CA>
</Https>
```

- CipherSuite Follows the [SSLCipherSuite Directive in Apache mod_ssl](#).

A higher level of security can be obtained by ensuring [forward secrecy](#) (refer to links below).

- [SSL Labs: Deploying Forward Secrecy](#)
- [SSL/TLS & Perfect Forward Secrecy](#)
- [Configuring Apache, Nginx, and OpenSSL for Forward Secrecy](#)

By default, a CipherSuite that ensures forward secrecy (FS) is prioritized.

```
# server.xml - <Server>

<Https FS="ON"> ... </Https>
```

- FS
 - `ON` (default) A CipherSuite that ensures forward secrecy is prioritized.
 - `OFF` Selects in the order specified by ClientHello.

The `FS` property takes priority over the `CipherSuite` property.

Note: Due to performance reasons, only ECDHE is supported. DHE is not supported.

3.6.4 Checking the CipherSuite

The results of configuring the `CipherSuite` can be checked. `CipherSuite` expression follows [OpenSSL 1.0.0E](#).

```
http://127.0.0.1:10040/monitoring/ssl?ciphersuite=...
```

The results are returned in JSON format.

```
{
  "version": "2.0.0",
  "method": "ssl",
  "status": "OK",
  "result":
  [
    {
      "Name" : "AES128-SHA",
      "Ver" : "SSLv3",
      "Kx" : "RSA",
      "Au" : "RSA",
      "Enc" : "AES(128)",
      "Mac" : "SHA1"
    },
    {
      "Name" : "AES256-SHA",
      "Ver" : "SSLv3",
      "Kx" : "RSA",
      "Au" : "RSA",
      "Enc" : "AES(256)",
      "Mac" : "SHA1"
    }
  ]
}
```

3.6.5 Multi-Domain Configuration

SSL configurations can cause problems when a single server runs multiple services simultaneously. Most Web/Cache servers decide which virtual host is used for the service by examining the `Host` header of the HTTP request.

Generally, the SSL identifies the server name (`winesoft.co.kr`) that the client (browser) tries to connect to using the certificate. However, if identification cannot be done with the certificate (e.g. the certificate is wrong or expired), then the user will be asked whether to trust the website or not (though there are cases when the site is blocked entirely). Though normal identification could not be done, SSL communication can still be established with the client's trust.

If there is only one virtual host on the server that uses SSL, no problems will occur. However, multiple virtual hosts running simultaneously on a server can cause problems. This is because when the server sends the certificate to the client ("2. Certificate Transmission" in "General HTTPS communication"), the server is unable to know which host the client is trying to reach.

The following show some typical solutions for this issue.

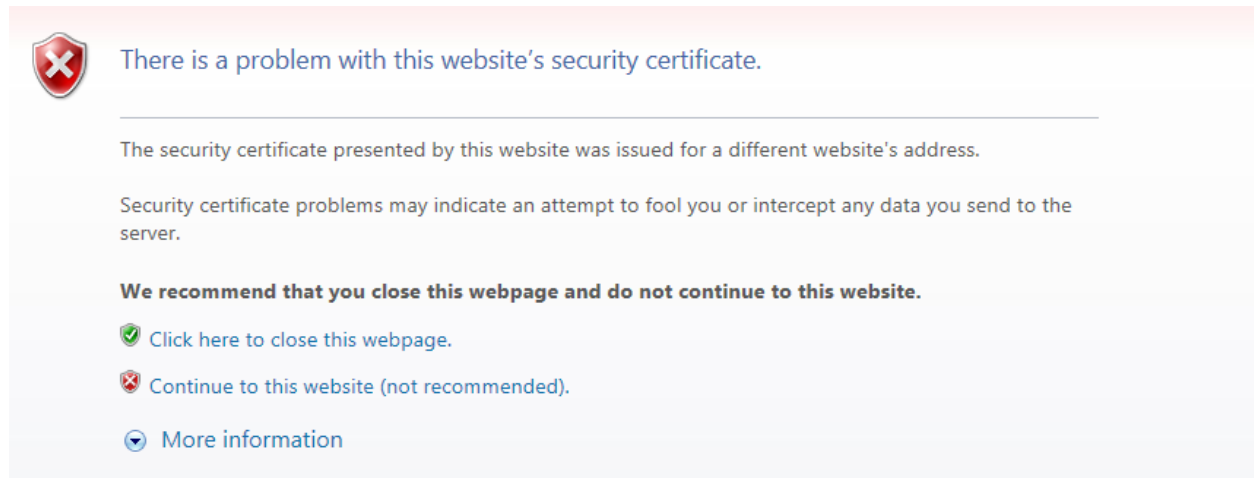


Fig. 3.15: General HTTPS communication.

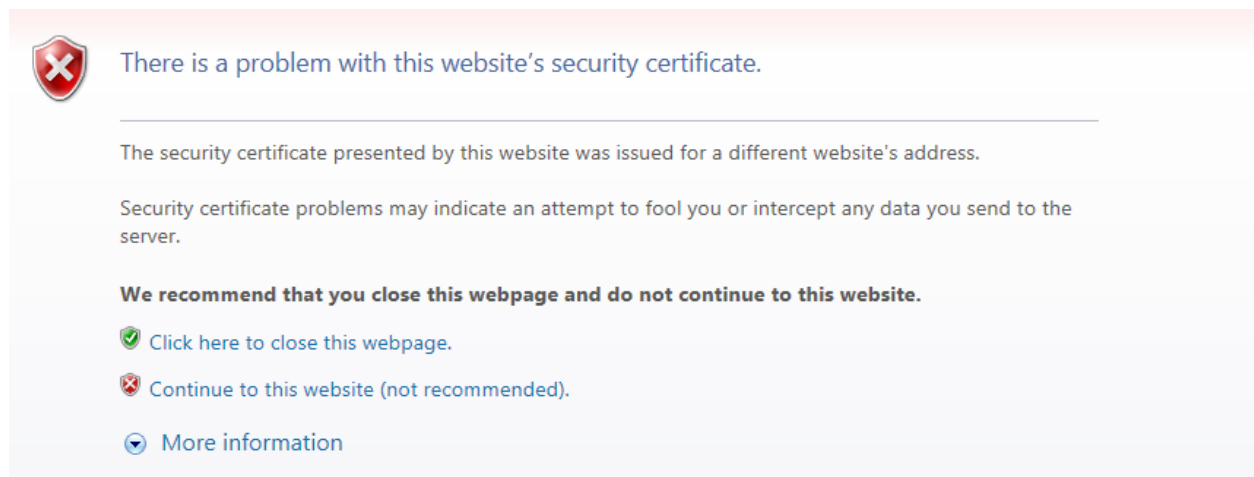


Fig. 3.16: It is left to the client to judge.

Method	Pros	Cons
SNI	Works with only server settings (standard)	Windows XP and IE6 unsupported
Multi Certificate	Works by replacing only the certificate	Main domain or service subject must be the same for faster reissuing
Multi Port	Works by changing only the port	HTTPS port must be specified on the web page
Multi NIC	Works with only server settings (most widely used)	Configuration of NIC and IP adding required

SNI (Server Name Indication)

This method makes use of the [SNI \(Server Name Indication\)](#) expansion field of SSL/TLS. This function is used by specifying the target virtual host like the Host header of an HTTP request when the client requests an SSL connection from the server. Though this is the most elegant solution, there can be some compatibility issues. The following lists the clients that do not support SNI (source: [Wikipedia - Server Name Indication](#)).

- Internet Explorer (any version) on Windows XP or Internet Explorer 6 or earlier
- Safari on Windows XP
- BlackBerry Browser
- Windows Mobile up to 6.5
- Android default browser on Android 2.x[34] (Fixed in Honeycomb for tablets and Ice Cream Sandwich for phones)
- wget before 1.14
- Java before 1.7

Realistically, SNI cannot be used, and STON does not support SNI.

Multi Certificate

This method places multiple domains or wildcards (i.e. *.winesoft.co.kr) in the certificate so that multiple domains can be identified with only one certificate.

```
[-] Extension (id-ce-subjectAltName)
  Extension Id: 2.5.29.17 (id-ce-subjectAltName)
  [-] GeneralNames: 3 items
    [-] GeneralName: dNSName (2)
      dNSName: *.gstatic.com
    [-] GeneralName: dNSName (2)
      dNSName: *.metric.gstatic.com
    [-] GeneralName: dNSName (2)
      dNSName: gstatic.com
```

Fig. 3.17: Multiple domains can be certified with one certificate.

This method is effective if the subjects of the service are the same, but if there is no relation, sharing the certificate is rather difficult. As this method can be used just by replacing the certificate, there is no need to configure anything in STON (see also [DigiCert](#)).

Multi Port

SSL uses port 443 by default. By setting up a port that does not overlap with the SSL port, multiple certificates can be installed. The port can be specified on the client to request an SSL connection.

```
https://winesoft.co.kr:543/
```

Multiple certificates can be set up in STON by specifying the ports in the Listen property, as shown below.

```
# server.xml - <Server>

<Https> ..Certificate A.. </Https>
<Https Listen="*:543"> ..Certificate B.. </Https>
<Https Listen="*:544"> ..Certificate C.. </Https>
```

Though this is the most economical solution, there is the problem of having to specify the HTTPS port on all webpage links.

Multi NIC

If the server has multiple NICs, a different IP can be assigned to each NIC. A different certificate will be installed for each server IP, and STON will be configured to choose the certificate based on the IP accessed by the client. In STON, the IP can be specified in the Listen property to allow multiple certificates to be configured.

```
# server.xml - <Server>

<Https Listen="10.10.10.10"> ..Certificate A.. </Https>
<Https Listen="10.10.10.11"> ..Certificate B.. </Https>
<Https Listen="10.10.10.12"> ..Certificate C.. </Https>
```

This method is the most widely used.

Note: If configurations are made public, problems can be caused by knowledge of IP addresses. In this case, NIC names can be used in place of IPs.

```
# server.xml - <Server>

<Https Listen="eth0"> ... </Https>
<Https Listen="eth1"> ... </Https>
<Https Listen="eth2"> ... </Https>
```

3.6.6 Enabling Security Protocol

The protocol can be established for each <Https>.

```
# server.xml - <Server>

<Https TLS1.2="ON" TLS1.1="ON" TLS1.0="ON" SSL3.0="ON"> ... </Https>
```

- TLS1.2 (default: ON) Uses TLS1.2.
- TLS1.1 (default: ON) Uses TLS1.1.
- TLS1.0 (default: ON) Uses TLS1.0.
- SSL3.0 (default: ON) Uses SSL3.0.

3.6.7 HSTS

HSTS (HTTP Strict Transport Security) can be easily set up using *Client Request/Response Header Modification*.

```
# /svc/www.example.com/headers.txt  
*, $RES[Strict-Transport-Security: max-age=31536000; includeSubDomains], set
```

The Qualys SSL Server Test only returns an A+ rating for sites with HSTS enabled.

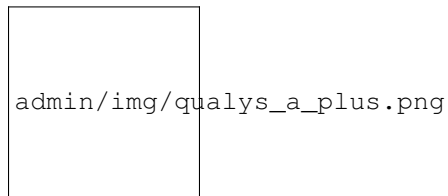


Fig. 3.18: STON can receive A+s starting from v2.2.

3. Advanced Features

4.1 Chapter 10. Monitoring & Statistics

This chapter will cover monitoring and statistics. While monitoring and statistics can be interpreted differently based on how they are used, they are similar in that they both use numbers to describe the system.

The most important feature is that it updates in real time. It is important to be able to view real-time status updates, and even a five-minute delay is too long. It is also important to know if the many policies are taking effect when they are applied. All statistics are collected every second.

Statistics are collected for each virtual host and are provided in real time (every second), with an average being provided every five minutes. The results are provided in JSON and XML formats for the users to analyze and process the results easily.

```
http://127.0.0.1:10040/monitoring/realtime?type=[JSON or XML]
http://127.0.0.1:10040/monitoring/average?type=[JSON or XML]
```

- **realtime** Displays the service status from one second ago.
- **average** Displays the average of five minutes of statistics.

4.1.1 Data Range

The range of data to be collected can be configured.

```
# server.xml - <Server><VHostDefault>
# vhosts.xml - <Vhosts><Vhost>

<Stats>
  <DirDepth>0</DirDepth>
  <DirDepthAccum>OFF</DirDepthAccum>
  <HttpsTraffic>OFF</HttpsTraffic>
  <ClientLocal>OFF</ClientLocal>
  <OriginLocal>OFF</OriginLocal>
</Stats>
```

- **<DirDepth> (default: 0)** Collects statistics for each directory. If set to zero, statistics will be collected in the root (/) directory. If set to one, statistics will be collected in directories one level down.

Note: Though there is no limit to the value that can be set, collecting statistics for too many directories can cause memory problems.

- **<DirDepthAccum>** Configures whether or not to accumulate statistics in the parent directory when collecting the statistics for each directory. If **<DirDepth>** is set to zero, this setting is ignored.
 - OFF (default) Statistics are not accumulated in the parent directory.
 - ON Statistics are accumulated in the parent directory.

For example, let's assume that **<DirDepth>** is set to two and all directories have ten lines of traffic. If **<DirDepthAccum>** is set to OFF, then statistics will be collected in each directory where traffic occurs, as shown in the left diagram. If set to ON, statistics from lower directories are accumulated in parent directories, as shown in the right diagram.

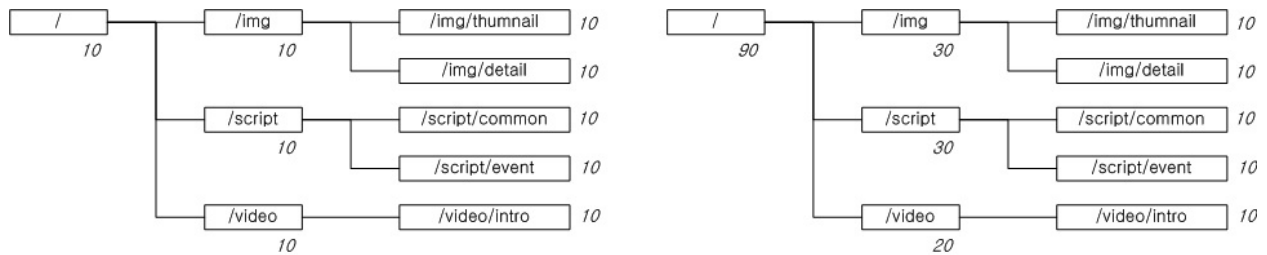


Fig. 4.1: Accumulated statistics in the parent directory.

In this example, the sum of the traffic in the `/img` directory and its subdirectories is 30, which is accumulated into the parent directory.

- **<HttpsTraffic>**
 - OFF (default) HTTPS traffic is only collected in SSL statistics.
 - ON HTTPS traffic is collected into both SSL and HTTP statistics.

Generally, traffic that passes through the SSL layer is stored separately as SSL statistics. However, HTTPS will be processed as HTTP in upper protocols, so more detailed statistics can be collected. Because SSL and HTTP statistics can overlap, it is recommended to only trust HTTP statistics.

- **<ClientLocal>** The traffic between the Loopback client and STON can be included in statistics.
 - OFF (default) Will not be included.
 - ON Will be included.
- **<OriginLocal>** The traffic between STON and the Loopback origin server will be included.
 - OFF (default) Will not be included.
 - ON Will be included.

4.1.2 Host Aggregate Statistics

Host statistics are the aggregate statistics of all the virtual hosts that run on the lowest level. The statistics can be provided in JSON and XML formats.

<pre>{ "Host": { "Version": "2.0.0", "Name": "localhost", "State": "Healthy", "Uptime": 155996, } }</pre>	<pre><Host Version="2.0.0" Name="localhost" State="Healthy" Uptime="155986" OriginSession="32" OriginActiveSession="20"</pre>
---	--

```

"OriginSession":33,
"OriginActiveSession":20,
"OriginInbound":688177,
"OriginOutbound":14184,
"OriginReqCount":62,
"OriginResTotalCount":62,
"OriginResTotalTimeRes":2375,
"OriginResTotalTimeComplete":2509,
"OriginRes2xxCount":54,
"OriginRes2xxTimeRes":2327,
"OriginRes2xxTimeComplete":2481,
"OriginRes3xxCount":8,
"OriginRes3xxTimeRes":2700,
"OriginRes3xxTimeComplete":2700,
"OriginRes4xxCount":0,
"OriginRes4xxTimeRes":0,
"OriginRes4xxTimeComplete":0,
"OriginRes5xxCount":0,
"OriginRes5xxTimeRes":0,
"OriginRes5xxTimeComplete":0,
"ClientSession":155,
"ClientActiveSession":80
"ClientInbound":35748,
"ClientOutbound":972906,
"ClientReqCount":152,
"ClientResTotalCount":152,
"ClientResTotalTimeRes":1411,
"ClientResTotalTimeComplete":1479,
"ClientRes2xxCount":93,
"ClientRes2xxTimeRes":2305,
"ClientRes2xxTimeComplete":2409,
"ClientRes3xxCount":59,
"ClientRes3xxTimeRes":3,
"ClientRes3xxTimeComplete":13,
"ClientRes4xxCount":0,
"ClientRes4xxTimeRes":0,
"ClientRes4xxTimeComplete":0,
"ClientRes5xxCount":0,
"ClientRes5xxTimeRes":0,
"ClientRes5xxTimeComplete":0,
"RequestHitRatio":6387,
"ByteHitRatio":2926,
"HttpCountSum" :
{
  "OriginReqCount" : 0,
  "OriginResTotalCount" : 0,
  "OriginRes2xxCount" : 0,
  "OriginRes3xxCount" : 0,
  "OriginRes4xxCount" : 0,
  "OriginRes5xxCount" : 0,
  "ClientReqCount" : 0,
  "ClientResTotalCount" : 0,
  "ClientRes2xxCount" : 0,
  "ClientRes3xxCount" : 0,
  "ClientRes4xxCount" : 0,
  "ClientRes5xxCount" : 0
},
"HttpRequestHitSum" :
OriginInbound="1140741"
OriginOutbound="10059"
OriginReqCount="42"
OriginResTotalCount="42"
OriginResTotalTimeRes="5071"
OriginResTotalTimeComplete="10288"
OriginRes2xxCount="19"
OriginRes2xxTimeRes="9989"
OriginRes2xxTimeComplete="21521"
OriginRes3xxCount="23"
OriginRes3xxTimeRes="1008"
OriginRes3xxTimeComplete="1008"
OriginRes4xxCount="0"
OriginRes4xxTimeRes="0"
OriginRes4xxTimeComplete="0"
OriginRes5xxCount="0"
OriginRes5xxTimeRes="0"
OriginRes5xxTimeComplete="0"
ClientSession="165"
ClientActiveSession="80"
ClientInbound="14792"
ClientOutbound="1981700"
ClientReqCount="64"
ClientResTotalCount="64"
ClientResTotalTimeRes="5535"
ClientResTotalTimeComplete="6840"
ClientRes2xxCount="44"
ClientRes2xxTimeRes="8050"
ClientRes2xxTimeComplete="9943"
ClientRes3xxCount="20"
ClientRes3xxTimeRes="5"
ClientRes3xxTimeComplete="15"
ClientRes4xxCount="0"
ClientRes4xxTimeRes="0"
ClientRes4xxTimeComplete="0"
ClientRes5xxCount="0"
ClientRes5xxTimeRes="0"
ClientRes5xxTimeComplete="0"
RequestHitRatio="6923"
ByteHitRatio="4243">
<HttpCountSum
  OriginReqCount="0"
  OriginResTotalCount="0"
  OriginRes2xxCount="0"
  OriginRes3xxCount="0"
  OriginRes4xxCount="0"
  OriginRes5xxCount="0"
  ClientReqCount="0"
  ClientResTotalCount="0"
  ClientRes2xxCount="0"
  ClientRes3xxCount="0"
  ClientRes4xxCount="0"
  ClientRes5xxCount="0"/>
<HttpRequestHitSum
  TCP_NONE="0"
  TCP_HIT="0"
  TCP_IMS_HIT="0"
  TCP_REFRESH_HIT="0"

```

```

{
    "TCP_NONE" : 0,
    "TCP_HIT" : 0,
    "TCP_IMS_HIT" : 0,
    "TCP_REFRESH_HIT" : 0,
    "TCP_REF_FAIL_HIT" : 0,
    "TCP_NEGATIVE_HIT" : 0,
    "TCP_REDIRECT_HIT" : 0,
    "TCP_MISS" : 0,
    "TCP_REFRESH_MISS" : 0,
    "TCP_CLIENT_REFRESH_MISS" : 0,
    "TCP_DENIED" : 0,
    "TCP_ERROR" : 0
},
"FileSystem":
{
    "RequestHitRatio":0,
    "ByteHitRatio":0,
    "Outbound":0,
    "Session":0
},
"System":{ ... },
"VirtualHost": [ ... ]
"View": [ ... ]
}
}
TCP_REF_FAIL_HIT="0"
TCP_NEGATIVE_HIT="0"
TCP_REDIRECT_HIT="0"
TCP_MISS="0"
TCP_REFRESH_MISS="0"
TCP_CLIENT_REFRESH_MISS="0"
TCP_DENIED="0"
TCP_ERROR="0"/>
<FileSystem>
<RequestHitRatio>0</RequestHitRatio>
<ByteHitRatio>0</ByteHitRatio>
<Outbound>0</Outbound>
<Session>0</Session>
</FileSystem>
<System> ... </System>
<VirtualHost> ... </VirtualHost>
<VirtualHost> ... </VirtualHost>
<VirtualHost> ... </VirtualHost>
<View> ... </View>
<View> ... </View>
</Host>

```

- Version STON version.
- Name The host name. If not defined, the system name will be used.
- State Service status. (Healthy=Normal service, Inactive=Inactive license, Emergency)
- Uptime (unit: seconds) The service running time.
- OriginSession The number of origin sessions.
- OriginActiveSession The number of transmitting origin sessions.
- OriginInbound (unit: bytes, average) The amount of data received from the origin server.
- OriginReqCount (average) The amount of requests to the origin server.
- OriginOutbound (unit: bytes, average) The amount of data transmitted to the origin server.
- OriginResTotalCount (average) The number of responses from the origin server.
- OriginResTotalTimeRes (unit: 0.01 ms, average) The origin server response time (from HTTP request to first HTTP response).
- OriginResTotalTimeComplete (unit: 0.01 ms, average) The origin server HTTP transaction completion time (from HTTP request to HTTP response completion).
- OriginRes2xxCount (average) The number of 2xx responses from the origin server.
- OriginRes2xxTimeRes (unit: 0.01 ms, average) The origin server 2xx response time.
- OriginRes2xxTimeComplete (unit: 0.01 ms, average) The origin server 2xx transaction completion time.
- OriginRes3xxCount (average) The number of 3xx responses from the origin server.
- OriginRes3xxTimeRes (unit: 0.01 ms, average) The origin server 3xx response time.

- `OriginRes3xxTimeComplete` (unit: 0.01ms, average) The origin server 3xx transaction completion time.
- `OriginRes4xxCount` (average) The number of 4xx responses from the origin server.
- `OriginRes4xxTimeRes` (unit: 0.01 ms, average) The origin server 4xx response time.
- `OriginRes4xxTimeComplete` (unit: 0.01ms, average) The origin server 4xx transaction completion time.
- `OriginRes5xxCount` (average) The number of 5xx responses from the origin server.
- `OriginRes5xxTimeRes` (unit: 0.01 ms, average) The origin server 5xx response time.
- `OriginRes5xxTimeComplete` (unit: 0.01 ms, average) The origin server 5xx transaction completion time.
- `ClientSession` The number of client sessions.
- `ClientActiveSession` The number of transmitting client sessions.
- `ClientInbound` (unit: bytes, average) The amount of inbound data from clients.
- `ClientOutbound` (unit: bytes, average) The amount of outbound data to clients.
- `ClientReqCount` (average) The number of client requests.
- `ClientResTotalCount` (average) The number of client responses.
- `ClientResTotalTimeRes` (unit: 0.01 ms, average) The client response time (from HTTP request to first HTTP response).
- `ClientResTotalTimeComplete` (unit: 0.01 ms, average) The client HTTP transaction completion time (from HTTP request to HTTP response completion).
- `ClientRes2xxCount` (average) The number of 2xx responses from the client.
- `ClientRes2xxTimeRes` (unit: 0.01 ms, average) The client 2xx response time.
- `ClientRes2xxTimeComplete` (unit: 0.01 ms, average) The client 2xx transaction completion time.
- `ClientRes3xxCount` (average) The number of 3xx responses from the client.
- `ClientRes3xxTimeRes` (unit: 0.01 ms, average) The client 3xx response time.
- `ClientRes3xxTimeComplete` (unit: 0.01 ms, average) The client 3xx transaction completion time.
- `ClientRes4xxCount` (average) The number of 4xx responses from the client.
- `ClientRes4xxTimeRes` (unit: 0.01 ms, average) The client 4xx response time.
- `ClientRes4xxTimeComplete` (unit: 0.01 ms, average) The client 4xx transaction completion time.
- `ClientRes5xxCount` (average) The number of 5xx responses from the client.
- `ClientRes5xxTimeRes` (unit: 0.01 ms, average) The client 5xx response time.
- `ClientRes5xxTimeComplete` (unit: 0.01 ms, average) The client 5xx transaction completion time.
- `RequestHitRatio` (unit: 0.01%, average) Hit ratio. If a caching object is created and the corresponding object is initialized, it is considered a hit. On the other hand, if the caching object is missing or the object is not initialized from the origin server, it does not count as a hit. The response code is unrelated to the hit ratio.

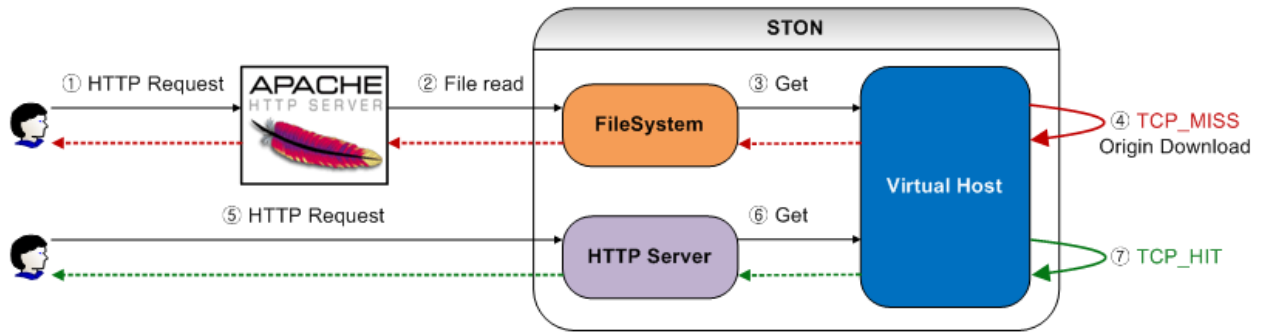


Fig. 4.2: HTTP and File I/O share the virtual host.

The RequestHitRatio of File I/O accessed via Apache will become 0%. The HTTP server, however, accesses cached files due to File I/O, making the RequestHitRatio 100%. ByteHitRatio is calculated as the ratio of the origin inbound to either HTTP outbound or File I/O outbound.

- ByteHitRatio (unit: 0.01%, average) The transfer ratio of the origin server to the client.

$$(\text{Client Outbound} - \text{Origin Server Inbound}) / \text{Client Outbound}$$

If the origin server is much faster or the client session closes quickly, then the total ratio can become a negative value.

- FileSystem Independent FileSystem statistics that do not count other stat values.
 - RequestHitRatio (unit: 0.01%, average) Hit ratio based on File I/O.
 - ByteHitRatio (unit: 0.01%, average) Transfer ratio of origin server to File I/O.
 - Outbound (unit: bytes, average) Size of data that goes through File I/O.
 - Session (average) The number of threads in the File I/O process.

Note: These statistics are only provided in five-minute increments.

- HttpCountSum The total number of HTTP transactions.
- HttpRequestHitSum The cache hit results.

4.1.3 System Statistics

The statistics of the system and global resources can be provided in JSON and XML formats.

"System":	<System>
{	<CPU
"CPU":	Kernel="689"
{	User="1316"
"Kernel":689,	Idle="7993"
"User":1316,	ProcKernel="570"
"Idle":7993,	ProcUser="1216"
"ProcKernel":570,	Nice="0"
"ProcUser":1216,	IOWait="52"
"Nice":0,	IRQ="10"
"IOWait":52,	SoftIRQ="12"
"IRQ":10,	Steal="0" />

```

    "SoftIRQ":12,
    "Steal":0
  },
  "Mem":
  {
    "Free":5914644,
    "STON":9785800
  },
  "Storage":
  {
    "Disk":
    [
      {
        "Path":"/cache1",
        "Status":"Normal",
        "Read":23,
        "ReadMerged":0,
        "ReadSectors":344,
        "ReadTime":117,
        "Write":24,
        "WriteMerged":93,
        "WriteSectors":936,
        "WriteTime":256,
        "IOProgress":0,
        "IOTime":173,
        "IOWeightedTime":373
      },
      {
        "Path":"/cache2",
        "Status":"Normal",
        "Read":27,
        "ReadMerged":1,
        "ReadSectors":488,
        "ReadTime":144,
        "Write":24,
        "WriteMerged":86,
        "WriteSectors":880,
        "WriteTime":254,
        "IOProgress":0,
        "IOTime":189,
        "IOWeightedTime":380
      }
    ]
  },
  "ServerSocket":
  {
    "Total":42,
    "Established":1,
    "Accepted":0,
    "Closed":0
  },
  "ClientSocket":
  {
    "Total":1,
    "Established":0,
    "Connected":0,
    "Closed":0
  },
  },
  <Mem Free="5914644" STON="9785800"/>
  <Storage>
    <Disk
      Path="/cache1"
      Status="Normal"
      Read="23"
      ReadMerged="0"
      ReadSectors="344"
      ReadTime="117"
      Write="24"
      WriteMerged="93"
      WriteSectors="936"
      WriteTime="256"
      IOProgress="0"
      IOTime="173"
      IOWeightedTime="373"/>
    <Disk
      Path="/cache2"
      Status="Normal"
      Read="27"
      ReadMerged="1"
      ReadSectors="488"
      ReadTime="144"
      Write="24"
      WriteMerged="86"
      WriteSectors="880"
      WriteTime="254"
      IOProgress="0"
      IOTime="189"
      IOWeightedTime="380"/>
    </Storage>
    <ServerSocket
      Total="42"
      Established="2"
      Accepted="1"
      Closed="0"/>
    <ClientSocket
      Total="1"
      Established="0"
      Connected="0"
      Closed="0"/>
    <TCPSocket
      Established="30"
      Timewait="2"
      Orphan="0"
      Alloc="0"
      Mem="20"/>
    <EQ>0</EQ>
    <RQ>1000000</RQ>
    <WaitingFiles2Write>0</WaitingFiles2Write>
    <ServiceAccess Allow="60" Deny="2"/>
    <SystemLoadAverage Min1="0" Min5="0" Min15="0"/>
    <URLRewrite>57</URLRewrite>
  </System>

```

```
"TCPSocket":
{
  "Established":30,
  "Timewait":2,
  "Orphan":0,
  "Alloc":0,
  "Mem":20
},
"EQ":0,
"RQ":1000000,
"WaitingFiles2Write":0,
"ServiceAccess":{"Allow":60, "Deny":2}
"SystemLoadAverage":
{
  "Min1":0,
  "Min5":0,
  "Min15":0
},
"URLRewrite":57
}
```

- CPU (unit: 0.01%) CPU usage. Total CPU usage can be calculated by Kernel + User.
 - Kernel CPU (Kernel) usage.
 - User CPU (User) usage.
 - Idle Idle CPU.
 - ProcKernel CPU (Kernel) usage by STON.
 - ProcUser CPU (User) usage by STON.
 - Nice Time spent running ‘niced’ user processes.
 - IOWait Time spent waiting for I/O to complete.
 - IRQ Time spent servicing interrupts.
 - SoftIRQ Time spent servicing software interrupts.
 - Steal Time spent while other CPUs are serviced.
- Mem (unit: bytes) Memory usage.
 - Free Size of free memory in the system.
 - STON Size of memory used by STON.
- Disk Disk performance stats.
 - Path Disk path.
 - Status Disk status (Normal: normal, Invalid: excluded due to failure, Unmounted: unmounted by administrator).
 - Read The number of successful reads.
 - ReadMerged The number of merged reads.
 - ReadSectors The number of read sectors.
 - ReadTime (unit: ms) The elapsed time per read.
 - Write The number of successful writes.

- WriteMerged The number of merged writes.
- WriteSectors The number of written sectors.
- WriteTime (unit: ms) The elapsed time per write.
- IOProgress The number of running I/Os.
- IOTime (unit: ms) The elapsed time per I/O.
- IOWeightedTime (unit: ms) The elapsed time per I/O (weight applied).
- ServerSocket Server socket (between client and STON) information.
 - Total The total number of server sockets.
 - Established The number of connected server sockets.
 - Accepted The number of newly connected server sockets.
 - Closed The number of closed server sockets.
- ClientSocket Client socket (between STON and the origin server) information.
 - Total The total number of client sockets.
 - Established The number of connected client sockets.
 - Connected The number of newly connected client sockets.
 - Closed The number of closed client sockets.
- TCPSocket TCP status information provided by the system (OS).
 - Established The number of established status TCP connections.
 - Timewait The number of TIME_WAIT status TCP connections.
 - Orphan The number of orphaned TCP connections (not attached to a file handle).
 - Alloc The number of allocated TCP sockets.
 - Mem The amount of memory used by TCP sockets.
- EQ The number of unprocessed events in the STON Framework.
- RQ The number of events saved in the recently serviced content reference queue.
- WaitingFiles2Write The number of disk write pending files.
- ServiceAccess The number of sockets allowed and denied by ServiceAccess.
- SystemLoadAverage The 1/5/15 minute average of the System Load Average.
- URLRewrite The number of successful conversions made by the URL preprocessor.

4.1.4 Virtual Host Statistics

Statistics are provided for each virtual host. There are four types of virtual host statistics: HTTP transfer (per directory), URL bypass, port bypass, and SSL.

<pre>"VirtualHost": [{ "Name": "image.11st.co.kr", "Uptime": 155966, "OriginSession": 12,</pre>	<pre><VirtualHost Name="image.11st.co.kr" Uptime="155956" OriginSession="12" OriginActiveSession="6" OriginInbound="106914"</pre>
--	--

```

"OriginActiveSession":6,
"OriginInbound":169,
"OriginOutbound":269,
"OriginReqCount":62,
"OriginResTotalCount":1,
"OriginResTotalTimeRes":3300,
"OriginResTotalTimeComplete":3300,
"OriginRes2xxCount":0,
"OriginRes2xxTimeRes":0,
"OriginRes2xxTimeComplete":0,
"OriginRes3xxCount":1,
"OriginRes3xxTimeRes":3300,
"OriginRes3xxTimeComplete":3300,
"OriginRes4xxCount":0,
"OriginRes4xxTimeRes":0,
"OriginRes4xxTimeComplete":0,
"OriginRes5xxCount":0,
"OriginRes5xxTimeRes":0,
"OriginRes5xxTimeComplete":0,
"ClientSession":26,
"ClientActiveSession":16,
"ClientInbound":13968,
"ClientOutbound":110398,
"ClientReqCount":152,
"ClientResTotalCount":52,
"ClientResTotalTimeRes":94,
"ClientResTotalTimeComplete":107,
"ClientRes2xxCount":1,
"ClientRes2xxTimeRes":4700,
"ClientRes2xxTimeComplete":4800,
"ClientRes3xxCount":51,
"ClientRes3xxTimeRes":3,
"ClientRes3xxTimeComplete":15,
"ClientRes4xxCount":0,
"ClientRes4xxTimeRes":0,
"ClientRes4xxTimeComplete":0,
"ClientRes5xxCount":0,
"ClientRes5xxTimeRes":0,
"ClientRes5xxTimeComplete":0,
"RequestHitRatio":10000,
"ByteHitRatio":9984,
"FileSystem":
{
  "RequestHitRatio":0,
  "ByteHitRatio":0,
  "Outbound":0,
  "Session":0
},
"Memory":785740769,
"SecuredMemory":0,
"Disk": { ... },
"Session": { ... },
"Dims": { ... },
"Compression": { ... },
"FileTotal":458308,
"FileOpened":15,
"FileInstance":458320,
"Cached": { ... },
OriginOutbound="3238"
OriginReqCount="42"
OriginResTotalCount="13"
OriginResTotalTimeRes="1553"
OriginResTotalTimeComplete="6630"
OriginRes2xxCount="1"
OriginRes2xxTimeRes="3300"
OriginRes2xxTimeComplete="69300"
OriginRes3xxCount="12"
OriginRes3xxTimeRes="1408"
OriginRes3xxTimeComplete="1408"
OriginRes4xxCount="0"
OriginRes4xxTimeRes="0"
OriginRes4xxTimeComplete="0"
OriginRes5xxCount="0"
OriginRes5xxTimeRes="0"
OriginRes5xxTimeComplete="0"
ClientSession="30"
ClientActiveSession="12"
ClientInbound="4113"
ClientOutbound="895937"
ClientReqCount="64"
ClientResTotalCount="18"
ClientResTotalTimeRes="666"
ClientResTotalTimeComplete="4377"
ClientRes2xxCount="10"
ClientRes2xxTimeRes="1200"
ClientRes2xxTimeComplete="7870"
ClientRes3xxCount="8"
ClientRes3xxTimeRes="0"
ClientRes3xxTimeComplete="12"
ClientRes4xxCount="0"
ClientRes4xxTimeRes="0"
ClientRes4xxTimeComplete="0"
ClientRes5xxCount="0"
ClientRes5xxTimeRes="0"
ClientRes5xxTimeComplete="0"
RequestHitRatio="10000"
ByteHitRatio="8806">
<FileSystem>
  <RequestHitRatio>0</RequestHitRatio>
  <ByteHitRatio>0</ByteHitRatio>
  <Outbound>0</Outbound>
  <Session>0</Session>
</FileSystem>
<Memory>784786700</Memory>.
<SecuredMemory>0</SecuredMemory>.
<Disk> ... </Disk>
<Session> ... </Session>
<Dims> ... </Dims>
<Compression> ... </Compression>
<File Total="458278" Opened="15" Instance="458292"/>
<Cached> ... </Cached>
<CacheFileEvent> ... </CacheFileEvent>
<WaitingFiles2Delete>1087593</WaitingFiles2Delete>
<CacheFileEvent Create="%u\" Swap="%u\" Erase="%u\"
<ClientHttpRequestBypass Sum="8100">27</ClientHttpRequestBypass>
<ClientHttpRequestDenied Sum="400">1</ClientHttpRequestDenied>

```

```

"CacheFileEvent": { ... },
"WaitingFiles2Delete":1087595,
"ClientHttpReqBypassSum":8100,
"ClientHttpReqBypass":27,
"ClientHttpReqDeniedSum":400,
"ClientHttpReqDenied":1,
"OriginTraffic": { ... },
"PortBypass": { ... },
"ClientTraffic": { ... },
"UrlBypass": { ... }

},
...
]

```

Note: The values will be the same as host statistics from Name to FileSystem.

- Memory (unit: bytes) The amount of content loaded into memory.
- SecuredMemory (unit: bytes) The amount of content deleted from memory.
- Disk Disk information.
- Session Session information.
- Dims DIMS conversion statistics.
- Compression Compression statistics.
- FileTotal The total number of files.
- FileOpened The number of opened local files.
- FileInstance The number of caching files.
- Cached Caching information.
- CacheFileEvent A caching file event.
- WaitingFiles2Delete The number of files pending deletion.
- ClientHttpReqBypass The number of bypassed client HTTP requests.
- ClientHttpReqDenied The number of denied HTTP requests.
- OriginTraffic Origin server traffic statistics.
- PortBypass Port bypass traffic statistics.
- ClientTraffic Client traffic statistics.
- UrlBypass HTTP traffic statistics bypassed to the origin server via URL matching or <BypassNoCacheRequest>.

Note: These statistics are only provided in five-minute increments.

- ClientHttpReqBypassSum The total number of bypassed HTTP requests.
 - ClientHttpReqDeniedSum The total number of denied HTTP requests.
-

Disk Statistics

Provides disk statistics used by virtual hosts.

```

"Disk":
{
  "TotalSize":22004057982,
  "Create":0,
  "Open":1,
  "Delete":0,
  "ReadCount":1,
  "ReadSize":104744,
  "WriteCount":0,
  "WriteSize":0,
  "Distribution":
  {
    "U1K="45725,
    "U2K="192523,
    "U4K="137055,
    "U8K="39740,
    "U16K="13408,
    "U32K="12303,
    "U64K="11462,
    "U128K="2560,
    "U256K="22,
    "U512K="0,
    "U1M="45725,
    "U2M="192523,
    "U4M="137055,
    "U8M="39740,
    "U16M="13408,
    "U32M="12303,
    "U64M="11462,
    "U128M="2560,
    "U256M="22,
    "U512M="0,
    "U1G="0,
    "U2G="0,
    "U4G="0,
    "U8G="0,
    "U16G":0,
    "O16G":0
  }
}

```

```

<Disk>
  <TotalSize>22003701435</TotalSize>
  <Create>1</Create>
  <Open>10</Open>
  <Delete>0</Delete>
  <ReadCount>9</ReadCount>
  <ReadSize>735726</ReadSize>
  <WriteCount>1</WriteCount>
  <WriteSize>157145</WriteSize>
  <Distribution
    U1K="45725"
    U2K="192523"
    U4K="137055"
    U8K="39740"
    U16K="13408"
    U32K="12303"
    U64K="11462"
    U128K="2560"
    U256K="22"
    U512K="0"
    U1M="45725"
    U2M="192523"
    U4M="137055"
    U8M="39740"
    U16M="13408"
    U32M="12303"
    U64M="11462"
    U128M="2560"
    U256M="22"
    U512M="0"
    U1G="0"
    U2G="0"
    U4G="0"
    U8G="0"
    U16G="0"
    O16G="0" />
  </Disk>

```

- **TotalSize** (unit: bytes) The total size of local files.
- **Create** The number of created local files.
- **Open** The number of opened local files.
- **Delete** The number of deleted local files.
- **ReadCount** The number of times a local file is read.
- **ReadSize** (unit: bytes) The total size of read local files.
- **WriteCount** The number of times a local file is written.
- **WriteSize** (unit: bytes) The total size of written local files.

- **Distribution** Distribution of local files based on size.

- **U1K** The number of files under 1 KB.
- **U2K** The number of files under 2 KB.
- **U4K** The number of files under 4 KB.
- **U8K** The number of files under 8 KB.
- **U16K** The number of files under 16 KB.
- **U32K** The number of files under 32 KB.
- **U64K** The number of files under 64 KB.
- **U128K** The number of files under 128 KB.
- **U256K** The number of files under 256 KB.
- **U512K** The number of files under 512 KB.
- **U1M** The number of files under 1 MB.
- **U2M** The number of files under 2 MB.
- **U4M** The number of files under 4 MB.
- **U8M** The number of files under 8 MB.
- **U16M** The number of files under 16 MB.
- **U32M** The number of files under 32 MB.
- **U64M** The number of files under 64 MB.
- **U128M** The number of files under 128 MB.
- **U256M** The number of files under 256 MB.
- **U512M** The number of files under 512 MB.
- **U1G** The number of files under 1 GB.
- **U2G** The number of files under 2 GB.
- **U4G** The number of files under 4 GB.
- **U8G** The number of files under 8 GB.
- **U16G** The number of files under 16 GB.
- **O16G** The number of files over 16 GB.

Session Statistics

Provides the session statistics.

<pre>"Session": { "Client":30, "ActiveClient":20, "Origin":12, "ActiveOrigin":7 },</pre>	<pre><Session Client="30" ActiveClient="20" Origin="12" ActiveOrigin="7" /></pre>
--	---

- **Client** The number of total HTTP client sessions.

- **ActiveClient** The number of transmitting sessions among HTTP clients.
- **Origin** The number of total origin server sessions.
- **ActiveOrigin** The number of transmitting sessions among origin server sessions.

DIMS Statistics

Provides DIMS performance statistics.

"Dims":	<Dims
{	Requests="30"
"Requests": 30,	Converted="29"
"Converted": 29,	Failed="1"
"Failed": 1,	AvgSrcSize="1457969"
"AvgSrcSize": 1457969,	AvgDestSize="598831"
"AvgDestSize": 598831,	AvgTime="34" />
"AvgTime": 34	
},	

- **Requests** The number of conversion requests.
- **Converted** The number of conversion successes.
- **Failed** The number of conversion failures.
- **AvgSrcSize** (unit: bytes) The average size of source images.
- **AvgDestSize** (unit: bytes) The average size of converted images.
- **AvgTime** (unit: ms) The elapsed time for conversion.

Compression Statistics

Provides compression performance statistics.

"Compression":	<Compression
{	Requests="30"
"Requests": 30,	Converted="29"
"Converted": 29,	Failed="1"
"Failed": 1,	AvgSrcSize="1457969"
"AvgSrcSize": 1457969,	AvgDestSize="598831"
"AvgDestSize": 598831,	AvgTime="34" />
"AvgTime": 34	
},	

- **Requests** The number of compression requests.
- **Converted** The number of compression statistics.
- **Failed** The number of compression failures.
- **AvgSrcSize** (unit: bytes) The average size of source files.
- **AvgDestSize** (unit: bytes) The average size of compressed files.
- **AvgTime** (unit: ms) The elapsed time for compression.

Origin Server Statistics

Provides statistics for the traffic between STON and the origin server.

```

"OriginTraffic":
{
  "HttpReqCountSum":0,
  "HttpReqCount":0,
  "HttpReqHeaderSize":269,
  "HttpReqBodySize":0,
  "HttpResHeaderSize":169,
  "HttpResBodySize":0,
  "Response":
  {
    "ResTotal":
    {
      "CountSum":0,
      "Count":1,
      "CompletedSum":0,
      "Completed":1,
      "TimeRes":3300,
      "TimeComplete":3300
    },
    "Res2xx":
    {
      "CountSum":0,
      "Count":0,
      "CompletedSum":0,
      "Completed":0,
      "TimeRes":0,
      "TimeComplete":0
    },
    "Res3xx":
    {
      "CountSum":0,
      "Count":1,
      "CompletedSum":0,
      "Completed":1,
      "TimeRes":3300,
      "TimeComplete":3300
    },
    "Res4xx":
    {
      "CountSum":0,
      "Count":0,
      "CompletedSum":0,
      "Completed":0,
      "TimeRes":0,
      "TimeComplete":0
    },
    "Res5xx":
    {
      "CountSum":0,
      "Count":0,
      "CompletedSum":0,
      "Completed":0,
      "TimeRes":0,
      "TimeComplete":0
    }
  }
}

<OriginTraffic>
  <HttpReqCount Sum="600">2</HttpReqCount>
  <HttpReqHeaderSize>3238</HttpReqHeaderSize>
  <HttpReqBodySize>0</HttpReqBodySize>
  <HttpResHeaderSize>2020</HttpResHeaderSize>
  <HttpResBodySize>104894</HttpResBodySize>
  <Response>
    <ResTotal>
      <Count Sum="8100">13</Count>
      <Completed Sum="8100">12</Completed>
      <TimeRes>1553</TimeRes>
      <TimeComplete>6630</TimeComplete>
    </ResTotal>
    <Res2xx>
      <Count Sum="8100">1</Count>
      <Completed Sum="8100">1</Completed>
      <TimeRes>3300</TimeRes>
      <TimeComplete>69300</TimeComplete>
    </Res2xx>
    <Res3xx>
      <Count Sum="8100">12</Count>
      <Completed Sum="8100">11</Completed>
      <TimeRes>1408</TimeRes>
      <TimeComplete>1408</TimeComplete>
    </Res3xx>
    <Res4xx>
      <Count Sum="8100">0</Count>
      <Completed Sum="8100">0</Completed>
      <TimeRes>0</TimeRes>
      <TimeComplete>0</TimeComplete>
    </Res4xx>
    <Res5xx>
      <Count Sum="8100">0</Count>
      <Completed Sum="8100">0</Completed>
      <TimeRes>0</TimeRes>
      <TimeComplete>0</TimeComplete>
    </Res5xx>
    <ConnectTimeout Sum="8100">0</ConnectTimeout>
    <ReceiveTimeout Sum="8100">0</ReceiveTimeout>
    <Close Sum="8100">0</Close>
  </Response>
  <Connect>
    <Count>0</Count>
    <AvgDNSQueryTime>0</AvgDNSQueryTime>
    <AvgConnTime>0</AvgConnTime>
  </Connect>
</OriginTraffic>

```

```
"ConnectTimeoutSum":0,
"ConnectTimeout":0,
"ReceiveTimeoutSum":0,
"ReceiveTimeout":0,
"CloseSum":0,
"Close":0
},
"Connect":
{
  "Count":0,
  "AvgDNSQueryTime":0,
  "AvgConnTime":0
}
},
```

- **HttpReqCount** The number of HTTP requests sent to the origin server.
- **HttpReqHeaderSize** (unit: bytes) The size of the HTTP header sent to the origin server.
- **HttpReqBodySize** (unit: bytes) The size of the HTTP body sent to the origin server.
- **HttpResHeaderSize** (unit: bytes) The size of the HTTP header received by the origin server.
- **HttpResBodySize** (unit: bytes) The size of the HTTP body received by the origin server.
- **Response** The responses from the origin server.
 - **ResXXX** The statistics for the type of response (2xx, 3xx, 4xx, 5xx, total).
 - * **Count** The number of responses.
 - * **Completed** The number of properly transferred HTTP transactions.
 - * **TimeRes** The HTTP response time.
 - * **TimeComplete** The completion time for HTTP transactions.
 - **ConnectTimeout** The number of connection failures.
 - **ReceiveTimeout** The number of transmission delays.
 - **Close** The number of times the origin server closes the socket during transmissions.
- **Connect** Origin server connection statistics.
 - **Count** The number of connections.
 - **AvgDNSQueryTime** (unit: 0.01 ms) The average DNS query time.
 - **AvgConnTime** (unit: 0.01 ms) The average connection time (from TCP SYN transmission to TCP SYN ACK reception).

Note: These statistics are only provided in five-minute increments.

- **HttpReqCountSum** The total number of HTTP requests.
 - **CountSum** The total number of HTTP responses.
 - **CompletedSum** The total number of completed HTTP transactions.
 - **ConnectTimeoutSum** The total number of origin server connection failures.
 - **ReceiveTimeoutSum** The total number of origin server transmission delays.
 - **CloseSum** The total number of connections closed by the origin server.
-

Port Bypass Statistics

Provides statistics on traffic from <PortBypass>

```

"PortBypass":
[
  {
    "SrcPort":1935, "DestPort":1935, "Session":0,
    "Hit":
    {
      "Established":0, "ClientClosed":0,
      "OriginClosed":0, "OriginCT":0
    },
    "ClientTraffic": { "In":0, "Out":0 },
    "OriginTraffic": { "In":0, "Out":0 }
  },
  {
    "SrcPort":1936, "DestPort":1936, "Session":17,
    ...
  }
],
  <PortBypass SrcPort="1935" DestPost="1935">
    <Session>0</Session>
    <Hit Established="0"
      ClientClosed="0"
      OriginClosed="0"
      OriginCT="0" />
    <ClientTraffic In="0" Out="0"/>
    <OriginTraffic In="0" Out="0"/>
  </PortBypass>
  <PortBypass SrcPort="1936" DestPost="1936">
    <Session>17</Session>
    ...
  </PortBypass>

```

- **SrcPort/DestPort** The bypassed STON/origin server port.
- **Session** The number of currently connected sessions.
- **Hit** Bypass connection statistics.
 - **Established** The number of established connections.
 - **ClientClosed** The number of connections closed by clients.
 - **OriginClosed** The number of connections closed by the origin server.
 - **OriginCT** The number of origin server connection failures.
- **ClientTraffic** (unit: bytes) Client traffic (In=Inbound, Out=Outbound).
- **OriginTraffic** (unit: bytes) Origin server traffic (In=Inbound, Out=Outbound).

Client Statistics

Client traffic can be portrayed in **Traffic** in multiple ways depending on the statistics configuration for each directory. If directory statistics have not been configured, all traffic will be counted as the root (/) directory. If they have been configured, only the root directory and directories with traffic will be displayed.

```

"ClientTraffic":
{
  "Depth":0,
  "Accum":"OFF",
  "HttpsTraffic":"OFF",
  "TrafficCount":1,
  "Traffic":
  [
    {
      "RequestHitRatio" : 9984,
      "Path":"/",
      "HttpReqCountSum":0,
      "HttpReqCount":100,
      "HttpReqHeaderSize":13968,
      "HttpReqBodySize":0,
      <ClientTraffics Depth="0" Accum="OFF" HttpsTraffic="OFF">
        <TrafficCount>1</TrafficCount>
        <Traffic RequestHitRatio="0">
          <Path>/</Path>
          <HttpReqCount Sum="0">0</HttpReqCount>
          <HttpReqHeaderSize>4113</HttpReqHeaderSize>
          <HttpReqBodySize>0</HttpReqBodySize>
          <HttpResHeaderSize>3066</HttpResHeaderSize>
          <HttpResBodySize>892871</HttpResBodySize>
          <Response>
            <ResTotal>
              <Count Sum="0">18</Count>
              <Completed Sum="0">18</Completed>
              <TimeRes>666</TimeRes>
              <TimeComplete>4377</TimeComplete>
            </ResTotal>
          </Response>
        </Traffic>
      </ClientTraffics>
    }
  ]
}

```

```

"HttpResHeaderSize":5654,
"HttpResBodySize":104744,
"Response":
{
  "ResTotal":
  {
    "CountSum":0,
    "Count":52,
    "CompletedSum":0,
    "Completed":52,
    "TimeRes":94,
    "TimeComplete":107
  },
  "Res2xx":
  {
    "CountSum":0,
    "Count":1,
    "CompletedSum":0,
    "Completed":1,
    "TimeRes":4700,
    "TimeComplete":4800
  },
  "Res3xx":
  {
    "CountSum":0,
    "Count":51,
    "CompletedSum":0,
    "Completed":51,
    "TimeRes":3,
    "TimeComplete":15
  },
  "Res4xx":
  {
    "CountSum":0,
    "Count":0,
    "CompletedSum":0,
    "Completed":0,
    "TimeRes":0,
    "TimeComplete":0
  },
  "Res5xx":
  {
    "CountSum":0,
    "Count":0,
    "CompletedSum":0,
    "Completed":0,
    "TimeRes":0,
    "TimeComplete":0
  }
},
"SSL":
{
  "RecvSize":0,
  "SendSize":0
},
"RequestHit":
{
  "TCP_NONE":0,

```

```

</ResTotal>
<Res2xx>
  <Count Sum="0">10</Count>
  <Completed Sum="0">10</Completed>
  <TimeRes>1200</TimeRes>
  <TimeComplete>7870</TimeComplete>
</Res2xx>
<Res3xx>
  <Count Sum="0">8</Count>
  <Completed Sum="0">8</Completed>
  <TimeRes>0</TimeRes>
  <TimeComplete>12</TimeComplete>
</Res3xx>
<Res4xx>
  <Count Sum="0">0</Count>
  <Completed Sum="0">0</Completed>
  <TimeRes>0</TimeRes>
  <TimeComplete>0</TimeComplete>
</Res4xx>
<Res5xx>
  <Count Sum="0">0</Count>
  <Completed Sum="0">0</Completed>
  <TimeRes>0</TimeRes>
  <TimeComplete>0</TimeComplete>
</Res5xx>
</Response>
<SSL RecvSize="0" SendSize="0"/>
<RequestHit
  TCP_NONE="0"
  TCP_HIT="0"
  TCP_IMS_HIT="0"
  TCP_REFRESH_HIT="0"
  TCP_REF_FAIL_HIT="0"
  TCP_NEGATIVE_HIT="0"
  TCP_REDIRECT_HIT="0"
  TCP_MISS="0"
  TCP_REFRESH_MISS="0"
  TCP_CLIENT_REFRESH_MISS="0"
  TCP_DENIED="0"
  TCP_ERROR="0"/>
<RequestHitSum
  TCP_NONE="0"
  TCP_HIT="0"
  TCP_IMS_HIT="0"
  TCP_REFRESH_HIT="0"
  TCP_REF_FAIL_HIT="0"
  TCP_NEGATIVE_HIT="0"
  TCP_REDIRECT_HIT="0"
  TCP_MISS="0"
  TCP_REFRESH_MISS="0"
  TCP_CLIENT_REFRESH_MISS="0"
  TCP_DENIED="0"
  TCP_ERROR="0"/>
</Traffic>
<FileSystem>
  <GetAttr
    TimeRes="0"
    FileCount="0"

```

```

    "TCP_HIT":0,
    "TCP_IMS_HIT":0,
    "TCP_REFRESH_HIT":0,
    "TCP_REF_FAIL_HIT":0,
    "TCP_NEGATIVE_HIT":0,
    "TCP_REDIRECT_HIT":0,
    "TCP_MISS":0,
    "TCP_REFRESH_MISS":0,
    "TCP_CLIENT_REFRESH_MISS":0,
    "TCP_DENIED":0,
    "TCP_ERROR":0
  },
  "RequestHitSum":
  {
    "TCP_NONE":0,
    "TCP_HIT":0,
    "TCP_IMS_HIT":0,
    "TCP_REFRESH_HIT":0,
    "TCP_REF_FAIL_HIT":0,
    "TCP_NEGATIVE_HIT":0,
    "TCP_REDIRECT_HIT":0,
    "TCP_MISS":0,
    "TCP_REFRESH_MISS":0,
    "TCP_CLIENT_REFRESH_MISS":0,
    "TCP_DENIED":0,
    "TCP_ERROR":0
  },
  "FileSystem":
  {
    "GetAttr" :
    {
      "TimeRes" : 0,
      "FileCount" : 0,
      "DirCount" : 0,
      "FailCount" : 0,
      "TotalCount" : 0
    },
    "Open" :
    {
      "TimeRes" : 0,
      "Count" : 0
    },
    "Read" :
    {
      "TimeRes" : 0,
      "BufferSize" : 0,
      "BufferFilled" : 0,
      "Count" : 0
    },
    "RequestHit":
    {
      "TCP_NONE":0,
      "TCP_HIT":0,
      "TCP_IMS_HIT":0,
      "TCP_REFRESH_HIT":0,
      "TCP_REF_FAIL_HIT":0,
      "TCP_NEGATIVE_HIT":0,
      "TCP_REDIRECT_HIT":0,
      "DirCount"="0"
      FailCount="0"></GetAttr>
    <Open TimeRes="0"></Open>
    <Read
      TimeRes="0"
      BufferSize="0"
      BufferFilled="0"></Read>
    <RequestHit
      TCP_NONE="0"
      TCP_HIT="0"
      TCP_IMS_HIT="0"
      TCP_REFRESH_HIT="0"
      TCP_REF_FAIL_HIT="0"
      TCP_NEGATIVE_HIT="0"
      TCP_REDIRECT_HIT="0"
      TCP_MISS="0"
      TCP_REFRESH_MISS="0"
      TCP_CLIENT_REFRESH_MISS="0"
      TCP_DENIED="0"
      TCP_ERROR="0"/>
    <RequestHitSum
      TCP_NONE="0"
      TCP_HIT="0"
      TCP_IMS_HIT="0"
      TCP_REFRESH_HIT="0"
      TCP_REF_FAIL_HIT="0"
      TCP_NEGATIVE_HIT="0"
      TCP_REDIRECT_HIT="0"
      TCP_MISS="0"
      TCP_REFRESH_MISS="0"
      TCP_CLIENT_REFRESH_MISS="0"
      TCP_DENIED="0"
      TCP_ERROR="0"/>
    </FileSystem>
  </ClientTraffics>

```

```
        "TCP_MISS":0,
        "TCP_REFRESH_MISS":0,
        "TCP_CLIENT_REFRESH_MISS":0,
        "TCP_DENIED":0,
        "TCP_ERROR":0
    },
    "RequestHitSum":
    {
        "TCP_NONE":0,
        "TCP_HIT":0,
        "TCP_IMS_HIT":0,
        "TCP_REFRESH_HIT":0,
        "TCP_REF_FAIL_HIT":0,
        "TCP_NEGATIVE_HIT":0,
        "TCP_REDIRECT_HIT":0,
        "TCP_MISS":0,
        "TCP_REFRESH_MISS":0,
        "TCP_CLIENT_REFRESH_MISS":0,
        "TCP_DENIED":0,
        "TCP_ERROR":0
    }
    }
}
]
```

- **Depth** The directory depth for statistics to be collected.
- **Accum** Whether or not directory statistics are accumulated in parent directories.
- **HttpsTraffic** Whether or not HTTPS traffic is included in HTTP traffic.
- **TrafficCount** The aggregated traffic count.
- **Traffic** Statistics for each directory. The root (/) always has traffic.
 - **Path** The service directory.
 - **HttpReqCount** (unit: bytes) The number of HTTP requests sent by clients.
 - **HttpReqHeaderSize** (unit: bytes) The size of HTTP request headers sent by clients.
 - **HttpReqBodySize** (unit: bytes) The size of HTTP request bodies sent by clients.
 - **HttpResHeaderSize** (unit: bytes) The size of HTTP response headers sent by STON.
 - **HttpResBodySize** (unit: bytes) The size of HTTP response bodies sent by STON.
 - **Response** Responses sent by STON.
 - * **Count** The number of responses.
 - * **Completed** The number of properly completed HTTP transactions.
 - * **TimeRes** The HTTP response time.
 - * **TimeComplete** The HTTP transaction completion time.
- **SSL** (unit: bytes) HTTPS traffic (RecvSize=received size, SendSize=transmitted size).
- **RequestHit** The cache HIT result.
- **FileSystem** FileSystem access.

- GetAttr The getattr function call count and response time (FileCount: File response, DirCount: Dir response, FailCount: failure response).
- Open The open function call count and response time.
- Read The read function call count, response time, requested size (BufferSize), and response size (Buffer-Filled).
- RequestHit (File I/O access) The cache HIT result.

Note: These statistics are only provided in five-minute increments.

- HttpReqCountSum The total number of HTTP requests.
 - CountSum The total number of HTTP responses.
 - CompletedSum The total number of completed HTTP transactions.
 - RequestHitSum The cache HIT result.
-

4.1.5 View

View is a method that ties multiple virtual hosts into one to extract statistics. The concept came from viewing multiple tables as one table in a database. As shown below, the setup is very simple.

```
# vhosts.xml

<Vhosts>
  <Vhost> ... </Vhost>
  <Vhost> ... </Vhost>
  ... (omitted) ...
  <View Name="SK">
    <Vhost>...</Vhost>
    <Vhost>...</Vhost>
  </View>
  <View Name="KT">
    <Vhost>...</Vhost>
    <Vhost>...</Vhost>
    <Vhost>...</Vhost>
  </View>
  <View Name="LG">
    <Vhost>...</Vhost>
    <Vhost>...</Vhost>
  </View>
</Vhosts>
```

View can even be set up with virtual hosts that don't exist. The following are the formats that View provides statistics in.

```
- Realtime XML/JSON
- SNMP - cache(1.3.6.1.4.1.40001.1.4).10 ~ 12
```

Let's explore an example where View can be used. Say there are three administrators running communities for their favorite sports: baseball.com, basketball.com, and football.com.

```
# vhosts.xml

<Vhosts>
```

```
<Vhost Name="baseball.com"> ... </Vhost>
<Vhost Name="basketball.com"> ... </Vhost>
<Vhost Name="football.com"> ... </Vhost>
</Vhosts>
```

They decide to come together to open a combined sports community service, choosing `sports.com` as the domain name to encompass all the different sports. The objectives that must be met by the development/management team are as follows.

- The combined service must be provided through `sports.com`.
- The existing domains and services must be maintained for the existing users.
- The development teams must be combined. The management teams must be combined.
- Only the home page should be developed, connecting to existing services via links.

To realistically meet these demands, the development team decides to specify the existing domains as part of the first directory, as shown below.

```
# Existing services
http://baseball.com/standing/list.html
http://basketball.com/stats/2014/view.html
http://football.com/player/messi.php

# Combined service
http://sports.com/baseball/standing/list.html
http://sports.com/basketball/stats/2014/view.html
http://sports.com/football/player/messi.php
```

This can easily be configured using the URL preprocessor.

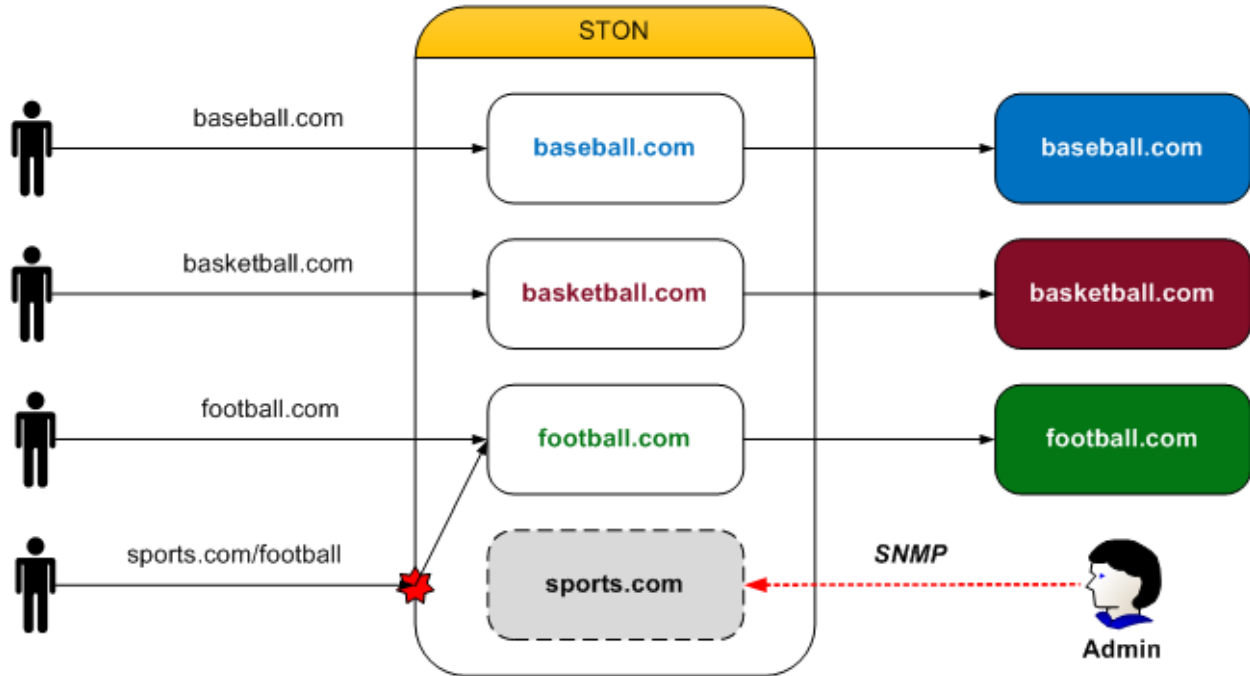
```
# vhosts.xml

<Vhosts>
  <Vhost Name="baseball.com"> ... </Vhost>
  <Vhost Name="basketball.com"> ... </Vhost>
  <Vhost Name="football.com"> ... </Vhost>
  <URLRewrite>
    <Pattern>sports.com/(.*)/(.*)</Pattern>
    <Replace>#1.com/#2</Replace>
  </URLRewrite>
</Vhosts>
```

The newly merged management team must now monitor not only their individual services but also the combined service (e.g. traffic, session, response codes). Most administrators familiar with SNMP will set up View to obtain these combined statistics.

```
# vhosts.xml

<Vhosts>
  <Vhost Name="baseball.com"> ... </Vhost>
  <Vhost Name="basketball.com"> ... </Vhost>
  <Vhost Name="football.com"> ... </Vhost>
  <URLRewrite>
    <Pattern>sports.com/(.*)/(.*)</Pattern>
    <Replace>#1.com/#2</Replace>
  </URLRewrite>
  <View Name="sports.com">
    <Vhost>baseball.com</Vhost>
    <Vhost>basketball.com</Vhost>
```



```
<Vhost>football.com</Vhost>
</View>
</Vhosts>
```

As seen in the above example, the combination of URL Rewrite and View can effectively tie existing sites together into a single service.

View Statistics

Provides statistics identical to the virtual host statistics, with the only difference being the names of the tags, as shown below.

```
"View":
[
  { ... },
  { ... },
]
<View ...>
...
</View>
<View> ... </View>
<View> ... </View>
```

4.1.6 Checking the Virtual Host List

The virtual host list can be checked.

```
http://127.0.0.1:10040/monitoring/vhostslist
```

The results are returned in JSON format.

```
{
  "version": "2.0.0",
  "method": "vhostslist",
  "status": "OK",
}
```

```
"result": [ "www.example.com", "www.winesoft.com", "site1.com" ]
}
```

4.1.7 Caching Information

The status of files being cached can be monitored. Generally, files can be distinguished by URLs, but if the same URL can have different options (e.g. Accept-Encoding), then there may also be multiple files.

```
http://127.0.0.1:10040/monitoring/fileinfo?url=example.com/sample.dat
```

The results are returned in JSON format. The following is the result of looking up the information of a /sample.dat file.

```
{
  "version": "2.0.0",
  "method": "fileinfo",
  "status": "OK",
  "result":
  [
    {
      "URI": "/sample.dat",
      "Accept-Encoding": "N",
      "RefCount": 0,
      "Disk-Index": 0,
      "Size": 2100267,
      "FID": 24267,
      "LocalPath": "/cache1/example.com/000i/q3.bin",
      "File-Opened": "N",
      "File-Updating": "-",
      "Downloader-Count": "0",
      "LastAccess": "[ 2012.09.03 14:29:50, -2 ]",
      "UpdateTime": "[ 2012.09.03 13:53:43, -2169 ]",
      "TTL-Left": "[ 2012.10.03 13:53:43, 2589831 ]",
      "ResponseCode": 200,
      "ContentType": "text/plain",
      "LastModifiedTime": "[ 2010.11.22 20:31:47, -56224685 ]",
      "ExpireTime": "[ 0, 0 ]",
      "CacheControl": "not-specified",
      "ETag": "502dd614:200c2b",
      "CustomTTL": 0,
      "NoMoreExist": "N",
      "LocalFileExist": "Y",
      "SmallFile": "N",
      "State": "Cached",
      "Deleted": "N",
      "AddedSize": "Y",
      "TransferEncoding": "N",
      "Compression": "-",
      "Purge": "N",
      "Ignore-IMS": "N",
      "Redirect-Location": "-",
      "Content-Disposition": "-",
      "NoCache": "N"
    }
  ]
}
```

- `URI` The file URI.
- `Accept-Encoding` (“Y” or “N”) “Y” if Accept-Encoding is supported.
- `RefCount` The file reference count.
- `Size (bytes)` The file size.
- `Disk-Index` (starts from 0) The saved disk index.
- `FID` The file ID.
- `LocalPath` The local path.
- `File-Opened` (“Y” or “N”) “Y” if a local file is opened.
- `File-Updating` Specifies the pointer to the updated object if a file is being updated.
- `Downloader-Count` The number of sessions downloading this file from the origin server.
- `LastAccess` (last accessed time, last accessed time - current time) [2012.09.03 14:29:50, -2] would mean that the file was last accessed 2 seconds before the current time, on 2012.09.03 14:29:50.
- `UpdateTime` (modified time, modified time - current time) The last time the file was modified. A 304 Not Modified response also updates the time.
- `TTL-Left` (expiration time, expiration time - current time) The time left until the content expires. The value is positive if there is still TTL left, and negative if already expired.
- `ResponseCode` The origin server response code.
- `ContentType` The MIME Type.
- `LastModifiedTime` (Last Modified Time, Last Modified Time - current time) The Last Modified Time sent by the origin server. If the origin server did not send this value, it will return zero.
- `ExpireTime` (Expire Time, Expire Time - current time) The Expire Time sent by the origin server. If the origin server did not send this value, it will return zero.
- `CacheControl` (“no-cache” or “not-specified” or an integer) The Cache-Control value sent by the origin server.
- `ETag` The ETag created by STON.
- `CustomTTL` Custom TTL. If not configured, zero is returned.
- `NoMoreExist` (“Y” or “N”) “Y” if file is pending deletion.
- `LocalFileExist` (“Y” or “N”) “Y” if the file exists locally (files not 200 OK are always “Y”).
- `SmallFile` (“Y” or “N”) “Y” if the file is considered a small file (for development purposes).
- `State` (“Not Init” or “Cached” or “Error”) The file status.
- `Deleted` (“Y” or “N”) “Y” if the file is deleted (for development purposes).
- `AddedSize` (“Y” or “N”) “Y” if the size is reflected in the statistics (for development purposes).
- `TransferEncoding` (“Y” or “N”) “Y” if Transfer-Encoding is supported.
- `Compression` The compression method.
- `Purge` (“Y” or “N”) “Y” if purged.
- `Ignore-IMS` (“Y” or “N”) “Y” if not configured to send an If-Modified-Since header during updates.
- `Redirect-Location` The Location header value.
- `Content-Disposition` The Content-Disposition header value.

- NoCache (“Y” or “N”) “Y” if the origin server responds with no-cache.

4.1.8 Log Trace

Receives the log in real time while it’s being recorded. Access, Origin, and Monitoring logs must specify a virtual host.

```
http://127.0.0.1:10040/monitoring/logtrace/info
http://127.0.0.1:10040/monitoring/logtrace/deny
http://127.0.0.1:10040/monitoring/logtrace/sys
http://127.0.0.1:10040/monitoring/logtrace/originerror
http://127.0.0.1:10040/monitoring/logtrace/access?vhost=www.site1.com
http://127.0.0.1:10040/monitoring/logtrace/origin?vhost=www.site1.com
http://127.0.0.1:10040/monitoring/logtrace/monitoring?vhost=www.site1.com
```

4.2 Chapter 11. SNMP

This chapter will explain the Simple Network Management Protocol (SNMP). All values in *Chapter 10. Monitoring & Statistics* can also be provided by SNMP. Moreover, SNMP can provide more subdivided time units and more detailed system status information. Real-time statistics and statistics averaged over a number of minutes (up to 60) per each virtual host can be provided.

- No additional package is required.
- SNMP is not run separately.
- Supports SNMP v1 and v2c.

4.2.1 Variables

Values that can be changed by a configuration or intentionally by the administrator will be specified as [variable name]. For example, if multiple disks exist, each disk will be represented by a number, assigned in order starting from one. This variable will be labeled as [diskIndex].

- **[diskIndex]** Stands for the disks configured in storage.

```
# server.xml - <Server><Cache>

<Storage>
  <Disk>/cache1</Disk>
  <Disk>/cache2</Disk>
  <Disk>/cache3</Disk>
</Storage>
```

In the above environment with three configured disks, /cache1 has a [diskIndex] of 1 while /cache3 has a [diskIndex] of 3. For example, the OID that refers to the entire volume of /cache1 is system.diskInfo.diskInfoTotalSize.1 (1.3.6.1.4.1.40001.1.2.18.1.3.1). The last .1 refers to the first disk.

- **[vhostIndex]** Automatically assigned when virtual hosts are loaded.

```
# vhosts.xml

<Vhosts>
  <Vhost Status="Active" Name="kim.com"> ... </Vhost>
  <Vhost Status="Active" Name="lee.com"> ... </Vhost>
```

```
<Vhost Status="Active" Name="park.com" StaticIndex="10300"> ... </Vhost>
</Vhosts>
```

In the example above, the first three virtual hosts loaded will be assigned a `[vhostIndex]` in order starting from 1. The virtual host will remember this `[vhostIndex]`, and the indexes will not change even if virtual hosts are deleted. If virtual host deletion and loading take place at the same time, deletion will occur first, and the new new loaded virtual host will be assigned the empty `[vhostIndex]`.

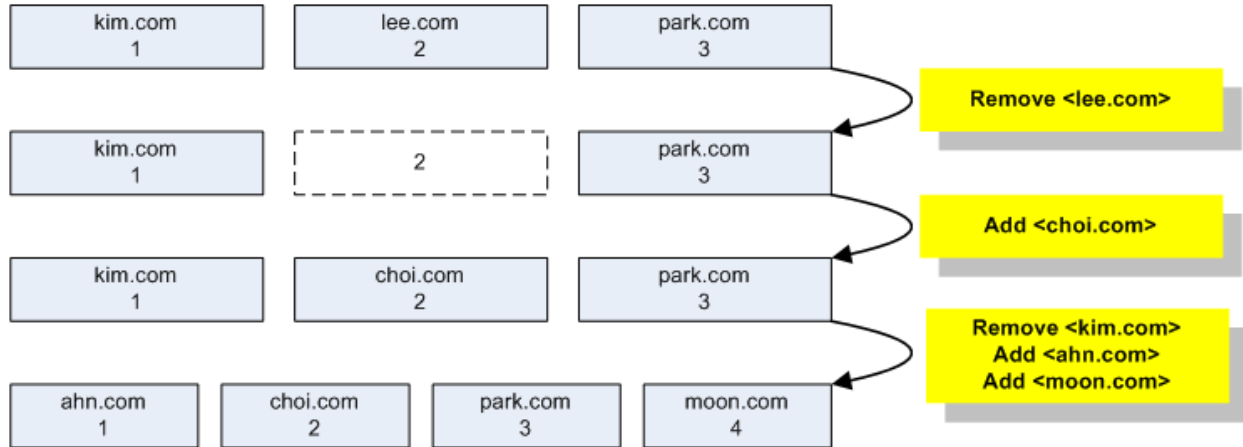


Fig. 4.3: Behavior of `[vhostIndex]`

- **[diskMin], [vhostMin]** Refers to time in minutes. A value of 5 refers to an average over the last 5 minutes, while a value of 60 refers to an average over the last 60 minutes. This value has a range of 0 to 60, where 0 refers to real-time (1 second) data.

SNMP uses table structures for items with values that can change dynamically. For example, the number of values in “total disk size” can change with the number of disks, so a table is used. STON provides virtual host statistics in minute increments, allowing for more complex expressions such as “`[vhostMin] . [vhostIndex]`”.

This expression allows you to request statistics for each virtual host at your desired time increment, but because there are two variables, it is hard to represent in a table. This problem can be solved by setting a default value for `[vhostMin]` for `SNMPWalk` to run.

4.2.2 Activation

SNMP behavior and ACL can be configured in global settings (`server.xml`).

```
# server.xml - <Server><Host>

<SNMP Port="161" Status="Inactive">
  <Allow>192.168.5.1</Allow>
  <Allow>192.168.6.0/24</Allow>
</SNMP>
```

- **<SNMP>** SNMP behavior can be configured using these properties.
 - **Port** (default: 161) SNMP service port.
 - **Status** (default: Inactive) Activates SNMP when set to Active.
- **<Allow>** Configures IP addresses that allow SNMP access. Designated IP, designated IP range, bitmask, and subnet forms are supported. If the connected socket is not one of the approved IPs, no response is returned.

4.2.3 Virtual Host/View Variables

The number of virtual host/View variables and default time (minutes) provided by SNMP can be configured.

```
# server.xml - <Server><Host>

<SNMP VHostCount=0, VHostMin=5 ViewCount=0, ViewMin=5 />
```

- **VHostCount** (default: 0) If set to 0, only existing virtual hosts respond. If greater than 0, all configured virtual hosts will respond whether they exist or not.
- **ViewCount** (default: 0) Applies to View. (Same as VHostCount)
- **VHostMin** (default: 5 min, maximum: 60 min) Configures the value of [vhostMin]. Has a range of 0 to 60, with 0 resulting in real-time data and 1~60 resulting in an average over the set amount of time.
- **ViewMin** (default: 0) Applies to View. (Same as VHostMin)

In an example with three configured virtual hosts, SNMPWalk's behavior can differ.

- If VHostCount=0

```
SNMPv2-SMI::enterprises.40001.1.4.2.1.2.1 = STRING: "web.winesoft.co.kr"
SNMPv2-SMI::enterprises.40001.1.4.2.1.2.2 = STRING: "img.winesoft.co.kr"
SNMPv2-SMI::enterprises.40001.1.4.2.1.2.3 = STRING: "vod.winesoft.co.kr"
```

- If VHostCount=5

```
SNMPv2-SMI::enterprises.40001.1.4.2.1.2.1 = STRING: "web.winesoft.co.kr"
SNMPv2-SMI::enterprises.40001.1.4.2.1.2.2 = STRING: "img.winesoft.co.kr"
SNMPv2-SMI::enterprises.40001.1.4.2.1.2.3 = STRING: "vod.winesoft.co.kr"
SNMPv2-SMI::enterprises.40001.1.4.2.1.2.4 = ""
SNMPv2-SMI::enterprises.40001.1.4.2.1.2.5 = ""
```

Other Variables

Other variables can be configured.

```
# server.xml - <Server><Host>

<SNMP GlobalMin="5" DiskMin="5" ConfCount="10" />
```

- **GlobalMin** (default: 5 min, maximum: 60 min) Sets the value of [globalMin].
- **DiskMin** (default: 5 min, maximum: 60 min) Sets the value of [diskMin].
- **ConfCount** (default: 10) Browses previous configuration lists. Has a range of 1 to 100. 1 will browse only the current configuration, and 2 will browse the current and previous configurations. Therefore, a value of 100 allows you to browse the current configuration and the past 99 configurations.

4.2.4 Community

Community can be configured to allow/deny access to the given OIDs.

```
# server.xml - <Server><Host>

<SNMP UnregisteredCommunity="Allow">
  <Community Name="example1" OID="Allow">
```

```

<OID>1.3.6.1.4.1.40001.1.4.1</OID>
<OID>1.3.6.1.4.1.40001.1.4.2</OID>
<OID>1.3.6.1.4.1.40001.1.4.4</OID>
</Community>
<Community Name="example2" OID="Deny">
  <OID>1.3.6.1.4.1.40001.1.4.3.1.11.11.10.1-61</OID>
</Community>
</SNMP>

```

If the `UnregisteredCommunity` value in `<SNMP>` is set to “Deny”, unregistered Community requests will be blocked.

- `<Community>` Configures Community.
 - Name The Community name.
 - OID (default: Allow) Configures the access of the `<OID>` tags below. If set to Allow, only the `<OID>` tags below will be allowed access. If set to Deny, the `<OID>` tags below will be denied access.

Specific OID (1.3.6.1.4.1.40001.1.4.4) and ranged OID (1.3.6.1.4.1.40001.1.4.3.1.11.11.10.1-61) formats are supported. When OIDs are allowed/denied, all OIDs set below will be configured the same way.

4.2.5 meta

```
OID = 1.3.6.1.4.1.40001.1.1
```

Provides meta information.

OID	Name	Type	Description
.1	manufacture	String	“WineSOFT Inc.”
.2	software	String	“STON”
.3	version	String	Version
.4	hostname	String	Host Name
.5	state	String	“Healthy” or “Inactive” or “Emergency”
.6	uptime	Integer	Runtime (seconds)
.7	admin	String	<code><Admin> ... </Admin></code>
.10	Conf	OID	Conf expansion

meta.conf

```
OID = 1.3.6.1.4.1.40001.1.1.10
```

`[confIndex]` is set in the `ConfCount` property of `<SNMP>`. A `[confIndex]` of 1 always refers to the current configuration values, while 2 refers to the previous configuration values. If `[confIndex]` is 10, the ninth past configuration values will be returned.

OID	Name	Type	Description
.1. [confIndex]	ID	Integer	Configuration ID
.2. [confIndex]	Time	Integer	Configuration time (Unix time)
.3. [confIndex]	Type	Integer	Configuration type (0 = Unknown, 1 = STON start, 2 = /conf/reload, 3 = /conf/upload, 4 = /conf/restore)
.4. [confIndex]	Size	Integer	Configuration file size
.5. [confIndex]	Hash	String	Configuration file Hash string
.6. [confIndex]	Path	String	Saved path of the configuration file
.7. [confIndex]	Ver	String	STON version of the configuration

4.2.6 system

OID = 1.3.6.1.4.1.40001.1.2

Provides information about the system running STON. The [sysMin] variable can be set from 0~60 minutes, allowing either real-time or averaged data. The [sysMin] in SNMPWalk can also be set to 0 to provide current information.

OID	Name	Type	Description
.1. [sysMin]	cpuTotal	Integer	Total CPU usage (100%)
.2. [sysMin]			Total CPU usage (10000%)
.3. [sysMin]	cpuKernel	Integer	CPU(Kernel) usage (100%)
.4. [sysMin]			CPU(Kernel) usage (10000%)
.5. [sysMin]	cpuUser	Integer	CPU(User) usage (100%)
.6. [sysMin]			CPU(User) usage (10000%)
.7. [sysMin]	cpuIdle	Integer	CPU(Idle) usage (100%)
.8. [sysMin]			CPU(Idle) usage (10000%)
.9	memTotal	Integer	Total system memory (KB)
.10. [sysMin]	memUse	Integer	Used system memory (KB)
.11. [sysMin]	memFree	Integer	Free system memory (KB)
.12. [sysMin]	memSTON	Integer	Memory used by STON (KB)
.13. [sysMin]	memUseRatio	Integer	System memory usage (100%)
.14. [sysMin]			System memory usage (10000%)
.15. [sysMin]	memSTONRatio	Integer	STON memory usage (100%)
.16. [sysMin]			STON memory usage (10000%)
.17	diskCount	Integer	Disk count
.18.1	diskInfo	OID	diskInfo expansion
.19.1	diskPerf	OID	diskPerf expansion
.20. [sysMin]	cpuProcKernel	Integer	Usage of CPU(Kernel) by STON (100%)
.21. [sysMin]			Usage of CPU(Kernel) by STON (10000%)
.22. [sysMin]	cpuProcUser	Integer	Usage of CPU(User) by STON (100%)
.23. [sysMin]			Usage of CPU(User) by STON (10000%)
.24. [sysMin]	sysLoadAverage	Integer	Load Average of 1 minute (0.01)
.25. [sysMin]			Load Average of 5 minutes (0.01)
.26. [sysMin]			Load Average of 15 minutes (0.01)

Continued on next page

Table 4.1 – continued from previous page

OID	Name	Type	Description
.27. [sysMin]	cpuNice	Integer	CPU(Nice) (100%)
.28. [sysMin]			CPU(Nice) (10000%)
.29. [sysMin]	cpuIOWait	Integer	CPU(IOWait) (100%)
.30. [sysMin]			CPU(IOWait) (10000%)
.31. [sysMin]	cpuIRQ	Integer	CPU(IRQ) (100%)
.32. [sysMin]			CPU(IRQ) (10000%)
.33. [sysMin]	cpuSoftIRQ	Integer	CPU(SoftIRQ) (100%)
.34. [sysMin]			CPU(SoftIRQ) (10000%)
.35. [sysMin]	cpuSteal	Integer	CPU(Steal) (100%)
.36. [sysMin]		Integer	CPU(Steal) (10000%)
.40. [sysMin]	TCPsocket.Established. [globalMin]	Integer	Number of established TCP connections
.41. [sysMin]	TCPsocket.Timewait. [globalMin]	Integer	Number of TIME_WAIT TCP connections
.42. [sysMin]	TCPsocket.Orphan. [globalMin]	Integer	Number of orphaned TCP connections
.43. [sysMin]	TCPsocket.Alloc. [globalMin]	Integer	Number of allocated TCP connections
.44. [sysMin]	TCPsocket.Mem. [globalMin]	Integer	TCP connection memory usage

system.diskInfo

OID = 1.3.6.1.4.1.40001.1.2.18.1

Provides disk information.

OID	Name	Type	Description
.2. [diskIndex]	diskInfoPath	String	Disk path
.3. [diskIndex]	diskInfoTotalSize	Integer	Total disk size (MB)
.4. [diskIndex]	diskInfoUseSize	Integer	Disk usage (MB)
.5. [diskIndex]	diskInfoFreeSize	Integer	Free disk size (MB)
.6. [diskIndex]	diskInfoUseRatio	Integer	Disk usage ratio (100%)
.7. [diskIndex]			Disk usage ratio (10000%)
.8. [diskIndex]	diskInfoStatus	String	“Normal” or “Invalid” or “Unmounted”

system.diskPerf

OID = 1.3.6.1.4.1.40001.1.2.19.1

Provides disk performance status.

OID	Name	Type	Description
.2. [diskMin] . [diskIndex]	diskPerfReadCount	Integer	Successful Read count
.3. [diskMin] . [diskIndex]	diskPerfReadMerged- Count	Integer	Merged Read count
.4. [diskMin] . [diskIndex]	diskPerfReadSec- torsCount	Integer	Read sectors count
.5. [diskMin] . [diskIndex]	diskPerfReadTime	Integer	Elapsed Read time (ms)
.6. [diskMin] . [diskIndex]	diskPerfWriteCount	Integer	Successful Write count
.7. [diskMin] . [diskIndex]	diskPerfWriteMerged- Count	Integer	Merged Write count
.8. [diskMin] . [diskIndex]	diskPerfWriteSec- torsCount	Integer	Written sectors count
.9. [diskMin] . [diskIndex]	diskPerfWriteTime	Integer	Elapsed Write time (ms)
.10. [diskMin] . [diskIndex]	diskPerfIOProgressCount	Integer	Number of IO in progress
.11. [diskMin] . [diskIndex]	diskPerfIOTime	Integer	Elapsed IO time (ms)
.12. [diskMin] . [diskIndex]	diskPerfIOTimeWeighted	Integer	Elapsed IO time (ms, weighted values)

4.2.7 global

OID = 1.3.6.1.4.1.40001.1.3

Provides resource information (e.g. sockets, events) shared by all modules.

- **ServerSocket** The client-STON connection. This socket is used by STON to process client requests.
- **ClientSocket** The STON-origin server connection. This socket is used by STON to send requests to the origin server.

OID	Name	Type	Description
.5	EQ. [globalMin]	Integer	The number of unprocessed Events in the STON Framework
.6	RQ. [globalMin]	Integer	The number of Events saved in the recently serviced content reference queue
.7	waitingFiles2Write. [globalMin]	Integer	The number of write pending files
.10	ServerSocket.Total. [globalMin]	Integer	Total number of server sockets
.11	ServerSocket.Established. [globalMin]	Integer	Total number of connected server sockets
.12	ServerSocket.Accepted. [globalMin]	Integer	Total number of newly connected server sockets
.13	ServerSocket.Closed. [globalMin]	Integer	The number of closed server sockets
.20	ClientSocket.Total. [globalMin]	Integer	Total number of client sockets
.21	ClientSocket.Established. [globalMin]	Integer	Total number of connected client sockets
.22	ClientSocket.Accepted. [globalMin]	Integer	Total number of newly connected client sockets
.23	ClientSocket.Closed. [globalMin]	Integer	The number of closed client sockets
.30	ServiceAccess.Allow. [globalMin]	Integer	The number of allowed(Allow) sockets by ServiceAccess
.31	ServiceAccess.Deny. [globalMin]	Integer	The number of denied(Deny) sockets by ServiceAccess

4.2.8 cache

OID = 1.3.6.1.4.1.40001.1.4

Cache service statistics are collected and provided in detail for each virtual host.

OID	Name	Type	Description
.1	host	OID	Host (expansion)
.2	vhostCount	Integer	The number of virtual hosts
.3.1	vhost	OID	Statistics for each virtual host
.4	vhostIndexMax	Integer	Max value of [vhostIndex]. SNMPWalk works based on this value.
.10	viewCount	Integer	View count
.11.1	view	OID	Stats per View
.12	viewIndexMax	Integer	Max value of [viewIndex]. SNMPWalk works based on this value.

cache.host

OID = 1.3.6.1.4.1.40001.1.4.1

Provides information of all virtual hosts.

OID	Name	Type	Description
.2	name	String	Host name
.3	status	String	“Healthy” or “Inactive”
.4	uptime	Integer	STON runtime (seconds)
.10	contents	OID	Content information (expansion)
.11	traffic	OID	Stats (expansion)

cache.host.contents

OID = 1.3.6.1.4.1.40001.1.4.1.10

Provides statistics for content in the service for all virtual hosts.

OID	Name	Type	Description
.1	memory	Integer	Memory caching size (KB)
.2	filesTotalCount	Integer	The number of files in service
.3	filesTotalSize	Integer	Total size of files in service (MB)
.10	filesCountU1KB	Integer	The number of files smaller than 1KB
.11	filesCountU2KB	Integer	The number of files smaller than 2KB
.12	filesCountU4KB	Integer	The number of files smaller than 4KB
.13	filesCountU8KB	Integer	The number of files smaller than 8KB
.14	filesCountU16KB	Integer	The number of files smaller than 16KB
.15	filesCountU32KB	Integer	The number of files smaller than 32KB
.16	filesCountU64KB	Integer	The number of files smaller than 64KB
.17	filesCountU128KB	Integer	The number of files smaller than 128KB
.18	filesCountU256KB	Integer	The number of files smaller than 256KB
.19	filesCountU512KB	Integer	The number of files smaller than 512KB
.20	filesCountU1MB	Integer	The number of files smaller than 1MB
.21	filesCountU2MB	Integer	The number of files smaller than 2MB
.22	filesCountU4MB	Integer	The number of files smaller than 4MB
.23	filesCountU8MB	Integer	The number of files smaller than 8MB
.24	filesCountU16MB	Integer	The number of files smaller than 16MB
.25	filesCountU32MB	Integer	The number of files smaller than 32MB
.26	filesCountU64MB	Integer	The number of files smaller than 64MB
.27	filesCountU128MB	Integer	The number of files smaller than 128MB
.28	filesCountU256MB	Integer	The number of files smaller than 256MB
.29	filesCountU512MB	Integer	The number of files smaller than 512MB
.30	filesCountU1GB	Integer	The number of files smaller than 1GB
.31	filesCountU2GB	Integer	The number of files smaller than 2GB
.32	filesCountU4GB	Integer	The number of files smaller than 4GB
.33	filesCountU8GB	Integer	The number of files smaller than 8GB
.34	filesCountU16GB	Integer	The number of files smaller than 16GB
.35	filesCountO16GB	Integer	The number of files larger than 16GB

cache.host.traffic

OID = 1.3.6.1.4.1.40001.1.4.1.11

Provides cache service and traffic statistics for all virtual hosts. Traffic statistics are provided as an average of up to 60 minutes. If the time value is omitted or set to 0, statistics will be provided in real time.

OID	Name	Type	Description
.1. [vhostMin]	requestHitRatio	Integer	Request Hit Ratio (100%)
.2. [vhostMin]			Request Hit Ratio (10000%)
.3. [vhostMin]	bytesHitRatio	Integer	Bytes Hit Ratio (100%)
.4. [vhostMin]			Bytes Hit Ratio (10000%)
.10	origin	OID	Origin traffic information (expansion)
.11	client	OID	Client traffic information (expansion)

cache.host.traffic.origin

OID = 1.3.6.1.4.1.40001.1.4.1.11.10

Provides origin server traffic statistics. Origin server traffic is divided into HTTP traffic and port bypass traffic.

OID	Name	Type	Description
.1. [vhostMin]	inbound	Integer	Average traffic received from the origin server (bytes)
.2. [vhostMin]	outbound	Integer	Average traffic sent to the origin server (bytes)
.3. [vhostMin]	sessionAverage	Integer	Average origin server session count
.4. [vhostMin]	activesessionAverage	Integer	Average origin server transmitting session count
.10	http	OID	Origin server HTTP traffic information
.10.1. [vhostMin]	http.inbound	Integer	Average HTTP traffic received from the origin server (bytes)
.10.2. [vhostMin]	http.outbound	Integer	Average HTTP traffic sent to the origin server (bytes)
.10.3. [vhostMin]	http.sessionAverage	Integer	Average origin server HTTP session count
.10.4. [vhostMin]	http.reqHeaderSize	Integer	Average HTTP Header traffic sent to the origin server (bytes)
.10.5. [vhostMin]	http.reqBodySize	Integer	Average HTTP Body traffic sent to the origin server (bytes)
.10.6. [vhostMin]	http.resHeaderSize	Integer	Average HTTP Header traffic received from the origin server (bytes)
.10.7. [vhostMin]	http.resBodySize	Integer	Average HTTP Body traffic received from the origin server (bytes)
.10.8. [vhostMin]	http.reqAverage	Integer	Average number of HTTP requests sent to the origin server
.10.9. [vhostMin]	http.reqCount	Integer	Total number of HTTP requests sent to the origin server
.10.10. [vhostMin]	http.resTotalAverage	Integer	Average number of all HTTP responses received from the origin server
.10.11. [vhostMin]	http.resTotalCompleteAverage	Integer	Average number of successful HTTP transactions from the origin server
.10.12. [vhostMin]	http.resTotalTimeRes	Integer	Average elapsed time to receive a response header from the origin server
.10.13. [vhostMin]	http.resTotalTimeComplete	Integer	Average completion time of HTTP transactions from the origin server
.10.14. [vhostMin]	http.resTotalCount	Integer	Total number of all HTTP responses received from the origin server
.10.15. [vhostMin]	http.resTotalCompleteCount	Integer	Total number of successful HTTP transactions from the origin server
.10.20. [vhostMin]	http.res2xxAverage	Integer	Number of 2xx responses from the origin server
.10.21. [vhostMin]	http.res2xxCompleteAverage	Integer	Number of successful 2xx transactions from the origin server
.10.22. [vhostMin]	http.res2xxTimeRes	Integer	Average elapsed time to receive a 2xx header from the origin server
.10.23. [vhostMin]	http.res2xxTimeComplete	Integer	Average completion time of 2xx transactions from the origin server
.10.24. [vhostMin]	http.res2xxCount	Integer	Total number of 2xx responses from the origin server
.10.25. [vhostMin]	http.res2xxCompleteCount	Integer	Total number of successful 2xx transactions from the origin server
.10.30. [vhostMin]	http.res3xxAverage	Integer	Number of 3xx responses from the origin server
.10.31. [vhostMin]	http.res3xxCompleteAverage	Integer	Number of successful 3xx transactions from the origin server
.10.32. [vhostMin]	http.res3xxTimeRes	Integer	Average elapsed time to receive a 3xx header from the origin server
.10.33. [vhostMin]	http.res3xxTimeComplete	Integer	Average completion time of 3xx transactions from the origin server
.10.34. [vhostMin]	http.res3xxCount	Integer	Total number of 3xx responses from the origin server
.10.35. [vhostMin]	http.res3xxCompleteCount	Integer	Total number of successful 3xx transactions from the origin server
.10.40. [vhostMin]	http.res4xxAverage	Integer	Number of 4xx responses from the origin server
.10.41. [vhostMin]	http.res4xxCompleteAverage	Integer	Number of successful 4xx transactions from the origin server
.10.42. [vhostMin]	http.res4xxTimeRes	Integer	Average elapsed time to receive a 4xx header from the origin server
.10.43. [vhostMin]	http.res4xxTimeComplete	Integer	Average completion time of 4xx transactions from the origin server

Table 4.2 – continued from previous page

OID	Name	Type	Description
.10.44. [vhostMin]	http.res4xxCount	Integer	Total number of 4xx responses from the origin server
.10.45. [vhostMin]	http.res4xxCompleteCount	Integer	Total number of successful 4xx transactions from the origin server
.10.50. [vhostMin]	http.res5xxAverage	Integer	Number of 5xx responses from the origin server
.10.51. [vhostMin]	http.res5xxCompleteAverage	Integer	Number of successful 5xx transactions from the origin server
.10.52. [vhostMin]	http.res5xxTimeRes	Integer	Average elapsed time to receive a 5xx header from the origin server
.10.53. [vhostMin]	http.res5xxTimeComplete	Integer	Average completion time of 5xx transactions from the origin server
.10.54. [vhostMin]	http.res5xxCount	Integer	Total number of 5xx responses from the origin server
.10.55. [vhostMin]	http.res5xxCompleteCount	Integer	Total number of successful 5xx transactions from the origin server
.10.60. [vhostMin]	http.connectTimeoutAverage	Integer	Average number of origin server connection timeouts
.10.61. [vhostMin]	http.receiveTimeoutAverage	Integer	Average number of origin server reception timeouts
.10.62. [vhostMin]	http.connectAverage	Integer	Average number of origin server connection successes
.10.63. [vhostMin]	http.dnsQueryTime	Integer	Average DNS query time when connecting to the origin server
.10.64. [vhostMin]	http.connectTime	Integer	Origin server average connection time (0.01 ms)
.10.65. [vhostMin]	http.connectTimeoutCount	Integer	Total number of origin server connection timeouts
.10.66. [vhostMin]	http.receiveTimeoutCount	Integer	Total number of origin server reception timeouts
.10.67. [vhostMin]	http.connectCount	Integer	Total number of origin server connection successes
.10.68. [vhostMin]	http.closeAverage	Integer	Average number of sockets closed by the origin server during
.10.69. [vhostMin]	http.closeCount	Integer	Total number of sockets closed by the origin server during
.11	portbypass	OID	Port bypass origin server traffic information
.11.1. [vhostMin]	portbypass.inbound	Integer	Average traffic received from the origin server via port bypass
.11.2. [vhostMin]	portbypass.outbound	Integer	Average traffic sent to the origin server via port bypass (bytes)
.11.3. [vhostMin]	portbypass.sessionAverage	Integer	Average number of origin server sessions in port bypass
.11.4. [vhostMin]	portbypass.closedAverage	Integer	Average number of connections closed by the origin server
.11.5. [vhostMin]	portbypass.connectTimeoutAverage	Integer	Average number of origin server connection timeouts during
.11.6. [vhostMin]	portbypass.closedCount	Integer	Total number of connections closed by the origin server during
.11.7. [vhostMin]	portbypass.connectTimeoutCount	Integer	Total number of origin server connection timeouts during

cache.host.traffic.client

OID = 1.3.6.1.4.1.40001.1.4.1.11.11

Provides client traffic statistics. Client traffic is divided into HTTP traffic, SSL traffic, and port bypass traffic. SNMP does not provide statistics for each directory. Even if directory statistics are configured, the data will be accumulated before being provided.

OID	Name	Type	Description
.1. [vhostMin]	inbound	Integer	Average traffic received from clients (bytes)
.2. [vhostMin]	outbound	Integer	Average traffic sent to clients (bytes)
.3. [vhostMin]	sessionAverage	Integer	Average client session count
.4. [vhostMin]	activesessionAverage	Integer	Average client transmitting session count
.10	http	OID	Client HTTP traffic information
.10.1. [vhostMin]	http.inbound	Integer	Average HTTP traffic received from clients
.10.2. [vhostMin]	http.outbound	Integer	Average HTTP traffic sent to clients (bytes)
.10.3. [vhostMin]	http.sessionAverage	Integer	Average client HTTP session count
.10.4. [vhostMin]	http.reqHeaderSize	Integer	Average HTTP Header traffic received from clients
.10.5. [vhostMin]	http.reqBodySize	Integer	Average HTTP Body traffic received from clients
.10.6. [vhostMin]	http.resHeaderSize	Integer	Average HTTP Header traffic sent to clients
.10.7. [vhostMin]	http.resBodySize	Integer	Average HTTP Body traffic sent to clients
.10.8. [vhostMin]	http.reqAverage	Integer	Average number of HTTP requests received

Table 4.3 – continued from previous page

OID	Name	Type	Description
.10.9. [vhostMin]	http.reqCount	Integer	Total number of HTTP requests received
.10.10. [vhostMin]	http.resTotalAverage	Integer	Average number of all HTTP responses sent
.10.11. [vhostMin]	http.resTotalCompleteAverage	Integer	Average number of HTTP transactions completed
.10.12. [vhostMin]	http.resTotalTimeRes	Integer	Average elapsed time of client responses
.10.13. [vhostMin]	http.resTotalTimeComplete	Integer	Average elapsed time of client HTTP transactions
.10.14. [vhostMin]	http.resTotalCount	Integer	Total number of all HTTP responses sent
.10.15. [vhostMin]	http.resTotalCompleteCount	Integer	Total number of HTTP transactions completed
.10.20. [vhostMin]	http.res2xxAverage	Integer	Average number of 2xx responses sent to clients
.10.21. [vhostMin]	http.res2xxCompleteAverage	Integer	Average number of 2xx transactions completed
.10.22. [vhostMin]	http.res2xxTimeRes	Integer	Average elapsed time of client 2xx responses
.10.23. [vhostMin]	http.res2xxTimeComplete	Integer	Average completion time of client 2xx transactions
.10.24. [vhostMin]	http.res2xxCount	Integer	Total number of 2xx responses sent to clients
.10.25. [vhostMin]	http.res2xxCompleteCount	Integer	Total number of 2xx transactions completed
.10.30. [vhostMin]	http.res3xxAverage	Integer	Average number of 3xx responses sent to clients
.10.31. [vhostMin]	http.res3xxCompleteAverage	Integer	Average number of 3xx transactions completed
.10.32. [vhostMin]	http.res3xxTimeRes	Integer	Average elapsed time of client 3xx responses
.10.33. [vhostMin]	http.res3xxTimeComplete	Integer	Average completion time of client 3xx transactions
.10.34. [vhostMin]	http.res3xxCount	Integer	Total number of 3xx responses sent to clients
.10.35. [vhostMin]	http.res3xxCompleteCount	Integer	Total number of 3xx transactions completed
.10.40. [vhostMin]	http.res4xxAverage	Integer	Average number of 4xx responses sent to clients
.10.41. [vhostMin]	http.res4xxCompleteAverage	Integer	Average number of 4xx transactions completed
.10.42. [vhostMin]	http.res4xxTimeRes	Integer	Average elapsed time of client 4xx responses
.10.43. [vhostMin]	http.res4xxTimeComplete	Integer	Average completion time of client 4xx transactions
.10.44. [vhostMin]	http.res4xxCount	Integer	Total number of 4xx responses sent to clients
.10.45. [vhostMin]	http.res4xxCompleteCount	Integer	Total number of 4xx transactions completed
.10.50. [vhostMin]	http.res5xxAverage	Integer	Average number of 5xx responses sent to clients
.10.51. [vhostMin]	http.res5xxCompleteAverage	Integer	Average number of 5xx transactions completed
.10.52. [vhostMin]	http.res5xxTimeRes	Integer	Average elapsed time of client 5xx responses
.10.53. [vhostMin]	http.res5xxTimeComplete	Integer	Average completion time of client 5xx transactions
.10.54. [vhostMin]	http.res5xxCount	Integer	Total number of 5xx responses sent to clients
.10.55. [vhostMin]	http.res5xxCompleteCount	Integer	Total number of 5xx transactions completed
.10.60. [vhostMin]	http.reqDeniedAverage	Integer	Average number of denied requests
.10.61. [vhostMin]	http.reqDeniedCount	Integer	Total number of denied requests
.11	portbypass	OID	Port bypass client traffic information
.11.1. [vhostMin]	portbypass.inbound	Integer	Average traffic received from clients via port bypass
.11.2. [vhostMin]	portbypass.outbound	Integer	Average traffic sent to clients via port bypass
.11.3. [vhostMin]	portbypass.sessionAverage	Integer	Average number of client sessions in port bypass
.11.4. [vhostMin]	portbypass.closedAverage	Integer	Average number of connections closed by port bypass
.11.5. [vhostMin]	portbypass.closedCount	Integer	Total number of connections closed by port bypass
.12	ssl	OID	SSL client traffic information
.12.2. [vhostMin]	ssl.inbound	Integer	Average traffic received from clients via SSL
.12.3. [vhostMin]	ssl.outbound	Integer	Average traffic sent to clients via SSL (by client)
.13	requestHitAverage	OID	Average number of cache HIT results
.13.1. [vhostMin]	requestHitAverage.TCP_HIT	Integer	TCP_HIT
.13.2. [vhostMin]	requestHitAverage.TCP_IMS_HIT	Integer	TCP_IMS_HIT
.13.3. [vhostMin]	requestHitAverage.TCP_REFRESH_HIT	Integer	TCP_REFRESH_HIT
.13.4. [vhostMin]	requestHitAverage.TCP_REF_FAIL_HIT	Integer	TCP_REF_FAIL_HIT
.13.5. [vhostMin]	requestHitAverage.TCP_NEGATIVE_HIT	Integer	TCP_NEGATIVE_HIT
.13.6. [vhostMin]	requestHitAverage.TCP_MISS	Integer	TCP_MISS

Table 4.3 – continued from previous page

OID	Name	Type	Description
.13.7. [vhostMin]	requestHitAverage.TCP_REFRESH_MISS	Integer	TCP_REFRESH_MISS
.13.8. [vhostMin]	requestHitAverage.TCP_CLIENT_REFRESH_MISS	Integer	TCP_CLIENT_REFRESH_MISS
.13.9. [vhostMin]	requestHitAverage.TCP_DENIED	Integer	TCP_DENIED
.13.10. [vhostMin]	requestHitAverage.TCP_ERROR	Integer	TCP_ERROR
.13.11. [vhostMin]	requestHitAverage.TCP_REDIRECT_HIT	Integer	TCP_REDIRECT_HIT
.14	requestHitCount	OID	Total number of cache HIT results
.14.1. [vhostMin]	requestHitCount.TCP_HIT	Integer	TCP_HIT
.14.2. [vhostMin]	requestHitCount.TCP_IMS_HIT	Integer	TCP_IMS_HIT
.14.3. [vhostMin]	requestHitCount.TCP_REFRESH_HIT	Integer	TCP_REFRESH_HIT
.14.4. [vhostMin]	requestHitCount.TCP_REF_FAIL_HIT	Integer	TCP_REF_FAIL_HIT
.14.5. [vhostMin]	requestHitCount.TCP_NEGATIVE_HIT	Integer	TCP_NEGATIVE_HIT
.14.6. [vhostMin]	requestHitCount.TCP_MISS	Integer	TCP_MISS
.14.7. [vhostMin]	requestHitCount.TCP_REFRESH_MISS	Integer	TCP_REFRESH_MISS
.14.8. [vhostMin]	requestHitCount.TCP_CLIENT_REFRESH_MISS	Integer	TCP_CLIENT_REFRESH_MISS
.14.9. [vhostMin]	requestHitCount.TCP_DENIED	Integer	TCP_DENIED
.14.10. [vhostMin]	requestHitCount.TCP_ERROR	Integer	TCP_ERROR
.14.11. [vhostMin]	requestHitCount.TCP_REDIRECT_HIT	Integer	TCP_REDIRECT_HIT

cache.host.traffic.filesystem

OID = 1.3.6.1.4.1.40001.1.4.1.11.20

Provides File I/O statistics of hosts.

OID	Name	Type	Description
.1. [vhostMin]	requestHitRatio	Integer	Request Hit Ratio (100%)
.2. [vhostMin]			Request Hit Ratio (10000%)
.3. [vhostMin]	byteHitRatio	Integer	Byte Hit Ratio (100%)
.4. [vhostMin]			Byte Hit Ratio (10000%)
.5. [vhostMin]	outbound	Integer	Average traffic sent to File I/O (bytes)
.6. [vhostMin]	session	Integer	Average number of threads in File I/O
.7	requestHitAverage	OID	Average number of cache HIT results
.7.1. [vhostMin]	requestHitAverage.TCP_HIT	Integer	TCP_HIT
.7.2. [vhostMin]	requestHitAverage.TCP_IMS_HIT	Integer	TCP_IMS_HIT
.7.3. [vhostMin]	requestHitAverage.TCP_REFRESH_HIT	Integer	TCP_REFRESH_HIT
.7.4. [vhostMin]	requestHitAverage.TCP_REF_FAIL_HIT	Integer	TCP_REF_FAIL_HIT
.7.5. [vhostMin]	requestHitAverage.TCP_NEGATIVE_HIT	Integer	TCP_NEGATIVE_HIT
.7.6. [vhostMin]	requestHitAverage.TCP_MISS	Integer	TCP_MISS
.7.7. [vhostMin]	requestHitAverage.TCP_REFRESH_MISS	Integer	TCP_REFRESH_MISS
.7.8. [vhostMin]	requestHitAverage.TCP_CLIENT_REFRESH_MISS	Integer	TCP_CLIENT_REFRESH_MISS
.7.9. [vhostMin]	requestHitAverage.TCP_DENIED	Integer	TCP_DENIED
.7.10. [vhostMin]	requestHitAverage.TCP_ERROR	Integer	TCP_ERROR
.7.11. [vhostMin]	requestHitAverage.TCP_REDIRECT_HIT	Integer	TCP_REDIRECT_HIT
.8	requestHitCount	OID	Total number of cache HIT results
.8.1. [vhostMin]	requestHitCount.TCP_HIT	Integer	TCP_HIT
.8.2. [vhostMin]	requestHitCount.TCP_IMS_HIT	Integer	TCP_IMS_HIT
.8.3. [vhostMin]	requestHitCount.TCP_REFRESH_HIT	Integer	TCP_REFRESH_HIT
.8.4. [vhostMin]	requestHitCount.TCP_REF_FAIL_HIT	Integer	TCP_REF_FAIL_HIT
.8.5. [vhostMin]	requestHitCount.TCP_NEGATIVE_HIT	Integer	TCP_NEGATIVE_HIT

Table 4.4 – continued from previous page

OID	Name	Type	Description
.8.6. [vhostMin]	requestHitCount.TCP_MISS	Integer	TCP_MISS
.8.7. [vhostMin]	requestHitCount.TCP_REFRESH_MISS	Integer	TCP_REFRESH_MISS
.8.8. [vhostMin]	requestHitCount.TCP_CLIENT_REFRESH_MISS	Integer	TCP_CLIENT_REFRESH_MISS
.8.9. [vhostMin]	requestHitCount.TCP_DENIED	Integer	TCP_DENIED
.8.10. [vhostMin]	requestHitCount.TCP_ERROR	Integer	TCP_ERROR
.8.11. [vhostMin]	requestHitCount.TCP_REDIRECT_HIT	Integer	TCP_REDIRECT_HIT
.10. [vhostMin]	getattr.filecount	Integer	(getattr function call) Number of FILE resp
.11. [vhostMin]	getattr.dircount	Integer	(getattr function call) Number of DIR resp
.12. [vhostMin]	getattr.failcount	Integer	(getattr function call) Number of failure res
.13. [vhostMin]	getattr.timeres	Integer	(getattr function call) Response time (0.01
.14. [vhostMin]	open.count	Integer	Number of open function calls
.15. [vhostMin]	open.timeres	Integer	Response time of the open function (0.01 m
.16. [vhostMin]	read.count	Integer	Number of read function calls
.17. [vhostMin]	read.timeres	Integer	Response time of the read function (0.01 m
.18. [vhostMin]	read.bufferize	Integer	Size of the buffer requested by the read fun
.19. [vhostMin]	read.bufferfilled	Integer	Size of filled space in the buffer requested

cache.host.traffic.dims

OID = 1.3.6.1.4.1.40001.1.4.1.11.21

Provides DIMS conversion statistics of hosts.

OID	Name	Type	Description
.1. [vhostMin]	requests	Integer	Number of DIMS conversion requests
.2. [vhostMin]	converted	Integer	Number of conversion successes
.3. [vhostMin]	failed	Integer	Number of conversion failures
.4. [vhostMin]	avgsrcsize	Integer	Average size of origin images (bytes)
.5. [vhostMin]	avgdestsize	Integer	Average size of converted images (bytes)
.6. [vhostMin]	avgtime	Integer	Conversion time (ms)

cache.host.traffic.compression

OID = 1.3.6.1.4.1.40001.1.4.1.11.22

Provides compression statistics of hosts.

OID	Name	Type	Description
.1. [vhostMin]	requests	Integer	Number of compression requests
.2. [vhostMin]	converted	Integer	Number of compression successes
.3. [vhostMin]	failed	Integer	Number of compression failures
.4. [vhostMin]	avgsrcsize	Integer	Average size of origin files (bytes)
.5. [vhostMin]	avgdestsize	Integer	Average size of compressed files (bytes)
.6. [vhostMin]	avgtime	Integer	Compression time (ms)

4.2.9 cache.vhost

OID = 1.3.6.1.4.1.40001.1.4.3.1

Provides virtual host information. [vhostIndex] starts at 1 and ranges up to the number of virtual hosts.

OID	Name	Type	Description
.2. [vhostIndex]	name	String	Virtual host name
.3. [vhostIndex]	status	String	“Healthy” or “Inactive” or “Emergency”
.4. [vhostIndex]	uptime	Integer	Virtual host runtime (seconds)
.10	contents	OID	Content information (expansion)
.11	traffic	OID	Statistics (expansion)

cache.vhost.contents

OID = 1.3.6.1.4.1.40001.1.4.3.1.10

Provides statistics for content in the service for a virtual host.

OID	Name	Type	Description
.1. [vhostIndex]	memory	Integer	Memory caching size (KB)
.2. [vhostIndex]	filesTotalCount	Integer	The number of files in service
.3. [vhostIndex]	filesTotalSize	Integer	Total size of files in service (MB)
.10. [vhostIndex]	filesCountU1KB	Integer	The number of files smaller than 1KB
.11. [vhostIndex]	filesCountU2KB	Integer	The number of files smaller than 2KB
.12. [vhostIndex]	filesCountU4KB	Integer	The number of files smaller than 4KB
.13. [vhostIndex]	filesCountU8KB	Integer	The number of files smaller than 8KB
.14. [vhostIndex]	filesCountU16KB	Integer	The number of files smaller than 16KB
.15. [vhostIndex]	filesCountU32KB	Integer	The number of files smaller than 32KB
.16. [vhostIndex]	filesCountU64KB	Integer	The number of files smaller than 64KB
.17. [vhostIndex]	filesCountU128KB	Integer	The number of files smaller than 128KB
.18. [vhostIndex]	filesCountU256KB	Integer	The number of files smaller than 256KB
.19. [vhostIndex]	filesCountU512KB	Integer	The number of files smaller than 512KB
.20. [vhostIndex]	filesCountU1MB	Integer	The number of files smaller than 1MB
.21. [vhostIndex]	filesCountU2MB	Integer	The number of files smaller than 2MB
.22. [vhostIndex]	filesCountU4MB	Integer	The number of files smaller than 4MB
.23. [vhostIndex]	filesCountU8MB	Integer	The number of files smaller than 8MB
.24. [vhostIndex]	filesCountU16MB	Integer	The number of files smaller than 16MB
.25. [vhostIndex]	filesCountU32MB	Integer	The number of files smaller than 32MB
.26. [vhostIndex]	filesCountU64MB	Integer	The number of files smaller than 64MB
.27. [vhostIndex]	filesCountU128MB	Integer	The number of files smaller than 128MB
.28. [vhostIndex]	filesCountU256MB	Integer	The number of files smaller than 256MB
.29. [vhostIndex]	filesCountU512MB	Integer	The number of files smaller than 512MB
.30. [vhostIndex]	filesCountU1GB	Integer	The number of files smaller than 1GB
.31. [vhostIndex]	filesCountU2GB	Integer	The number of files smaller than 2GB
.32. [vhostIndex]	filesCountU4GB	Integer	The number of files smaller than 4GB
.33. [vhostIndex]	filesCountU8GB	Integer	The number of files smaller than 8GB
.34. [vhostIndex]	filesCountU16GB	Integer	The number of files smaller than 16GB
.35. [vhostIndex]	filesCountO16GB	Integer	The number of files larger than 16GB

cache.vhost.traffic

OID = 1.3.6.1.4.1.40001.1.4.3.1.11

Provides cache service and traffic statistics for a virtual host. Traffic statistics are provided as an average of up to 60 minutes. If the time value is omitted or set to 0, statistics will be provided in real time.

OID	Name	Type	Description
.1. [vhostMin] . [vhostIndex]	requestHitRatio	Integer	Request Hit Ratio (100%)
.2. [vhostMin] . [vhostIndex]			Request Hit Ratio (10000%)
.3. [vhostMin] . [vhostIndex]	bytesHitRatio	Integer	Bytes Hit Ratio (100%)
.4. [vhostMin] . [vhostIndex]			Bytes Hit Ratio (10000%)
.10	origin	OID	Origin traffic information (expansion)
.11	client	OID	Client traffic information (expansion)

cache.vhost.traffic.origin

OID = 1.3.6.1.4.1.40001.1.4.3.1.11.10

Provides origin server traffic statistics. Origin server traffic is divided into HTTP traffic and port bypass traffic.

OID	Name	Type	Description
.1. [vhostMin] . [vhostIndex]	inbound	Integer	Average traffic received from the origin server
.2. [vhostMin] . [vhostIndex]	outbound	Integer	Average traffic sent to the origin server
.3. [vhostMin] . [vhostIndex]	sessionAverage	Integer	Average origin server session count
.4. [vhostMin] . [vhostIndex]	activesessionAverage	Integer	Average origin server transmitting sessions
.10	http	OID	Origin server HTTP traffic information (expansion)
.10.1. [vhostMin] . [vhostIndex]	http.inbound	Integer	Average HTTP traffic received from the origin server
.10.2. [vhostMin] . [vhostIndex]	http.outbound	Integer	Average HTTP traffic sent to the origin server
.10.3. [vhostMin] . [vhostIndex]	http.sessionAverage	Integer	Average origin server HTTP session count
.10.4. [vhostMin] . [vhostIndex]	http.reqHeaderSize	Integer	Average HTTP Header traffic sent to the origin server
.10.5. [vhostMin] . [vhostIndex]	http.reqBodySize	Integer	Average HTTP Body traffic sent to the origin server
.10.6. [vhostMin] . [vhostIndex]	http.resHeaderSize	Integer	Average HTTP Header traffic received from the origin server
.10.7. [vhostMin] . [vhostIndex]	http.resBodySize	Integer	Average HTTP Body traffic received from the origin server
.10.8. [vhostMin] . [vhostIndex]	http.reqAverage	Integer	Average number of HTTP requests sent to the origin server
.10.9. [vhostMin] . [vhostIndex]	http.reqCount	Integer	Total number of HTTP requests sent to the origin server
.10.10. [vhostMin] . [vhostIndex]	http.resTotalAverage	Integer	Average number of all HTTP responses received from the origin server
.10.11. [vhostMin] . [vhostIndex]	http.resTotalCompleteAverage	Integer	Average number of successful HTTP transactions received from the origin server
.10.12. [vhostMin] . [vhostIndex]	http.resTotalTimeRes	Integer	Average elapsed time to receive a response from the origin server
.10.13. [vhostMin] . [vhostIndex]	http.resTotalTimeComplete	Integer	Average completion time of HTTP transactions received from the origin server
.10.14. [vhostMin] . [vhostIndex]	http.resTotalCount	Integer	Total number of all HTTP responses received from the origin server
.10.15. [vhostMin] . [vhostIndex]	http.resTotalCompleteCount	Integer	Total number of successful HTTP transactions received from the origin server
.10.20. [vhostMin] . [vhostIndex]	http.res2xxAverage	Integer	Number of 2xx responses from the origin server
.10.21. [vhostMin] . [vhostIndex]	http.res2xxCompleteAverage	Integer	Number of successful 2xx transactions received from the origin server
.10.22. [vhostMin] . [vhostIndex]	http.res2xxTimeRes	Integer	Average elapsed time to receive a 2xx response from the origin server
.10.23. [vhostMin] . [vhostIndex]	http.res2xxTimeComplete	Integer	Average completion time of 2xx transactions received from the origin server
.10.24. [vhostMin] . [vhostIndex]	http.res2xxCount	Integer	Total number of 2xx responses from the origin server
.10.25. [vhostMin] . [vhostIndex]	http.res2xxCompleteCount	Integer	Total number of successful 2xx transactions received from the origin server
.10.30. [vhostMin] . [vhostIndex]	http.res3xxAverage	Integer	Number of 3xx responses from the origin server
.10.31. [vhostMin] . [vhostIndex]	http.res3xxCompleteAverage	Integer	Number of successful 3xx transactions received from the origin server
.10.32. [vhostMin] . [vhostIndex]	http.res3xxTimeRes	Integer	Average elapsed time to receive a 3xx response from the origin server
.10.33. [vhostMin] . [vhostIndex]	http.res3xxTimeComplete	Integer	Average completion time of 3xx transactions received from the origin server
.10.34. [vhostMin] . [vhostIndex]	http.res3xxCount	Integer	Total number of 3xx responses from the origin server
.10.35. [vhostMin] . [vhostIndex]	http.res3xxCompleteCount	Integer	Total number of successful 3xx transactions received from the origin server
.10.40. [vhostMin] . [vhostIndex]	http.res4xxAverage	Integer	Number of 4xx responses from the origin server
.10.41. [vhostMin] . [vhostIndex]	http.res4xxCompleteAverage	Integer	Number of successful 4xx transactions received from the origin server
.10.42. [vhostMin] . [vhostIndex]	http.res4xxTimeRes	Integer	Average elapsed time to receive a 4xx response from the origin server
.10.43. [vhostMin] . [vhostIndex]	http.res4xxTimeComplete	Integer	Average completion time of 4xx transactions received from the origin server

Table 4.5 – continued from previous page

OID	Name	Type	Description
.10.44. [vhostMin] . [vhostIndex]	http.res4xxCount	Integer	Total number of 4xx responses from the origin server
.10.45. [vhostMin] . [vhostIndex]	http.res4xxCompleteCount	Integer	Total number of successful 4xx transactions
.10.50. [vhostMin] . [vhostIndex]	http.res5xxAverage	Integer	Number of 5xx responses from the origin server
.10.51. [vhostMin] . [vhostIndex]	http.res5xxCompleteAverage	Integer	Number of successful 5xx transactions
.10.52. [vhostMin] . [vhostIndex]	http.res5xxTimeRes	Integer	Average elapsed time to receive a 5xx response
.10.53. [vhostMin] . [vhostIndex]	http.res5xxTimeComplete	Integer	Average completion time of 5xx transactions
.10.54. [vhostMin] . [vhostIndex]	http.res5xxCount	Integer	Total number of 5xx responses from the origin server
.10.55. [vhostMin] . [vhostIndex]	http.res5xxCompleteCount	Integer	Total number of successful 5xx transactions
.10.60. [vhostMin] . [vhostIndex]	http.connectTimeoutAverage	Integer	Average number of origin server connection attempts
.10.61. [vhostMin] . [vhostIndex]	http.receiveTimeoutAverage	Integer	Average number of origin server reception attempts
.10.62. [vhostMin] . [vhostIndex]	http.connectAverage	Integer	Average number of origin server connection attempts
.10.63. [vhostMin] . [vhostIndex]	http.dnsQueryTime	Integer	Average DNS query time when connecting to the origin server
.10.64. [vhostMin] . [vhostIndex]	http.connectTime	Integer	Origin server average connection time
.10.65. [vhostMin] . [vhostIndex]	http.connectTimeoutCount	Integer	Total number of origin server connection attempts
.10.66. [vhostMin] . [vhostIndex]	http.receiveTimeoutCount	Integer	Total number of origin server reception attempts
.10.67. [vhostMin] . [vhostIndex]	http.connectCount	Integer	Total number of origin server connection attempts
.10.68. [vhostMin] . [vhostIndex]	http.closeAverage	Integer	Average number of sockets closed by the origin server
.10.69. [vhostMin] . [vhostIndex]	http.closeCount	Integer	Total number of sockets closed by the origin server
.11	portbypass	OID	Port bypass origin server traffic information
.11.1. [vhostMin] . [vhostIndex]	portbypass.inbound	Integer	Average traffic received from the origin server
.11.2. [vhostMin] . [vhostIndex]	portbypass.outbound	Integer	Average traffic sent to the origin server
.11.3. [vhostMin] . [vhostIndex]	portbypass.sessionAverage	Integer	Average number of origin server sessions
.11.4. [vhostMin] . [vhostIndex]	portbypass.closedAverage	Integer	Average number of connections closed by the origin server
.11.5. [vhostMin] . [vhostIndex]	portbypass.connectTimeoutAverage	Integer	Average number of origin server connection attempts
.11.6. [vhostMin] . [vhostIndex]	portbypass.closedCount	Integer	Total number of connections closed by the origin server
.11.7. [vhostMin] . [vhostIndex]	portbypass.connectTimeoutCount	Integer	Total number of origin server connection attempts

cache.vhost.traffic.client

OID = 1.3.6.1.4.1.40001.1.4.3.1.11.11

Provides client traffic statistics. Client traffic is divided into HTTP traffic, SSL traffic, and port bypass traffic. SNMP does not provide statistics for each directory. Even if directory statistics are configured, the data will be accumulated before being provided.

OID	Name	Type	Description
.1. [vhostMin] . [vhostIndex]	inbound	Integer	Average traffic received from the client
.2. [vhostMin] . [vhostIndex]	outbound	Integer	Average traffic sent to the client
.3. [vhostMin] . [vhostIndex]	sessionAverage	Integer	Average client session duration
.4. [vhostMin] . [vhostIndex]	activesessionAverage	Integer	Average client transmission rate
.10	http	OID	Client HTTP traffic information
.10.1. [vhostMin] . [vhostIndex]	http.inbound	Integer	Average HTTP traffic received from the client
.10.2. [vhostMin] . [vhostIndex]	http.outbound	Integer	Average HTTP traffic sent to the client
.10.3. [vhostMin] . [vhostIndex]	http.sessionAverage	Integer	Average client HTTP session duration
.10.4. [vhostMin] . [vhostIndex]	http.reqHeaderSize	Integer	Average HTTP Header size
.10.5. [vhostMin] . [vhostIndex]	http.reqBodySize	Integer	Average HTTP Body size
.10.6. [vhostMin] . [vhostIndex]	http.resHeaderSize	Integer	Average HTTP Header size
.10.7. [vhostMin] . [vhostIndex]	http.resBodySize	Integer	Average HTTP Body size
.10.8. [vhostMin] . [vhostIndex]	http.reqAverage	Integer	Average number of HTTP requests

Table 4.6 – continued from previous page

OID	Name	Type	Description
.10.9. [vhostMin] . [vhostIndex]	http.reqCount	Integer	Total number of HTTP requests
.10.10. [vhostMin] . [vhostIndex]	http.resTotalAverage	Integer	Average number of all HTTP responses
.10.11. [vhostMin] . [vhostIndex]	http.resTotalCompleteAverage	Integer	Average number of HTTP responses completed
.10.12. [vhostMin] . [vhostIndex]	http.resTotalTimeRes	Integer	Average elapsed time for all HTTP responses
.10.13. [vhostMin] . [vhostIndex]	http.resTotalTimeComplete	Integer	Average elapsed time for all HTTP responses completed
.10.14. [vhostMin] . [vhostIndex]	http.resTotalCount	Integer	Total number of all HTTP responses
.10.15. [vhostMin] . [vhostIndex]	http.resTotalCompleteCount	Integer	Total number of HTTP responses completed
.10.20. [vhostMin] . [vhostIndex]	http.res2xxAverage	Integer	Average number of 2xx responses
.10.21. [vhostMin] . [vhostIndex]	http.res2xxCompleteAverage	Integer	Average number of 2xx responses completed
.10.22. [vhostMin] . [vhostIndex]	http.res2xxTimeRes	Integer	Average elapsed time for 2xx responses
.10.23. [vhostMin] . [vhostIndex]	http.res2xxTimeComplete	Integer	Average completion time for 2xx responses
.10.24. [vhostMin] . [vhostIndex]	http.res2xxCount	Integer	Total number of 2xx responses
.10.25. [vhostMin] . [vhostIndex]	http.res2xxCompleteCount	Integer	Total number of 2xx responses completed
.10.30. [vhostMin] . [vhostIndex]	http.res3xxAverage	Integer	Average number of 3xx responses
.10.31. [vhostMin] . [vhostIndex]	http.res3xxCompleteAverage	Integer	Average number of 3xx responses completed
.10.32. [vhostMin] . [vhostIndex]	http.res3xxTimeRes	Integer	Average elapsed time for 3xx responses
.10.33. [vhostMin] . [vhostIndex]	http.res3xxTimeComplete	Integer	Average completion time for 3xx responses
.10.34. [vhostMin] . [vhostIndex]	http.res3xxCount	Integer	Total number of 3xx responses
.10.35. [vhostMin] . [vhostIndex]	http.res3xxCompleteCount	Integer	Total number of 3xx responses completed
.10.40. [vhostMin] . [vhostIndex]	http.res4xxAverage	Integer	Average number of 4xx responses
.10.41. [vhostMin] . [vhostIndex]	http.res4xxCompleteAverage	Integer	Average number of 4xx responses completed
.10.42. [vhostMin] . [vhostIndex]	http.res4xxTimeRes	Integer	Average elapsed time for 4xx responses
.10.43. [vhostMin] . [vhostIndex]	http.res4xxTimeComplete	Integer	Average completion time for 4xx responses
.10.44. [vhostMin] . [vhostIndex]	http.res4xxCount	Integer	Total number of 4xx responses
.10.45. [vhostMin] . [vhostIndex]	http.res4xxCompleteCount	Integer	Total number of 4xx responses completed
.10.50. [vhostMin] . [vhostIndex]	http.res5xxAverage	Integer	Average number of 5xx responses
.10.51. [vhostMin] . [vhostIndex]	http.res5xxCompleteAverage	Integer	Average number of 5xx responses completed
.10.52. [vhostMin] . [vhostIndex]	http.res5xxTimeRes	Integer	Average elapsed time for 5xx responses
.10.53. [vhostMin] . [vhostIndex]	http.res5xxTimeComplete	Integer	Average completion time for 5xx responses
.10.54. [vhostMin] . [vhostIndex]	http.res5xxCount	Integer	Total number of 5xx responses
.10.55. [vhostMin] . [vhostIndex]	http.res5xxCompleteCount	Integer	Total number of 5xx responses completed
.10.60. [vhostMin] . [vhostIndex]	http.reqDeniedAverage	Integer	Average number of denied requests
.10.61. [vhostMin] . [vhostIndex]	http.reqDeniedCount	Integer	Total number of denied requests
.11	portbypass	OID	Port bypass client traffic info
.11.1. [vhostMin] . [vhostIndex]	portbypass.inbound	Integer	Average traffic received
.11.2. [vhostMin] . [vhostIndex]	portbypass.outbound	Integer	Average traffic sent to
.11.3. [vhostMin] . [vhostIndex]	portbypass.sessionAverage	Integer	Average number of client sessions
.11.4. [vhostMin] . [vhostIndex]	portbypass.closedAverage	Integer	Average number of closed sessions
.11.5. [vhostMin] . [vhostIndex]	portbypass.closedCount	Integer	Total number of closed sessions
.12	ssl	OID	SSL client traffic info
.12.2. [vhostMin] . [vhostIndex]	ssl.inbound	Integer	Average traffic received
.12.3. [vhostMin] . [vhostIndex]	ssl.outbound	Integer	Average traffic sent to
.13	requestHitAverage	OID	Average number of cache hits
.13.1. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_HIT	Integer	TCP_HIT
.13.2. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_IMS_HIT	Integer	TCP_IMS_HIT
.13.3. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_REFRESH_HIT	Integer	TCP_REFRESH_HIT
.13.4. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_REF_FAIL_HIT	Integer	TCP_REF_FAIL_HIT
.13.5. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_NEGATIVE_HIT	Integer	TCP_NEGATIVE_HIT
.13.6. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_MISS	Integer	TCP_MISS

Table 4.6 – continued from previous page

OID	Name	Type	Description
.13.7. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_REFRESH_MISS	Integer	TCP_REFRESH_MISS
.13.8. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_CLIENT_REFRESH_MISS	Integer	TCP_CLIENT_REFRESH_MISS
.13.9. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_DENIED	Integer	TCP_DENIED
.13.10. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_ERROR	Integer	TCP_ERROR
.13.11. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_REDIRECT_HIT	Integer	TCP_REDIRECT_HIT
.14	requestHitCount	OID	Total number of cache hits
.14.1. [vhostMin] . [vhostIndex]	requestHitCount.TCP_HIT	Integer	TCP_HIT
.14.2. [vhostMin] . [vhostIndex]	requestHitCount.TCP_IMS_HIT	Integer	TCP_IMS_HIT
.14.3. [vhostMin] . [vhostIndex]	requestHitCount.TCP_REFRESH_HIT	Integer	TCP_REFRESH_HIT
.14.4. [vhostMin] . [vhostIndex]	requestHitCount.TCP_REF_FAIL_HIT	Integer	TCP_REF_FAIL_HIT
.14.5. [vhostMin] . [vhostIndex]	requestHitCount.TCP_NEGATIVE_HIT	Integer	TCP_NEGATIVE_HIT
.14.6. [vhostMin] . [vhostIndex]	requestHitCount.TCP_MISS	Integer	TCP_MISS
.14.7. [vhostMin] . [vhostIndex]	requestHitCount.TCP_REFRESH_MISS	Integer	TCP_REFRESH_MISS
.14.8. [vhostMin] . [vhostIndex]	requestHitCount.TCP_CLIENT_REFRESH_MISS	Integer	TCP_CLIENT_REFRESH_MISS
.14.9. [vhostMin] . [vhostIndex]	requestHitCount.TCP_DENIED	Integer	TCP_DENIED
.14.10. [vhostMin] . [vhostIndex]	requestHitCount.TCP_ERROR	Integer	TCP_ERROR
.14.11. [vhostMin] . [vhostIndex]	requestHitCount.TCP_REDIRECT_HIT	Integer	TCP_REDIRECT_HIT

cache.vhost.traffic.filesystem

OID = 1.3.6.1.4.1.40001.1.4.3.1.11.20

Provides File I/O statistics of a virtual host.

OID	Name	Type	Description
.1. [vhostMin] . [vhostIndex]	requestHitRatio	Integer	Request Hit Ratio (100%)
.2. [vhostMin] . [vhostIndex]			Request Hit Ratio (100%)
.3. [vhostMin] . [vhostIndex]	byteHitRatio	Integer	Byte Hit Ratio (100%)
.4. [vhostMin] . [vhostIndex]			Byte Hit Ratio (10000%)
.5. [vhostMin] . [vhostIndex]	outbound	Integer	Average traffic sent to vhost
.6. [vhostMin] . [vhostIndex]	session	Integer	Average number of threads
.7	requestHitAverage	OID	Average number of cache hits
.7.1. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_HIT	Integer	TCP_HIT
.7.2. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_IMS_HIT	Integer	TCP_IMS_HIT
.7.3. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_REFRESH_HIT	Integer	TCP_REFRESH_HIT
.7.4. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_REF_FAIL_HIT	Integer	TCP_REF_FAIL_HIT
.7.5. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_NEGATIVE_HIT	Integer	TCP_NEGATIVE_HIT
.7.6. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_MISS	Integer	TCP_MISS
.7.7. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_REFRESH_MISS	Integer	TCP_REFRESH_MISS
.7.8. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_CLIENT_REFRESH_MISS	Integer	TCP_CLIENT_REFRESH_MISS
.7.9. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_DENIED	Integer	TCP_DENIED
.7.10. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_ERROR	Integer	TCP_ERROR
.7.11. [vhostMin] . [vhostIndex]	requestHitAverage.TCP_REDIRECT_HIT	Integer	TCP_REDIRECT_HIT
.8	requestHitCount	OID	Total number of cache hits
.8.1. [vhostMin] . [vhostIndex]	requestHitCount.TCP_HIT	Integer	TCP_HIT
.8.2. [vhostMin] . [vhostIndex]	requestHitCount.TCP_IMS_HIT	Integer	TCP_IMS_HIT
.8.3. [vhostMin] . [vhostIndex]	requestHitCount.TCP_REFRESH_HIT	Integer	TCP_REFRESH_HIT
.8.4. [vhostMin] . [vhostIndex]	requestHitCount.TCP_REF_FAIL_HIT	Integer	TCP_REF_FAIL_HIT
.8.5. [vhostMin] . [vhostIndex]	requestHitCount.TCP_NEGATIVE_HIT	Integer	TCP_NEGATIVE_HIT

Table 4.7 – continued from previous page

OID	Name	Type	Description
.8.6. [vhostMin] . [vhostIndex]	requestHitCount.TCP_MISS	Integer	TCP_MISS
.8.7. [vhostMin] . [vhostIndex]	requestHitCount.TCP_REFRESH_MISS	Integer	TCP_REFRESH_MISS
.8.8. [vhostMin] . [vhostIndex]	requestHitCount.TCP_CLIENT_REFRESH_MISS	Integer	TCP_CLIENT_REFRESH_MISS
.8.9. [vhostMin] . [vhostIndex]	requestHitCount.TCP_DENIED	Integer	TCP_DENIED
.8.10. [vhostMin] . [vhostIndex]	requestHitCount.TCP_ERROR	Integer	TCP_ERROR
.8.11. [vhostMin] . [vhostIndex]	requestHitCount.TCP_REDIRECT_HIT	Integer	TCP_REDIRECT_HIT
.10. [vhostMin] . [vhostIndex]	getattr.filecount	Integer	(getattr function call) N
.11. [vhostMin] . [vhostIndex]	getattr.dircount	Integer	(getattr function call) N
.12. [vhostMin] . [vhostIndex]	getattr.failcount	Integer	(getattr function call) N
.13. [vhostMin] . [vhostIndex]	getattr.timeres	Integer	(getattr function call) R
.14. [vhostMin] . [vhostIndex]	open.count	Integer	Number of open functi
.15. [vhostMin] . [vhostIndex]	open.timeres	Integer	Response time of the o
.16. [vhostMin] . [vhostIndex]	read.count	Integer	Number of read functio
.17. [vhostMin] . [vhostIndex]	read.timeres	Integer	Response time of the r
.18. [vhostMin] . [vhostIndex]	read.bufferize	Integer	Size of the buffer requ
.19. [vhostMin] . [vhostIndex]	read.bufferfilled	Integer	Size of filled space in t

cache.vhost.traffic.dims

OID = 1.3.6.1.4.1.40001.1.4.3.1.11.21

Provides DIMS conversion statistics of a virtual host.

OID	Name	Type	Description
.1. [vhostMin] . [vhostIndex]	requests	Integer	Number of DIMS conversion requests
.2. [vhostMin] . [vhostIndex]	converted	Integer	Number of conversion successes
.3. [vhostMin] . [vhostIndex]	failed	Integer	Number of conversion failures
.4. [vhostMin] . [vhostIndex]	avgsrcsize	Integer	Average size of origin images (bytes)
.5. [vhostMin] . [vhostIndex]	avgdestsize	Integer	Average size of converted images (bytes)
.6. [vhostMin] . [vhostIndex]	avgtime	Integer	Conversion time (ms)

cache.vhost.traffic.compression

OID = 1.3.6.1.4.1.40001.1.4.3.1.11.22

Provides compression statistics of a virtual host.

OID	Name	Type	Description
.1. [vhostMin] . [vhostIndex]	requests	Integer	Number of compression requests
.2. [vhostMin] . [vhostIndex]	converted	Integer	Number of compression successes
.3. [vhostMin] . [vhostIndex]	failed	Integer	Number of compression failures
.4. [vhostMin] . [vhostIndex]	avgsrcsize	Integer	Average size of origin files (bytes)
.5. [vhostMin] . [vhostIndex]	avgdestsize	Integer	Average size of compressed files (bytes)
.6. [vhostMin] . [vhostIndex]	avgtime	Integer	Compression time (ms)

4.2.10 cache.view

OID = 1.3.6.1.4.1.40001.1.4.11.1

Provides information identical to the virtual host statistics. [viewIndex] starts at 1 and ranges up to the number of Views.

- 1.3.6.1.4.1.40001.1.4.3 - Virtual host statistics
- 1.3.6.1.4.1.40001.1.4.11 - View statistics

4.3 Chapter 12. Log

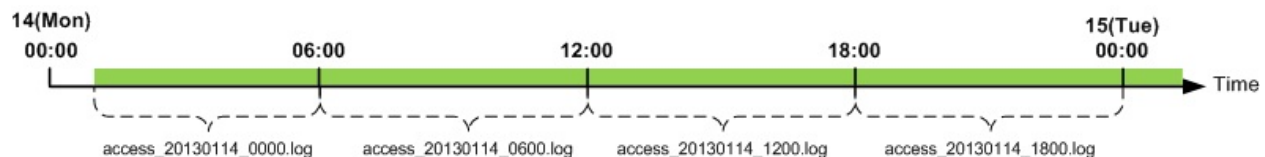
This chapter will explain the logs. A service both starts and ends with the logs. The logs can act as valuable assets to keep, laws to abide by, and arbitrators of system failure.

Logs are divided into global logs and virtual host logs. All logs can be configured to be on or off and share identical properties.

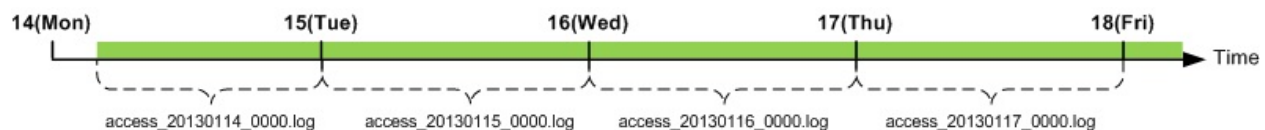
```
<XXX Type="time" Unit="1440" Retention="10" Compression="OFF">ON</XXX>
```

- **Type** (default: `time`), **Unit** (default: `1440 min`) Sets the rolling conditions for logs
 - `time` Rolls the log file for every configured unit of time (unit: min).
 - `size` Rolls the log file for every configured unit of size (unit: MB).
 - `both` Using a comma, time and size can be configured at the same time. For example, a configuration of `Unit="1440, 100"` rolls the log file every 24 hours (1440 minutes) or every 100 MB.
- **Retention** (default: `10 files`) Keeps up to the set number of log files.
- **Compression** (default: `OFF`) Compresses the log when rolling. For example, when the file `access_20140715_0000.log` is rolling, it is compressed and saved as `access_20140715_0000.log.gz`.

If `Type` is set to “time” and `Unit` set to 10, the log will be rolled on every multiple of 10 minutes. That is, even if the service started at 2:18, the log will be rolled at 2:20, 2:30, 2:40, and so on. Likewise, if you want to roll the log every day at midnight, you can set `Unit` to 1440 (60 min * 24 hours). If `Type` is set to “time”, the logs will be rolled at least once a day, and `Unit` cannot be set above 1440.



If the log is rolled with a maximum value of 24 hours (`Unit="1440"`), the log will be recorded as seen below.



4.3.1 Install Log

All details during installation/update will be recorded in the file `install.log`. No extra configuration is needed for this log.

```

#DownloadURL: http://foobar.com/ston/ston.2.0.0.rhel.2.6.32.x64.tar.gz
#DownloadTime: 13 sec
#Target: STON 2.0.0
#Date: 2014.03.03 16:48:35
Prepare for STON 2.0.0 install process
    Stopping STON...
    STON stopped
[Copying files]
    './fuse.conf' -> '/etc/fuse.conf'
    './libfuse.so.2' -> '/usr/local/ston/libfuse.so.2'
    './libtbbmalloc_proxy.so' -> '/usr/local/ston/libtbbmalloc_proxy.so'
    './start-stop-daemon' -> '/usr/sbin/start-stop-daemon'
    './libtbbmalloc_proxy.so.2' -> '/usr/local/ston/libtbbmalloc_proxy.so.2'
    './libtbbmalloc.so' -> '/usr/local/ston/libtbbmalloc.so'
    './libtbbmalloc.so.2' -> '/usr/local/ston/libtbbmalloc.so.2'
    './libtbb.so' -> '/usr/local/ston/libtbb.so'
    './libtbb.so.2' -> '/usr/local/ston/libtbb.so.2'
    './stond' -> '/usr/local/ston/stond'
    './stonx' -> '/usr/local/ston/stonx'
    './stonr' -> '/usr/local/ston/stonr'
    './stonu' -> '/usr/local/ston/stonu'
    './stonapi' -> '/usr/local/ston/stonapi'
    './server.xml.default' -> '/usr/local/ston/server.xml.default'
    './vhosts.xml.default' -> '/usr/local/ston/vhosts.xml.default'
    './ston_format.sh' -> '/usr/local/ston/ston_format.sh'
    './ston_diskinfo.sh' -> '/usr/local/ston/ston_diskinfo.sh'
    './wm.sh' -> '/usr/local/ston/wm.sh'
[Exporting config files]
    #Export so directory
    /usr/local/ston/ to ld.so.conf
    #Export sysctl to /etc/sysctl.conf
    vm.swappiness=0
    vm.min_free_kbytes=524288
    #Export sudoers for WM
    Defaults !requiretty
    winesoft ALL=NOPASSWD: /etc/init.d/ston stop, /etc/init.d/ston start, /bin/ps -ef
[Configuring STON daemon script]
    STON daemon activate in run-level 2345.
[Installing sub-packages]
    curl installed.
    libjpeg installed.
    libgomp installed.
    rrdtool installed.
[Installing WM]
    Stopping WM...
    WM stopped
    './wm.server_default.xml' -> '/usr/local/ston/wm/tmp/conf/server_default.xml'
    './wm.vhost_default.xml' -> '/usr/local/ston/wm/tmp/conf/vhost_default.xml'
    WM configuration found. Current WM port : 8500
    PHP module for Legacy(CentOS 5.5) installed
    './libphp5.so.5.5' -> '/usr/local/ston/wm/modules/libphp5.so'
    WM installation almost complete. Changing WM privileges.
Installation successfully complete

```

4.3.2 Info Log

The Info log can be configured in global settings (server.xml).

```
# server.xml - <Server><Cache>

<InfoLog Type="size" Unit="1" Retention="5">ON</InfoLog>
```

- <InfoLog> (default: ON, Type: size, Unit: 1) Records operation and configuration changes in STON.

4.3.3 Deny Log

The Deny log can be configured in global settings (server.xml).

```
# server.xml - <Server><Cache>

<DenyLog Type="size" Unit="1" Retention="5">ON</DenyLog>
```

- <DenyLog> (default: ON, Type: size, Unit: 1)

Records IP addresses denied by *Server Access Control*.

```
#Fields: date time c-ip deny
2012.11.15 07:06:10 1.1.1.1 AP
2012.11.15 07:06:26 2.2.2.2 GIN
2012.11.15 07:06:30 3.3.3.3 3.3.3.1-255
```

Fields are separated by spaces, and each field refers to the following:

- date Date.
- time Time.
- c-ip Client IP.
- deny Denial condition.

4.3.4 OriginError Log

The OriginError log can be configured in global settings (server.xml).

```
# server.xml - <Server><Cache>

<OriginErrorLog Type="size" Unit="5" Retention="5" Warning="OFF">ON</OriginErrorLog>
```

- <OriginErrorLog> (default: OFF, Type: size, Unit: 5, Warning: OFF)

Records errors that occur in the origin server for all virtual hosts. Errors can be either connection timeouts and reception timeouts, and results of origin server exclusion/recovery are also logged.

```
#Fields: date time vhostname level s-domain s-ip cs-method cs-uri time-taken sc-error sc-resinfo
2012.11.15 07:06:10 [example.com] [ERROR] 192.168.0.13 192.168.0.13 GET /Upload/ProductImage/sto
2012.11.15 07:06:26 [example.com] [ERROR] 192.168.0.13 192.168.0.13 GET /Upload/ProductImage/sto
2012.11.15 07:06:30 [example.com] [ERROR] 192.168.0.13 192.168.0.13 GET /Upload/ProductImage/sto
#2012.11.15 07:06:30 [example.com] 192.168.0.13 excluded from service
#2012.11.15 07:06:31 [example.com] Origin server list: 192.168.0.14
#2012.11.15 07:11:11 [example.com] 192.168.0.13 recovered back in service
#2012.11.15 07:11:12 [example.com] Origin server list: 192.168.0.13
```

Fields are separated by spaces, and each field refers to the following:

- date Date of error.

- time Time of error.
- vhostname [Virtual host].
- level [Error level (Error or Warning)].
- s-domain Origin server domain.
- s-ip Origin server IP.
- cs-method HTTP Method sent by STON to the origin server.
- cs-uri URI sent by STON to the origin server.
- time-taken Amount of time elapsed until the system error.
- sc-error Type of error.
- sc-resinfo Information of server response when error occurred (separated by commas).

If the `Warning` property is set to ON, HTTP communication errors will be logged as follows.

```
2012.11.15 07:09:03 [example.com] [WARNING] 10.10.10.10 121.189.63.219 GET /716439_SM.jpg 20110
2012.11.15 07:09:03 [example.com] [WARNING] 10.10.10.10 121.189.63.219 GET /716439_SM.jpg 20110
```

HTTP communication errors can occur in the following ways.

- `ClosedWithoutResponse` Connection closed by the origin server. HTTP response was not returned.
- `ClosedWhenDownloading` Connection closed by the origin server. Desired Content-Length was not downloaded.
- `NotPartialResponseOnRangeRequest` The response code to a Range request was not 206.
- `DifferentContentLengthOnRangeRequest` The requested Range and Content-Length were different.
- `PartialResponseOnNormalRequest` The response code to a non-Range request was 206.

4.3.5 SysLog Transfer

Logs can be forwarded to UDP in real time using the `syslog` protocol. All logs can be configured to be transferred via `syslog`.

```
# server.xml - <Server><Cache>

<InfoLog SysLog="OFF">ON</InfoLog>
<DenyLog SysLog="OFF">ON</DenyLog>
<OriginErrorLog SysLog="OFF">ON</OriginErrorLog>
```

- SysLog
 - OFF (default) syslog is not used.
 - ON Uses the `<SysLog>` tag configured within the current tag to transfer logs.

The following is an example of configuring syslog when `<OriginErrorLog>` is being logged.

```
# server.xml - <Server><Cache>

<OriginErrorLog SysLog="ON">
  <SysLog Priority="local3.info" Dest="192.168.0.1:514" />
  <SysLog Priority="user.alert" Dest="192.168.0.2" />
  <SysLog Priority="mail.debug" Dest="log.example.com" />
</OriginErrorLog>
```

1. The SysLog property of <OriginErrorLog> is set to ON.
2. The <SysLog> tag is created within <OriginErrorLog>. Logs can be transferred to any number of servers.
3. The Priority property of <SysLog> is configured. This property is expressed with a combination of [Facility Levels](#) and [Severity levels](#).
4. The Dest property of <SysLog> is configured. This is the syslog reception server, and the reception port can be omitted if it is 514.

The syslog example recorded from the above settings can be seen below. The syslog tag is recorded as STON/{log name}.

```
Mar 12 11:24:24 192.168.0.1 STON/ORIGINERROR: 2013-03-12 14:09:20 [ERROR] [example.com] - 192.168.0.1
Mar 12 11:24:24 192.168.0.1 STON/ORIGINERROR: 2013-03-12 14:09:22 [ERROR] [example.com] - 192.168.0.1
Mar 12 11:24:24 192.168.0.1 STON/ORIGINERROR: 2013-03-12 14:09:24 [ERROR] [example.com] - 192.168.0.1
Mar 12 11:24:24 192.168.0.1 STON/ORIGINERROR: #2013 .03.12 14:09:24 [example.com] 192.168.0.14:102 ex
Mar 12 11:24:24 192.168.0.1 STON/ORIGINERROR: #2013 .03.12 14:09:24 [example.com] Origin server list
```

4.3.6 Saving Virtual Host Logs

Logs are recorded separately for each virtual host. Even if the log is set to OFF, *Log Trace* will work as normal.

```
# server.xml - <Server><VHostDefault>
# vhosts.xml - <Vhosts><Vhost>

<Log Dir="/cache_log">
... (omitted) ...
</Log>
```

- <Log> The Dir property configures the directory in which the logs will be recorded. The logs are saved in virtual host directories that are created under the set directory.

4.3.7 DNS Log

If the origin server is set to a Domain, the results of Resolving are recorded in the DNS log.

```
# server.xml - <Server><VHostDefault><Log>
# vhosts.xml - <Vhosts><Vhost><Log>

<Dns Type="size" Unit="10" Retention="10" SysLog="OFF" Compression="OFF">ON</Dns>
```

```
#Fields: date time domain ttl ip-list ip-count time-taken result
2014-07-30 12:10:33 example.com 157 173.194.127.15,173.194.127.23,173.194.127.24,173.194.127.31 4 50
2014-07-30 12:10:38 example.com 152 173.194.127.23,173.194.127.24,173.194.127.31,173.194.127.15 4 9 s
2014-07-30 12:11:03 example.com 127 173.194.127.31,173.194.127.15,173.194.127.23,173.194.127.24 4 150
2014-07-30 12:12:53 example.com 17 173.194.127.15,173.194.127.23,173.194.127.24,173.194.127.31 4 6 su
2014-07-30 12:23:16 test.com 0 - 0 10008 fail
2014-07-30 12:23:21 test.com 0 - 0 5007 fail
2014-07-30 12:23:26 test.com 0 - 0 5011 fail
2014-07-30 12:24:38 example.com 152 173.194.127.23,173.194.127.24,173.194.127.31,173.194.127.15 4 9 s
2014-07-30 12:25:03 example.com 127 173.194.127.31,173.194.127.15,173.194.127.23,173.194.127.24 4 150
```

Fields are separated by spaces, and each field refers to the following:

- date Date.

- `time` Time.
- `domain` Target domain.
- `ttl` Time to live (Time when record is valid).
- `ip-list` IP list.
- `ip-count` IP count.
- `time-taken` Runtime.
- `result` “success” or “fail”.

4.3.8 Access Log

Records the HTTP transactions of all clients. The log is recorded when an HTTP transaction ends, whether the transfer is completed or interrupted.

```
# server.xml - <Server><VHostDefault><Log>
# vhosts.xml - <Vhosts><Vhost><Log>

<Access Type="time" Unit="1440" Retention="10" XFF="on" Form="ston" Local="Off">ON</Access>
```

- `XFF`
 - `ON` (default) Records values of the XFF (X-Forwarded For) header sent by the client together with the client IP. If there is no header, it will be the same as `OFF`.
 - `OFF` Records the client IP.
 - `TrimCIP` Records the client IP if there is no XFF header, but records the XFF header without the client IP if there is a header.
- `Form`
 - `ston` (default) W3C standard + expansion field
 - `apache` Apache format
 - `iis` IIS format
 - `custom` *admin-log-access-custom*
- `Local`
 - `OFF` (default) Local communications (loopback) are not logged).
 - `ON` Local communications (loopback) are logged.

```
#Fields: date time s-ip cs-method cs-uri-stem cs-uri-query s-port cs-username c-ip cs(User-Agent) sc-
2012.06.27 16:52:24 220.134.10.5 GET /web/h.gif - 80 - 61.50.7.9 Chrome/19.0.1084.56 200 98141 5 - By
2012.06.27 16:52:26 220.134.10.5 GET /favicon.ico - 80 - 61.50.7.9 Chrome/19.0.1084.56 200 949 2 - -
2012.06.27 17:00:06 220.168.0.13 GET /setup.Exe - 80 - 61.168.0.102 Mozilla/5.0+(Windows+NT+6.1;+W
```

Fields are separated by spaces, and each field refers to the following:

- `date` Date of HTTP transaction completion.
- `time` Time of HTTP transaction completion.
- `s-ip` Server IP.
- `cs-method` HTTP Method sent by the client.
- `cs-uri-stem` URL sent by the client (excluding QueryString).

- `cs-uri-query` QueryString of the URL sent by the client.
- `s-port` Server port.
- `cs-username` Client username.
- `c-ip` Client IP. If `XFF` is set to `ON`, the `X-Forwarded-For` header is recorded with the IP.
- `cs (User-Agent)` HTTP User-Agent sent by the client.
- `sc-status` Server response code.
- `sc-bytes` Bytes sent from the server (header + content).
- `time-taken` Total elapsed time until an HTTP transaction is completed (ms).
- `cs-referer` HTTP Referer sent by the client.
- `sc-resinfo` Additional information, separated by the “+” character. If the service provides encoded content, the encoding option (gzip or deflate) is specified. For secured communications, the SSL protocol version (SSL3, TLS1, TLS1.1, TLS1.2) is specified. For bypassed communications, “Bypass” is specified.
- `cs-range` Records the Range header sent by the client.
- `sc-cachehit` Cache HIT results.
- `cs-acceptencoding` Accept-Encoding header sent by the client.
- `session-id` HTTP client session ID (unsigned int64).
- `sc-content-length` Value of server response Content-Length header.

The access log records all HTTP transactions regardless of the success or failure of the transfer. An HTTP transaction begins when a client sends an HTTP request. If the HTTP connection is closed before STON sends a response to the client, then the transaction will still be considered completed. Both `sc-status` and `sc-bytes` will be recorded as 0. A log like this is recorded when the client closes the connection before the STON receives a response from the origin server.

4.3.9 Custom Access Log Format

The format of the access log can be customized.

```
# server.xml - <Server><VHostDefault><Log>
# vhosts.xml - <Vhosts><Vhost><Log>

<Access Form="custom">ON</Access>
<AccessFormat>%a %A %b id=%{userid}C %f %h %H "%{user-agent}i" %m %P "%r" %s %t %T %X %I %O %R %e %S
```

- `<Access>` The `Form` property is set to `custom`.
- `<AccessFormat>` The custom log format.

With the above configuration, the access log will be recorded as follows. (#Fields are not recorded.)

```
192.168.0.88 192.168.0.12 163276 id=winesoft; image.jpg example.com HTTP "STON" GET 80 "GET /ston/ima
192.168.0.88 192.168.0.12 63276 id=winesoft; vod.mp4 example.com HTTP "STON" POST 80 "GET /ston/vod.r
192.168.0.88 192.168.0.12 3634276 id=ston; news.html example.com HTTPS "STON" GET 443 "GET /news.html
192.168.0.88 192.168.0.12 6332476 id=winesoft; style.css example.com HTTP "STON" HEAD 80 "GET /style
192.168.0.88 192.168.0.12 6276 id=ston; ui.js example.com HTTP "STON" GET 80 "GET /ui.js HTTP/1.1" 20
192.168.0.88 192.168.0.12 626 id=winesoft; hls.m4u8 example.com HTTP "STON" GET 80 "GET /hls.m4u8 HT
```

This configuration was developed based on the [Apache log format](#) and there are several expansion fields. There is no restriction to the delimiters that can be used, but quotation marks (“...”) should be used for items that may include spaces, such as a User-Agent.

- `%.a` Client IP.

```
192.168.0.66
```

- `%.A` Server IP Address.

```
192.168.0.14
```

- `%.b` Byte size of transfer, excluding the HTTP header.

```
1024
```

- `%.{foobar}C` The content of the “foobar” cookie in the request received by the server.

```
If input as %{id=}c, the cookie value corresponding to id= is recorded.
```

- `%.D` Elapsed time to process the request (ms).

```
3000
```

- `%.f` File name.

```
If /mp4/iu.mp4, iu.mp4 will be recorded.
```

- `%.h` HostName.

```
example.com
```

- `%.H` Request protocol.

```
http or https
```

- `%.{foobar}i` The content of the “foobar” header in the request received by the client.

```
If input as %{User-Agent}i, the User-Agent value is recorded.
```

- `%.m` Request Method.

```
GET or POST or HEAD
```

- `%.P` Server PORT

```
80
```

- `%.q` QueryString

```
Id=10&value=20
```

- `%.r` The first line of the request (Request Line).

```
GET /img.jpg HTTP/1.1
```

- `%.s` Response code.

```
200
```

- `%.t` STON default time format.

```
2014-01-01 15:27:02
```

- `%.{format}t` Date shown using “format”.

```
If input as %{%Y-%m-%d %H:%M:%S}T, the output will be 2014-08-07 06:12:23.
```

- %...T TimeTaken (sec).

10

- %...U ShortURI.

/img/img.jpg

- %...u FullURI.

/img/img.jpg?session=1232&id=37

- %...X Status when transaction is completed.
 - X Closed before the response is completed.
 - C Response is completed.

C

- %...I Received bytes, including the request header.

2048

- %...O Received bytes, including the response header.

2048

- %...R Response time (ms).

2

- %...e Session-ID.

1

- %...S Cache HIT results.

TCP_HIT

- %...K Request HTTP version.

HTTP/1.1

- %...y Request HTTP header size.

488

- %...z Response HTTP header size.

362

If there is no value for the configured field, it will be recorded as “-”. If the format is wrong, the STON default format (Form=”ston”) will be used instead.

The “...” in the above notations for the fields (e.g. %h %U %r %b) can be left blank, or a condition for recording can be used. If the condition is not met, “-” will be recorded. HTTP status codes can be used for conditions, and exclamation points (!) can be used for NOT conditions.

The following example only records a User-Agent when there is a 400 (Bad Request) or a 501 (Not Implemented) response.

"%400,501{User-agent}i"

The following example logs Referers for all abnormal responses.

```
"%!200,304,302{Referer}i"
```

4.3.10 Origin Log

Records all HTTP transactions in the origin server. The log is recorded when an HTTP transaction ends, whether the transfer is completed or interrupted.

```
# server.xml - <Server><VHostDefault><Log>
# vhosts.xml - <Vhosts><Vhost><Log>

<Origin Type="time" Unit="1440" Retention="10" Local="Off">ON</Origin>
```

```
#Fields: date time cs-sid cs-tcount c-ip cs-method s-domain cs-uri s-ip sc-status cs-range sc-sock-e
2012.06.27 17:40:00 357 899 192.168.0.13 GET i.example.com /t/2.gif 115.71.9.136 200 - - 3874 197 2
2012.06.27 17:40:00 357 900 192.168.0.13 GET i.example.com /ex1.gif 115.71.9.136 200 - - 5673 223 2
2012.06.27 17:40:00 357 901 192.168.0.13 GET i.example.com /exB.jpg 115.71.9.136 200 - - 8150 189 2
#[ERROR:01] 2012.06.27 17:40:01 220.73.216.5 220.73.216.5 GET /web/nmb/img/main/v1/h1.gif 1824 Connec
2012.06.27 17:40:00 357 901 192.168.0.13 GET i.example.com /exB1.jpg 115.71.9.136 200 - - 8150 189
2012.06.27 17:40:00 357 901 192.168.0.13 GET i.example.com /exB2.jpg 115.71.9.136 200 - - 8150 189
2012.06.27 17:40:00 357 901 192.168.0.13 GET i.example.com /exB3.jpg 115.71.9.136 200 - - 8150 189
```

If there is an origin server failure, an error log starting with `#[ERROR:xx]` will be recorded. Fields are separated by spaces, and each field refers to the following:

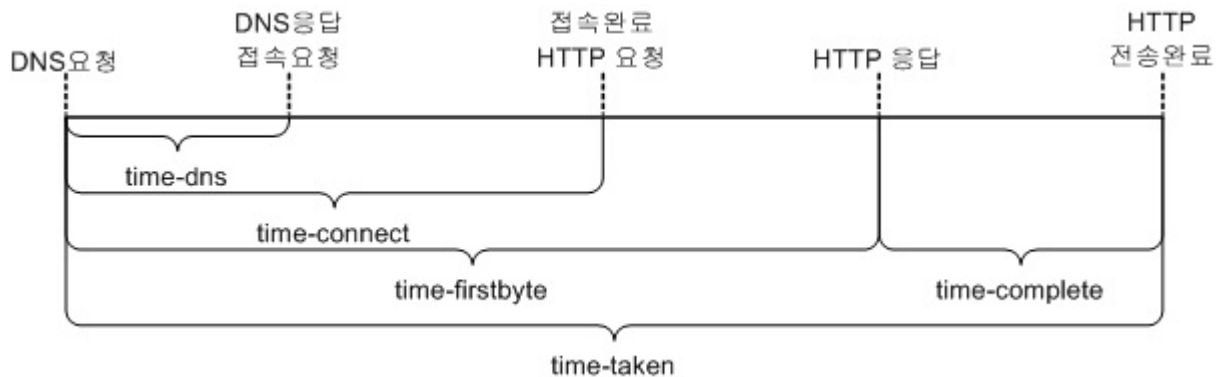


Fig. 4.4: Origin time measurements

- `date` Date of HTTP transaction completion.
- `time` Time of HTTP transaction completion.
- `cs-sid` Session unique ID. HTTP transactions processed (recycled) by the same session will have the same value.
- `cs-tcount` Transaction count. Records how many transactions the current session has processed. Transactions with the same `cs-sid` value cannot have the same `cs-tcount` value.
- `c-ip` STON IP.
- `cs-method` HTTP Method sent to the origin server.
- `s-domain` Origin server domain.
- `cs-uri` URI sent to the origin server.
- `s-ip` Origin server IP.

- `sc-status` Origin server HTTP response code.
- `cs-range` Value of Range request sent to the origin server.
- `sc-sock-error` Socket error code (1=Transfer timeout, 2=Transfer delay, 3=Connection close).
- `sc-http-error` Log of response code when origin server returns either 4xx or 5xx responses.
- `sc-content-length` Content Length sent by the origin server.
- `cs-requestsize` (unit: bytes) Size of the HTTP request header sent to the origin server.
- `sc-responsesize` (unit: bytes) Size of the HTTP header of the origin server's response.
- `sc-bytes` (unit: bytes) Received content size (header excluded).
- `time-taken` (unit: ms) Total elapsed time until the HTTP transaction is completed. If the session is not recycled, the socket connection time is included.
- `time-dns` (unit: ms) Elapsed time during DNS query.
- `time-connect` (unit: ms) Elapsed time until a socket is established with the origin server.
- `time-firstbyte` (unit: ms) Elapsed time from a sent request to a received response.
- `time-complete` (unit: ms) Elapsed time from the first response to completion.
- `cs-reqinfo` Additional information. Separated by the "+" character. Recorded as "Bypass" for bypass communications and "PrivateBypass" for private bypass communications.
- `cs-acceptencoding` Recorded as "gzip+deflate" if compressed content is requested from the origin server.
- `sc-cachecontrol` Cache-control header sent by the origin server.
- `s-port` Origin server port.
- `sc-contentencoding` Content-Encoding header sent by the origin server.
- `session-id` HTTP client session ID that created the origin server request (unsigned int64).
- `session-type` Session type requested from the origin server.
 - `cache` Sessions used for caching.
 - `recovery` Sessions used for recovery in *Error Detection and Recovery*.
 - `healthcheck` Sessions used by *Health-Checker*.

4.3.11 Monitoring Log

Records the average statistics of the last five minutes.

```
# server.xml - <Server><VHostDefault><Log>
# vhosts.xml - <Vhosts><Vhost><Log>

<Monitoring Type="size" Unit="10" Retention="10" Form="json">ON</Monitoring>
```

- `Form` Assigns the log format. (json or xml)

4.3.12 FileSystem Log

Records all File I/O transactions generated using the *Chapter 17. File System*.

```
# server.xml - <Server><VHostDefault><Log>
# vhosts.xml - <Vhosts><Vhost><Log>

<FileSystem Type="time" Unit="1440" Retention="10">ON</FileSystem>
```

Logging occurs when the File I/O transaction is completed. The time of completion differs based on the type of cs-method.

```
#Fields: date time cs-method cs-path sc-status sc-bytes response-time time-taken sc-cachehit attr ses
2012.06.27 16:52:24 ATTR /t 200 0 100 100 TCP_HIT FOLDER 1
2012.06.27 16:52:24 ATTR /t/2.gif 200 0 100 100 TCP_HIT FILE 1
2012.06.27 16:52:24 OPEN /file.txt 200 0 100 2000 TCP_HIT FILE 2
2012.06.27 16:52:24 READ /file.txt 200 1024768 100 2000 TCP_HIT FILE 2
```

- **date** Date of File I/O transaction completion.
- **time** Time of File I/O transaction completion.
- **cs-method** File I/O access type. One of the following three can be used:
 - ATTR getattr function call. Logs when the function is returned.
 - OPEN File is opened but not READ. Logs when the file is closed.
 - READ File is opened and READ. Logs when the file is closed.
- **cs-path** Access path.
- **sc-status** Response code. The following show the failure codes, as well as the code for normal service (200).
 - 200 Normal service.
 - 301 Bypass required.
 - 302 Service denied.
 - 303 Redirect required.
 - 400 Invalid request.
 - 401 Unable to find the virtual host.
 - 402 Initialization failure from the origin server.
 - 500 Object initialization failure.
 - 501 Object open failure.
 - 502 Save path generation failure.
 - 503 Memory initialization failure.
 - 504 Emergency status.
 - 600 Timeout during file service standby.
 - 601 Timeout during file data service standby.
 - 602 File initialization failure during file service standby.
 - 603 Data initialization failure during file data service standby.
 - 701 Invalid offset.
 - 702 Specific file section load failure.
 - 703 Not enough memory.

- 704 Origin session generation failure.
- `sc-bytes` Read byte size.
- `response-time` Elapsed time from the function call to the connection to the service object.
- `time-taken` Elapsed time from the function call to the completion of the File I/O transaction.
- `sc-cachehit` Cache HIT results.
- `attr` FILE or FOLDER.
- `session-id` File I/O session ID (unsigned int64).

Note: `session-id` is assigned when the Client (HTTP or File I/O) Context is generated. In the general file process flow of Open -> Read -> Close, the Client Context is constructed at Open and destructed at Close. On the other hand the `getattr` function is an “atomic function”, so the Client Context is created/destructed every time, and a new `session-id` is always assigned.

4.3.13 FTP Transfer

When the log is rolled, it is also uploaded using the designated FTP client.

FTP Client

Configures the FTP client. Rolled logs are uploaded to the FTP server in real time.

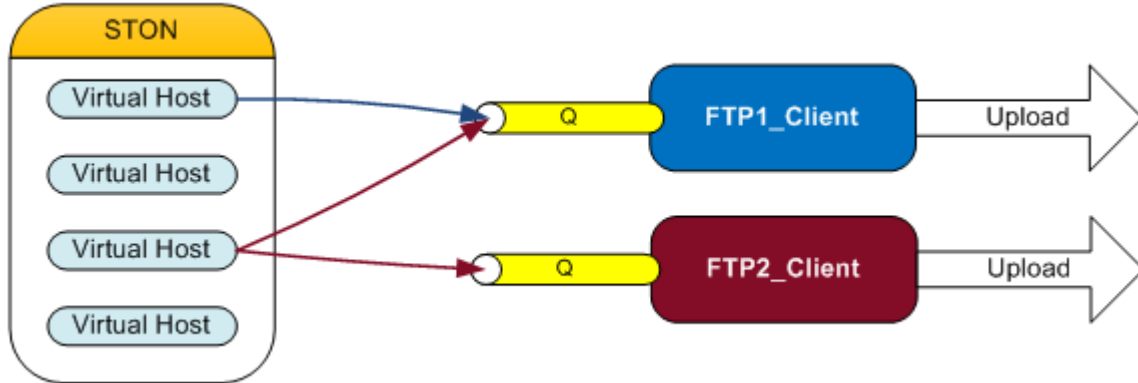


Fig. 4.5: FTP client structure and operation.

FTP clients exist outside of STON, as seen in the image above. STON is responsible for inputting logs stored locally into the FTP client queue, and has no effect on FTP operation. The FTP client can then process the uploads based on its own configuration.

The FTP clients can be configured in global settings (`server.xml`).

```
# server.xml - <Server>

<Ftp Name="backup1">
  <Mode>Passive</Mode>
  <Address>ftp.winesoft.co.kr:21</Address>
  <Account>
```

```

    <ID>test</ID>
    <Password>12345abc</Password>
</Account>
<ConnectTimeout>10</ConnectTimeout>
<TransferTimeout>600</TransferTimeout>
<TrafficCap>0</TrafficCap>
<DeleteUploaded>OFF</DeleteUploaded>
<BackupOnFail>OFF</BackupOnFail>
<UploadPath>/log_backup/%v/%s-%e.%p.log</UploadPath>
<Transfer Time="Rotate" />
</Ftp>

<Ftp Name="backup2">
  <Mode>Active</Mode>
  <Address>192.168.0.14:21</Address>
  <Account>
    <ID>test</ID>
    <Password>qwerty</Password>
  </Account>
  <ConnectTimeout>3</ConnectTimeout>
  <TransferTimeout>100</TransferTimeout>
  <TrafficCap>10240</TrafficCap>
  <DeleteUploaded>ON</DeleteUploaded>
  <BackupOnFail>ON</BackupOnFail>
  <Transfer Time="Static">04:00</Transfer>
</Ftp>

```

- <Ftp> Configures FTP clients. Individual names can be configured with the Name property.
 - Mode (default: Passive) Connection mode (Passive or Active).
 - Address FTP address.
 - Account FTP account. To encrypt the password (e.g. qwerty), the API below can be used.

```
/command/encryptpassword?plain=qwerty
```

The encrypted password can then be configured as follows.

```
<Password Type="enc">dXR9k0xNUZVVYQsK5Bilcg==</Password>
```

- ConnectTimeout Connection pending time.
- TransferTimeout Transfer pending time.
- TrafficCap (unit: KB) If set to greater than 0, the maximum transfer bandwidth is configured.
- DeleteUploaded (default: OFF) Deletes the corresponding log after transfer completion.
- BackupOnFail (default: OFF) Backs up the corresponding log in the following path so that the log is not deleted on a transfer failure.

```
/usr/local/ston/stonb/backup/
```

Backup logs are not retransmitted and will not be deleted unless done deliberately by the administrator.

- UploadPath Configures the upload path. If not configured, the path becomes “/virtual host/”. For example, logs for example.com are uploaded to the /example.com/ directory.
 - * %{time format}s Log start time.
 - * %{time format}e Log end time.

- * %p Prefix.
- * %v Virtual host name.
- * %h Device HOST name.

For example, if the following configuration is used,

```
# server.xml - <Server><Ftp>

<UploadPath>/log_backup/%v/%s-%e.%p.log</UploadPath>
```

the upload path will be as follows.

```
/log_backup/example.com/200140722_0000-200140722_2300.access.log
```

- Transfer Determines the log transfer time. The Type property determines the format of the value.
 - * Rotate (default) Transfers the log immediately after rolling. Does not require a value.
 - * Static Transfers the log once a day at a specific time. For example, if set to 04:00, a transfer will occur every day at 4 am.
 - * Interval Transfers the log after every set interval of time. For example, if set to 4, a transfer will occur every four hours.

It is important to configure a proper logging policy to ensure that rolling does not occur while logs are being transferred.

FTP clients use curl.

FTP Log

FTP logs are merged and saved in /usr/local/ston/sys/stonb/stonb.log.

```
#Fields: date time local-path cs-url file-size time-taken sc-status sc-error-msg
2014-04-23 17:10:20 /ston_log/winesoft.co.kr/origin_20140423_080000.log ftp://ftp.winesoft.co.kr:21/v
2014-04-23 17:10:20 /ston_log/winesoft.co.kr/access_20140423_1700.log ftp://192.168.0.14:21/winesoft
2014-04-23 17:11:00 /ston_log/winesoft.co.kr/origin_20140423_080000.log ftp://ftp.winesoft.co.kr:21/v
2014-04-23 17:11:00 /ston_log/winesoft.co.kr/filesystem_20140423_080000.log ftp://192.168.0.14:21/wir
```

Fields are separated by spaces, and each field refers to the following:

- date Date.
- time Time.
- local-path Local path of the log to be transferred.
- cs-url FTP address to transfer the log to.
- file-size Transfer file size.
- time-taken (unit: ms) Elapsed time of transfer.
- sc-status Transfer success/failure.
- sc-error-msg Curl error message on transfer failure.

Log FTP Transfer

When the log is rolling, the designated *FTP client* will be used to upload the log. If separated by commas, multiple *FTP clients* can be used at the same time.

```
# server.xml - <Server><VHostDefault>
# vhosts.xml - <Vhosts><Vhost>

<Log>
  <Access Ftp="backup1, backup2">ON</Access>
  <Origin Ftp="backup_org">ON</Origin>
  <Monitoring Ftp="backup1">ON</Monitoring>
  <FileSystem Ftp="backup2">ON</FileSystem>
</Log>
```

- Ftp The *FTP client* to be used.

The log is uploaded to `ftp://{FTP server address}/{virtual host name}/{rolled log name}`. For example, the upload address of the rolled log “access_20140424_0000.log” of the virtual host “example.com” in the server “ftp.dummy.com” will be `ftp://ftp.dummy.com/example.com/access_20140424_0000.log`.

4.4 Chapter 13. WM (Web Management)

This chapter will introduce Web Management (WM), a tool for web management using an API. Using WM, not only can you intuitively set up the service, you can also organize clusters to manage a large number of STON instances together.

When STON is installed, WM is installed in the `/usr/local/ston/wm` directory. WM is implemented with Apache 2.2.24 + PHP 5.3.24. Because WM uses Apache, you can change the settings (e.g. HTTPS) as desired by editing the `/usr/local/ston/wm/config/httpd.conf` file. WM and STON are not strongly connected. As in the figure below, WM uses STON configuration files and API to set the behavior of STON.

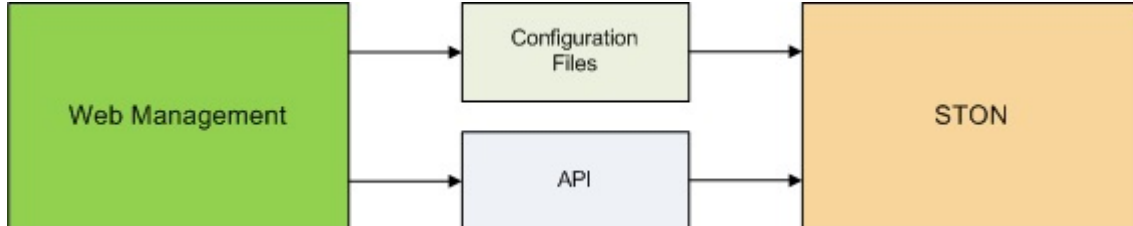


Fig. 4.6: WM uses STON configuration files and API.

There may exist better management techniques that improve on WM using a similar method.

4.4.1 Connection

WM uses port 8500 by default. If the IP of STON is set to 192.168.0.100, the WM connection address will be `http://192.168.0.100:8500`. As previously mentioned, this can be customized by editing the `httpd.conf` file.

4.4.2 Account

The default account is set to [ID: **admin**, Password: **ston**]. If the login succeeds, the dashboard page will be displayed, showing the general status of STON.

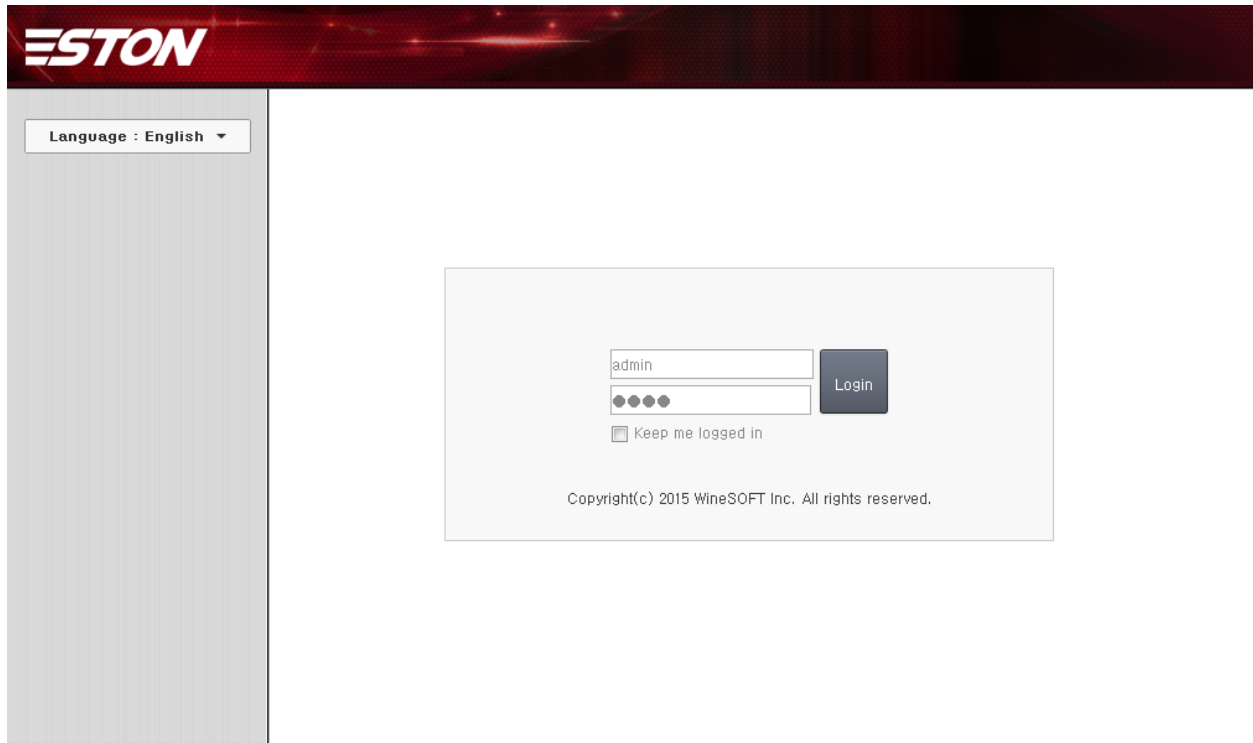


Fig. 4.7: WM Start Page

4.4.3 Update to Latest Version

When a new version is released, an “Update Available” message will be shown as seen below.

Clicking the message brings you to a page where you can update to the latest version. Depending on the status of the service, the safety level of performing the update will be shown.

When the update is completed, all services will automatically restart.

4.4.4 Menu Structure

Drop-down menus can be expanded/shrunk with mouse clicks.

1. **Global Settings** All functions except for virtual host default settings can be configured in global settings (server.xml).
2. **Virtual Host Management** You can add/suspend/delete virtual hosts and view the status of all virtual hosts in the service.
3. **Cluster** You can create/manage/destroy clusters, and all services in a cluster can be viewed by servers and services.
4. **Content Control** You can control the content in the service with functions such as Purge.
5. **Server Status** You can monitor global settings such as the system status. All graphs use global resource graphs.
6. **Service Status** You can monitor the service status of virtual hosts. All graphs use virtual host graphs.
7. **File System** STON can be mounted on Linux VFS.

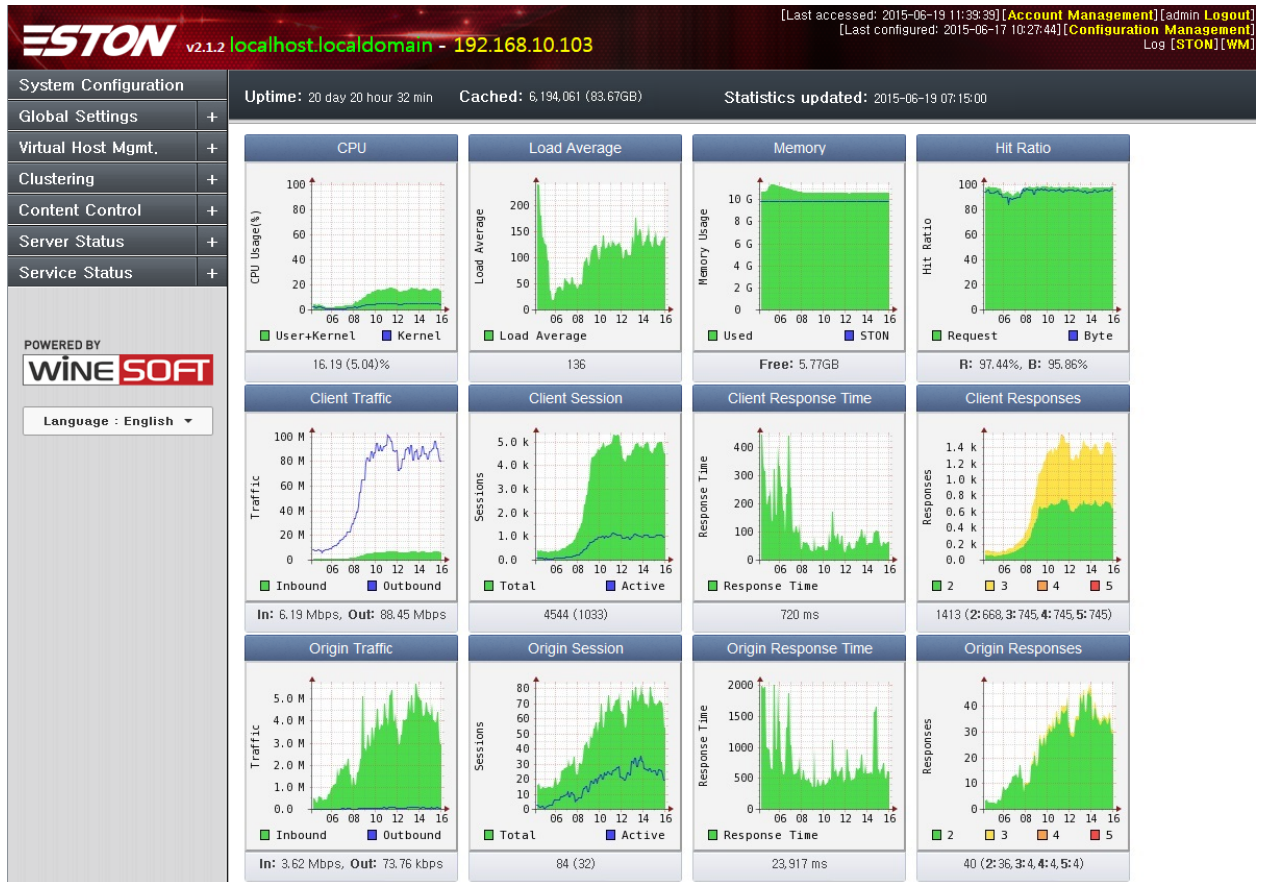


Fig. 4.8: WM Dashboard

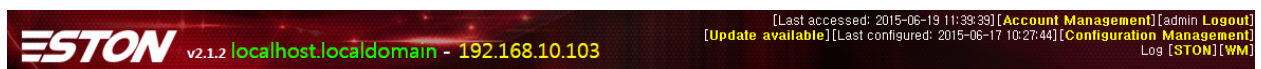


Fig. 4.9: A new update is available.

Update

Operating service found.
Outbound: 32.38 Mbps, Sessions: 50

STON version 2.1.2 is released.
Update to the latest version?

Update Later

New Features

- Web Management - English support

Feature / Policy Updates

- Single-core CPU support

Bug Fixes

- Customized module malfunction in the Memory Indexing mode

[Read more](#)

Fig. 4.10: WM Update could be dangerous.

System Configuration	
Global Settings	—
General	
Indexing	
Global Log	
Manager Access	
Service Access	
SNMP	
HTTPS Certificate	
File System	
URL Preprocessing	
View	
FTP	
Virtual Host Mgmt.	+
Clustering	+
Content Control	+
Server Status	+
Service Status	+

Fig. 4.11: WM Menu

4.4.5 Global Settings

All functions except for virtual host default settings can be configured in global settings (server.xml).

Global Settings – General
Apply
Apply to all the clustered
Refresh

General

Server Name	* Default: System Name (localhost.localdomain)
Administrator's Email	
Disk Optimization Time	02 h 00 m ☐ Delete files unaccessed for day(s)

Setting Management

Backs up all the applied settings within 30 days from today

10 settings accessible by SNMP (from 1 to 100)

Overload Management

Close all new sockets at 80000 connected sockets

Accept new sockets if the connected sockets decrease under 75 % (60000)

Do not activate emergency mode at overload

Caching Initialization
Restart STON

Fig. 4.12: WM Global Settings - General

4.4.6 Virtual Host Management

All virtual hosts in the service can be configured in detail, and new virtual hosts can be added. All virtual hosts that aren't configured independently will use the settings of the default virtual host (VHostDefault). This concept is identical to inheritance on object-oriented programming. The virtual hosts can override most parameters.

New

Creates a new virtual host for the service. If a cluster is configured, then multiple virtual hosts can be created in every server at the same time. All virtual hosts will inherit from the default virtual host (VHostDefault), so a virtual host can be ready to enter the service after only setting its name and origin server address. There are eight different sub-options that can be expanded with the **Expand** button to show more detailed settings.

List

You can monitor the status of all virtual hosts that are a part of the service. You can also start/stop each virtual host. If a cluster is configured, you can control the virtual hosts of all servers at the same time. The default virtual host can also be selected.

Virtual Host – New

Add a Virtual HostAdd a Virtual Host to the cluster

Virtual Host Name :

From () Virtual Host

Origin Server

Origin Server Management

Client

Log

Content Caching

Content Control

Bypass

Statistics

Media

DIMS

Bandwidth Control

File System

Fig. 4.13: WM Virtual Host Management - New

Virtual Host - List											
Default Virtual Host : test.com Edit Change Cluster											
Statistics Time : 2015-06-19 07:30:00											
※ STON traffic does not count TCP Headers. ※ Might differ by 8~10 percent from switch traffic.											
Virtual Host	Uptime	Hit Ratio	Bandwidth Saved	Client			Origin			Action	Del
				Response/Request	Session(Active/Total)	Outbound	Response/Request	Session(Active/Total)	Inbound		
image.winesoft.co.kr	20 day 20 hour 45 min 14 sec	97.27%	80.66%	111/112	105/680	6.01 Mbps	4/5	3/11	1.16 Mbps	Stop	Del
down.winesoft.co.kr	20 day 20 hour 45 min 14 sec	96.12%	95.85%	723/725	536/1395	59.46 Mbps	30/31	19/51	2.44 Mbps	Stop	Del
media.winesoft.co.kr	20 day 20 hour 45 min 14 sec	96.18%	94.63%	840/842	634/1522	60.28 Mbps	34/35	24/60	3.17 Mbps	Stop	Del
m.winesoft.co.kr	20 day 20 hour 45 min 14 sec	100%	98.94%	193/193	137/1379	5.35 Mbps	0/0	0/2	56.69 kbps	Stop	Del
search.winesoft.co.kr	20 day 20 hour 45 min 14 sec	100%	99.98%	280/281	204/2051	7.46 Mbps	0/0	0/1	1.35 kbps	Stop	Del
shop.winesoft.co.kr	20 day 20 hour 45 min 14 sec	97.82%	87.81%	48/48	6/8	11.63 Mbps	1/3	1/2	1.41 Mbps	Stop	Del
static.winesoft.co.kr	20 day 20 hour 45 min 14 sec	100%	95.97%	6/6	5/51	522.99 kbps	0/0	0/0	20.94 kbps	Stop	Del
help.winesoft.co.kr	20 day 20 hour 45 min 14 sec	100%	99.95%	6/6	4/20	271.89 kbps	0/0	0/0	121.99 bps	Stop	Del
support.winesoft.co.kr	20 day 20 hour 45 min 14 sec	100%	99.95%	1/1	0/5	57.67 kbps	0/0	0/0	24.26 bps	Stop	Del
admin.winesoft.co.kr	20 day 20 hour 45 min 14 sec	100%	99.99%	146/147	62/318	8.55 Mbps	0/0	0/1	670.77 bps	Stop	Del
dev.winesoft.co.kr	20 day 20 hour 45 min 14 sec	71.42%	51.07%	7/8	6/63	375.9 kbps	2/2	2/10	183.47 kbps	Stop	Del
qa.winesoft.co.kr	20 day 20 hour 45 min 14 sec	100%	98.47%	6/7	6/60	248.73 kbps	0/0	0/1	3.74 kbps	Stop	Del

Fig. 4.14: WM Virtual Host Management - List

Detailed Configuration

Here you can configure the default virtual host (VHostDefault) or individual virtual hosts. A virtual host can be selected after selecting the combo box in the upper left corner. **“Default Virtual Host”** is the default configurations that all virtual hosts will inherit. Therefore, any configurations that do not override this will be affected by changes to the “Default Virtual Host”.

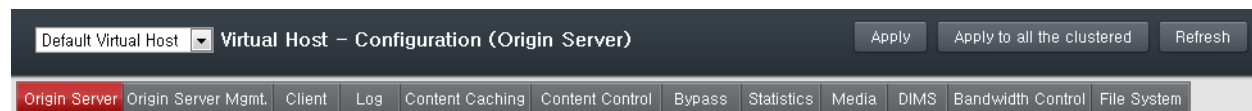


Fig. 4.15: WM Virtual Host Configuration - Top Menu

As in the above figure, there are many submenus, with the selected submenu colored red. Clicking on each menu will load a detailed configuration page, as shown in the figure below. All configurations will be reflected after “Apply” or “Apply to All Clusters” is clicked.

Almost all items in this section can be overridden, so you should have a thorough understanding of how it works. For example, if the TTL value of the default virtual host is set to 60, all virtual hosts will inherit this value. However, if this value is overridden, the corresponding virtual host will use the overridden TTL value.

The three possibilities are explained below.

- **Override with another value** While the default TTL is 60, the service for User A will use the overridden value of 180. This will not be affected by any changes to the default virtual host.
- **Override with the same value** Though the values are the same, this will be processed as an override and the service for User B will use the overridden value of 60. However, even if the default virtual host’s TTL is changed to 30, this will not affect User B’s overridden setting of 60.

Origin Server	Origin Server Mgmt.	Client	Log	Content Caching	Content Control	Bypass	Statistics	Media	DIMS	Bandwidth Control	File System
---------------	---------------------	--------	-----	-----------------	-----------------	--------	------------	-------	------	-------------------	-------------

Assign origin server address (Keyword:)

Address : Add to active Add to standby

Type	Address
Active	
Standby	

Connecting Options

Balance Mode : RoundRobin

Reuse sessions Terminate unused sessions for 60 seconds.

General

Transparent Mode : Off

Content Size Downloadable from origin: 0 MB (0 for no limit)

Caching for redirecting responses : Off

Range Request for initial content caching or update Off

Whole client request delivery for initial content caching or update Off

E-Tag from : STON (Etag: "Last-Modified Time: File Size")

HTTP Header

UserAgent Header : STON (Bypass : Off)

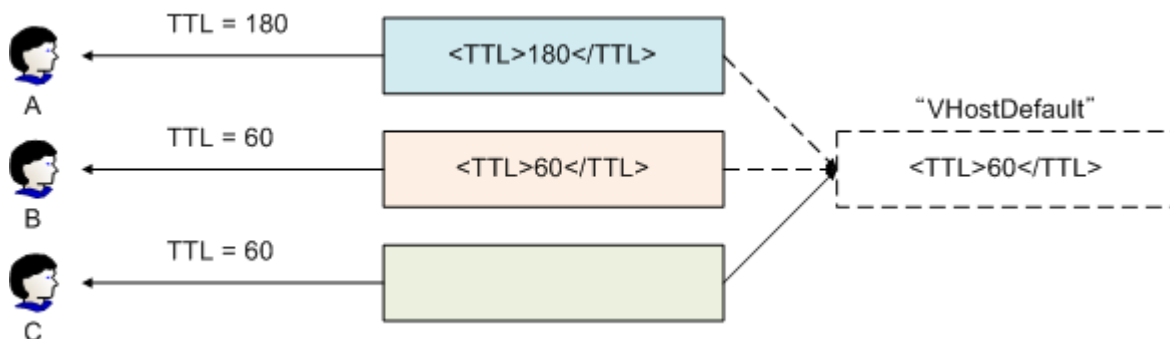
Host Header : Virtual Host Name (Bypass : On)

WL-Proxy-Client-IP Header : Off (IBM Weblogic only) : (Bypass : On)

X-Forwarded-For Header includes : All Proxy Ips (Bypass : On)

(X-Forwarded-For: 220.61.7.150, 61.1.9.100, 128.134.9.1)

Fig. 4.16: WM Virtual Host Configuration - Origin Server



- **Do not override** If a specific value is omitted, the service for User C will use the default value of 60. If the default virtual host's TTL is changed to 30, the TTL used for User C will also change to 30.

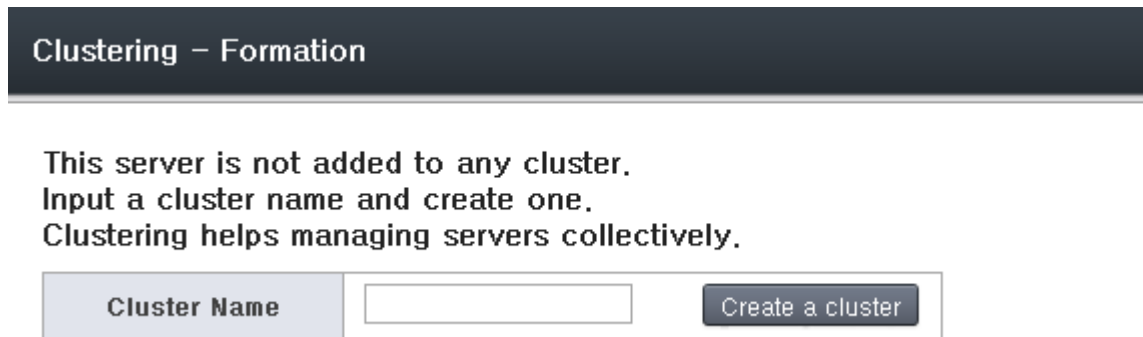
In WM, colors are used to identify overrides. A white background identifies inheritance of the default virtual host configuration. Overridden values use an apricot background to distinguish it from default values. All override settings have an X button on the right, which will remove the override when clicked.

4.4.7 Cluster

Multiple instances of STON can be merged into one cluster for integrated management/operation. All instances of STON are configured to have equal authority, so logging into any instance in the cluster will allow you to manage the entire cluster.

Structure

A cluster can be created, or a server can be added to an existing cluster. Adding to a cluster requires authentication for the WM account. If WM is configured with identical account information (ID and password), the authentication procedure is skipped.



Clustering – Formation

This server is not added to any cluster.
Input a cluster name and create one.
Clustering helps managing servers collectively.

Cluster Name

Fig. 4.17: Creating a new cluster.

When a cluster is set up, the “Apply to All Clusters” button can be used when managing virtual hosts to configure multiple hosts at the same time. In addition, you can duplicate and apply configurations from one server to another in the same server. If you want to put a server into a different cluster, it must be removed and reconfigured.

Cluster Port

When first configured, WM will use the same port as the cluster. Though it has the benefit of allowing clustering with only the WM account, this can pose problems in the restriction of access IPs.

- For security purposes, WM places a restriction so that only designated IPs can access it.
- All servers must specifically allow the IPs of the other servers to allow clustering.
- If there are too many servers or the server IPs are dynamic, it would be virtually impossible to properly fill out the IP list.

This problem can be resolved by using a separate port for clustering. Servers can then recognize each other not with a license file separate from the WM account. Clusters will only be possible between servers with the same license, which increases security.

1. [Apache server] httpd.conf Multi-port configuration

Clustering – Formation

Add a server to "stonCluster" cluster

Server IP	
Server Port	
User Account	
Password	

Add

Cluster List

Server Address	Import all settings to the server.	Duplicate this server.	Remove from the cluster
192.168.10.103 (localhost.localdomain)	Current Server	Current Server	Current Server
103.17.24.92 (QLRT001)	Clone to	Clone from	Remove
103.17.24.90 (QLRT002)	Clone to	Clone from	Remove
103.17.24.93 (QLRT003)	Clone to	Clone from	Remove
21.192.13.100 (QST010)	Clone to	Clone from	Remove
21.192.13.101 (QST011)	Clone to	Clone from	Remove
21.192.13.102 (QST012)	Clone to	Clone from	Remove
210.15.4.210 (QTC06)	Clone to	Clone from	Remove
210.15.4.215 (QTC11)	Clone to	Clone from	Remove
210.15.4.216 (QTC12)	Clone to	Clone from	Remove
192.168.10.64 (localhost)	Clone to	Clone from	Remove

[Import my setting to all clusters.](#)[Remove from test Cluster](#)[Delete test Cluster](#)

Fig. 4.18: Cluster list.

(For default installation) Open `/usr/local/ston/wm/conf/httpd.conf` and add ports as shown below.

```
ServerRoot "/usr/local/ston/wm"
ServerName ston_wm

Listen 8500
Listen 8501

LoadModule php5_module      modules/libphp5.so

<IfModule !mpm_netware_module>
<IfModule !mpm_winnt_module>

User winesoft
Group winesoft

</IfModule>
</IfModule>

ServerAdmin ston.cs@winesoft.co.kr
```

After saving, restart the Apache server to put the changes into effect.

2. [WM] Clustering

If multi-port configuration was successful, the “Allocate Clustering Port” button can be found.

Clustering – Formation

This server is not added to any cluster.
Input a cluster name and create one.
Clustering helps managing servers collectively.

Cluster Name		Create a cluster
--------------	----------------------	------------------

Current server to add

Server IP	192.168.10.103
Clustering Port	8501

Allocate Clustering Port

Click the button.

3. [WM] Cluster Port Selection

A list of ports that can be used will be shown. Select a port to configure.

All servers in the same cluster must use the same port.

Allocate Clustering Port

Please select Clustering Port.

	☐ No separate Clustering Port
Separate	☐ 8500 (Connected Port)
	☒ 8501
※ Multi-ports configured at Apache	

OK

Server Status

The status and service condition of all STON servers in a cluster can be checked. You can click on each item in the server list to view detailed information.

Virtual Host Status

The MRTG of all virtual hosts in the cluster can be put together in one screen and checked. The virtual hosts can all be started/stopped at the same time. You can click on each item in the virtual host list to view detailed information.

4.4.8 Content Control

You can browse/control content currently in the service or perform cleanup. If a cluster is configured, content in all of the servers can be browsed or controlled at the same time.

4.4.9 System Information

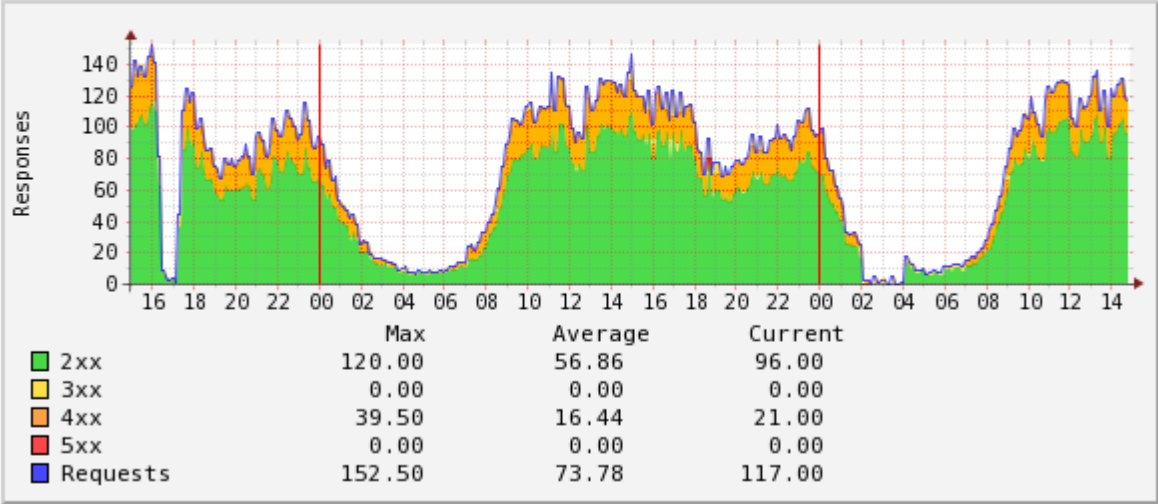
You can look into the system information of the server in operation.

4.4.10 Service Status

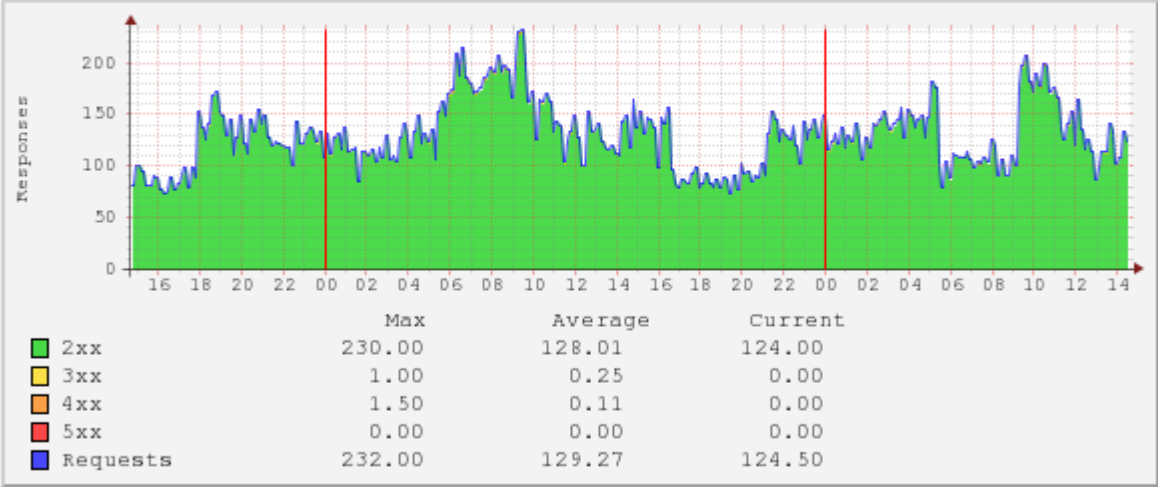
You can monitor the service status of each virtual host.

Clustering – Client Responses (Detail)

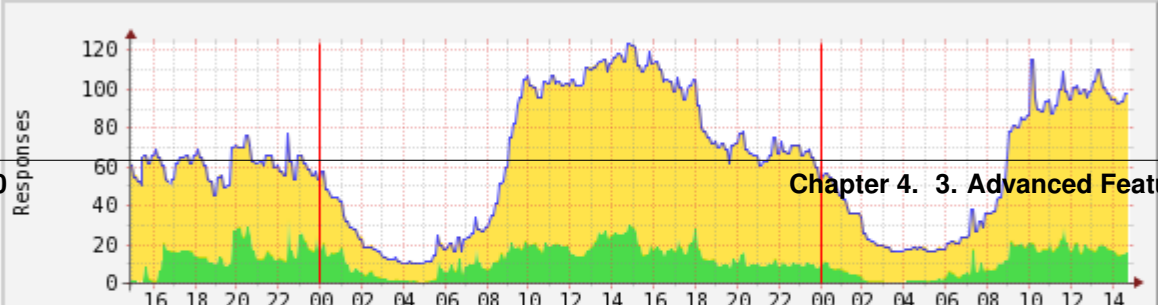
103.17.24.90 – Daily (48 Hours, 5 Min Average)



103.17.24.92 – Daily (48 Hours, 5 Min Average)



103.17.24.93 – Daily (48 Hours, 5 Min Average)



Clustering – Virtual Host Status

Virtual Host	Hit Ratio(Request)	Hit Ratio(Traffic)	Client			Origin			Execute	Del
			Response/Request	Session (Active/Total)	Outbound	Request/Response	Session (Active/Total)	Inbound		
image.winesoft.co.kr	93.01%	88.43%	101/103	9/11	21.55 Mbps	4/5	3/11	2.49 Mbps	Start Stop	Del
down.winesoft.co.kr	89.52%	80.16%	31/31	10/11	7.94 Mbps	3/5	8/13	1.58 Mbps	Start Stop	Del
media.winesoft.co.kr	80.68%	67.96%	23/23	7/8	18.61 Mbps	1/2	0/1	5.96 Mbps	Start Stop	Del
m.winesoft.co.kr	71.44%	60.55%	85/88	9/11	17.90 Mbps	1/2	5/12	7.06 Mbps	Start Stop	Del
search.winesoft.co.kr	78.60%	72.03%	121/123	20/21	27.23 Mbps	16/18	20/30	7.62 Mbps	Start Stop	Del
shop.winesoft.co.kr	90.10%	85.46%	61/65	12/13	16.85 Mbps	3/4	0/5	2.45 Mbps	Start Stop	Del
static.winesoft.co.kr	33.33%	32.16%	101/101	21/22	18.62 Mbps	21/21	17/28	12.63 Mbps	Start Stop	Del
help.winesoft.co.kr	100.00%	94.06%	50/50	9/11	9.67 Mbps	4/4	0/0	574.40 kbps	Start Stop	Del
support.winesoft.co.kr	98.78%	90.71%	72/73	12/14	11.03 Mbps	4/5	1/2	1.02 Mbps	Start Stop	Del
admin.winesoft.co.kr	95.18%	78.19%	85/85	14/17	13.32 Mbps	6/7	1/3	4.21 Mbps	Start Stop	Del
dev.winesoft.co.kr	82.74%	77.44%	141/142	16/18	36.44 Mbps	19/22	23/38	8.22 Mbps	Start Stop	Del
qa.winesoft.co.kr	99.53%	95.23%	156/159	20/21	43.03 Mbps	12/14	16/20	2.05 Mbps	Start Stop	Del
Total	84.41%	76.87%	1027/1043	159/178	248.19 Mbps	94/109	100/163	55.86 Mbps		

Fig. 4.20: Status for each virtual host.

Content Control – Caching Status Check

 /img/sub/2013/06/12/sr.jpg

Current Server

Caching Status			
URL	http://image.winesoft.co.kr/img/sub/2013/06/12/sr.jpg		Download
Local Path	/cache1/image.winesoft.co.kr/00/00/00/05.bin	File Creation Time	[2015.06.19 10:49:24, -7]
File Size	1626842	Cached File Size	1626842
Accept-Encoding	Y	Last-Modified	[2015.02.24 05:45:23, -9950648]
Response Code	200	Content-Type	image/jpeg
cache-control	[not-specified]	Compression	-
TTL Remained	[2015.06.19 11:19:25, 1794]	Transfer-Encoding	N
Redirect-Location	-	Content-Disposition	-
Caching Udate Time	[2015.06.19 10:49:25, -6]	Last Accessed Time	[2015.06.19 10:49:24, -7]
Expiration Time	[0, 0]	Status	Cached
ETag	"54ebdebb:18d2da"	Referral Time	3
Disk Index	0	File ID	4194318
Opened (File)	Y	Updating (File)	-
Downloaders	0	Custom TTL	-
Reserved to Destroyed	N	Local File	Y
Small-sized file Recognition	N	Deleted	N
Size-sensitive	Y	Purged	N
Ignore-IMS	N	No-Cache	N
Age Header Guideline	0	Origin Header	-
Split File	N		

Fig. 4.21: Caching status check

Content Control – Caching Management

image.winesoft.co.kr

Current Server :

Cluster :

Processed Result – Purge

Current Server					
URL	http://image.winesoft.co.kr/img/sub/2013/06/12/sr.jpg				
Files	2	Size	4.4 KB	Time Spent	0 ms

Fig. 4.22: API call (e.g. Purge)

Server Status – System Information

Version

OS	Linux version 2.6.32-358.el6.x86_64 (mockbuild@cb8.bsys.dev.centos.org) (gcc version 4.4.7 20120313 (Red Hat 4.4.7-3) (GCC)) #1 SMP Fri Feb 22 00:31:26 UTC 2013
STON	2.1.2

CPU

Cores	4
Model	Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz
CPU MHz	3601.000
Cache Size	6144 KB

Memory

Physical Memory	16 GB
-----------------	-------

NIC(1)

Device	Model	IP	MAC
eth2	Intel Corporation 82545EM Gigabit Ethernet Controller (Copper) (rev 01)	192.168.10.103	00:0c:29:18:17:fd

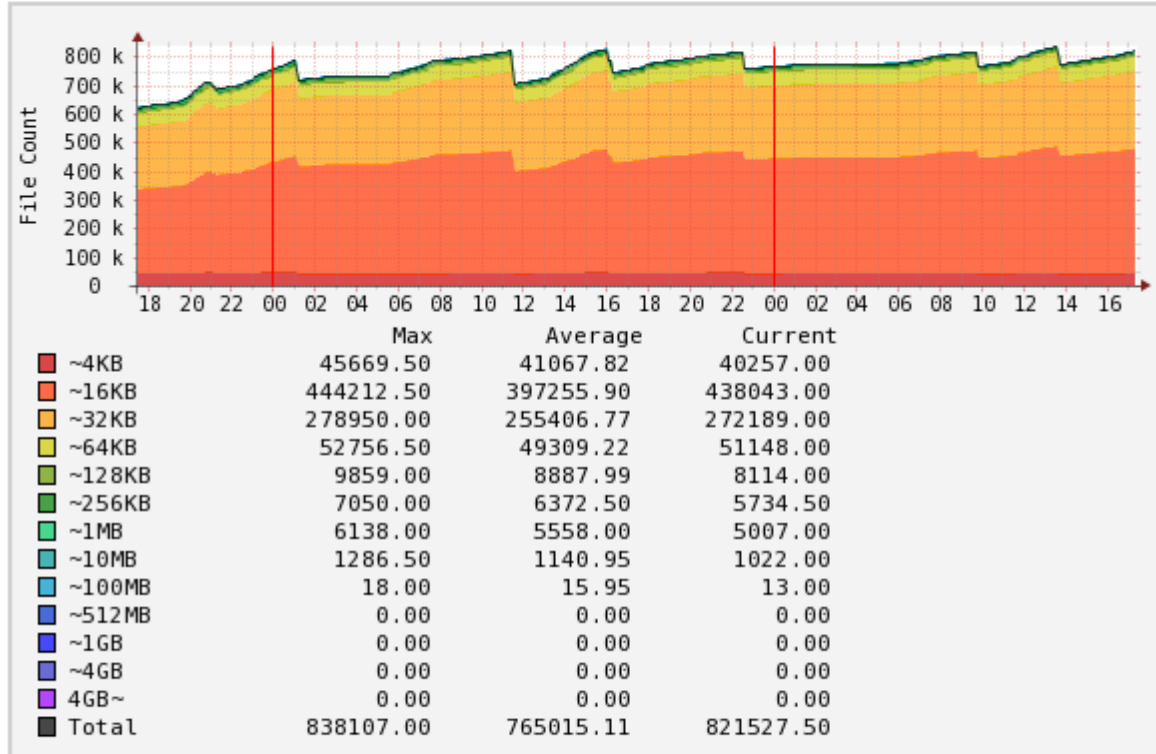
Disk

Device	Model	Total Space	Used Space
sda	DELL PERC 6/i (scsi)	105705.98 MB	24143.99 MB
sdb	DELL PERC 6/i (scsi)	105705.98 MB	35587.29 MB
sdc	DELL PERC 6/i (scsi)	105705.98 MB	23442.89 MB
sdd	DELL PERC 6/i (scsi)	105705.98 MB	24196.58 MB
sde	DELL PERC 6/i (scsi)	105705.98 MB	24268.41 MB

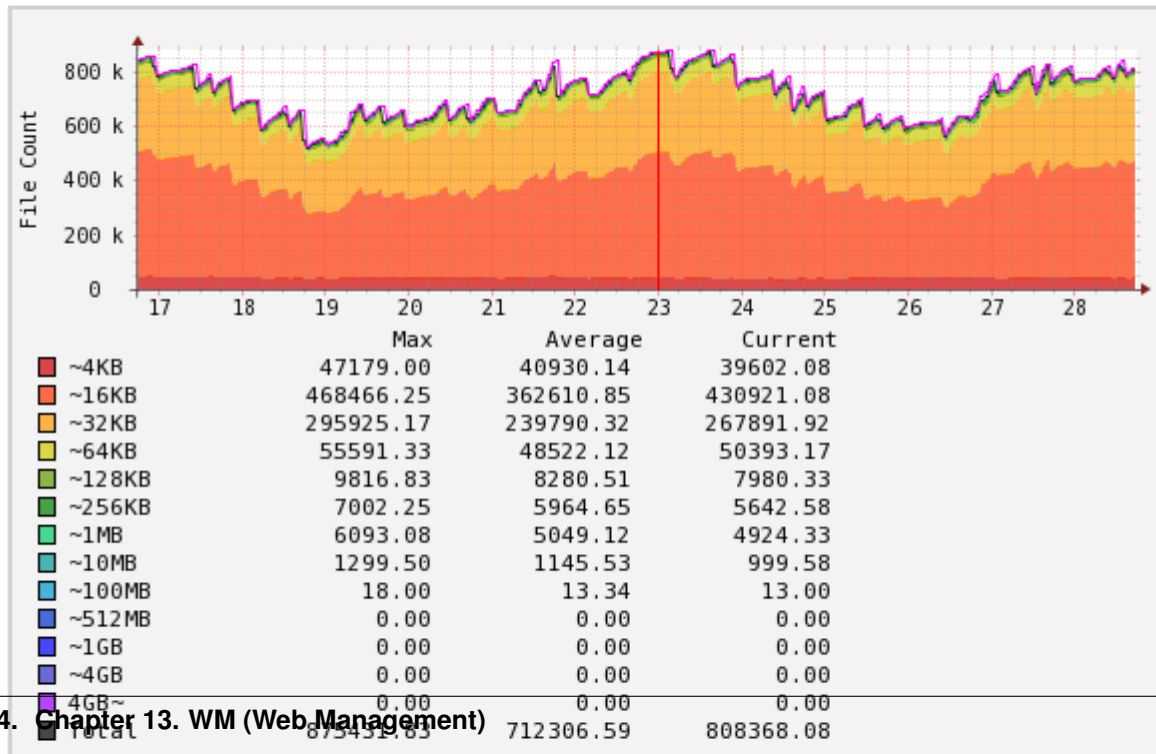
image.winesoft.co.kr

Service Status – Cached File Size Distribution

Daily (48 Hours, 5 Min Average)



Weekly (12 Days, 30 Min Average)



4. Management & Operation

5.1 Chapter 14. Advanced Virtual Host Configuration

This chapter will discuss advanced topics related to virtual host configuration.

Each virtual host is supposed to match its origin. (a domain or an IP) But sometimes a virtual host has to be connected to others, or the other way round. Depending on functionalities, *Client Statistics* / *Access Log* policies might be different.

5.1.1 URL Preprocessing

Regular expressions are used to modify the requested URLs. If URL preprocessing is defined, all client requests (HTTP or File I/O) must pass through the URL Rewriter.



Fig. 5.1: The request can only reach the virtual host by passing through the URL Rewriter.

If an approaching Host name is modified by the URL Rewriter, then it will consider it as if the Host header was modified by the client's HTTP request. URL preprocessing is configured in the virtual host settings (vhosts.xml). While most settings are under the virtual host, URL preprocessing can change the name of the Host requested by the client, so the settings must be on the same level as the virtual host.

```
# vhosts.xml

<Vhosts>
  <Vhost ...> ... </Vhost>
  <Vhost ...> ... </Vhost>
  <URLRewrite ...> ... </URLRewrite>
  <URLRewrite ...> ... </URLRewrite>
</Vhosts>
```

Multiple configurations are allowed, and the regular expressions will be checked in order.

```
# vhosts.xml - <Vhosts>

<URLRewrite AccessLog="Replace">
  <Pattern>www.example.com/ ([^/]+)/ (.*) </Pattern>
```

```
<Replace>#1.example.com/#2</Replace>
</URLRewrite>
```

- <URLRewrite>

Configures URL preprocessing. AccessLog (default: Replace) Configures URLs that will be recorded in the Access log. Replace records URLs after processing (/logo.jpg), while Pattern records URLs after processing (/baseball/logo.jpg).

- <Pattern> Configures the patterns to be matched. A single pattern is expressed with parentheses ().
- <Replace> Configures the conversion format. Patterns that match can be used with expressions like #1 and #2. #0 stands for the entire requested URL. A maximum of nine patterns (up to #9) can be configured.

Throughput is provided by [Chapter 10. Monitoring & Statistics](#) and can also be checked via [Successful URL Preprocessing](#). URL preprocessing can work alongside [Trimming](#) and [MP4 HLS](#) to simplify expressions further.

```
# vhosts.xml - <Vhosts>

<URLRewrite>
  <Pattern>example.com/([^\/]+)/(.*)</Pattern>
  <Replace>example.com/#1.php?id=#2</Replace>
</URLRewrite>
// Pattern : example.com/releasesnotes/1.3.4
// Replace : example.com/releasesnotes.php?id=1.3.4

<URLRewrite>
  <Pattern>example.com/download/(.*)</Pattern>
  <Replace>download.example.com/#1</Replace>
</URLRewrite>
// Pattern : example.com/download/1.3.4
// Replace : download.example.com/1.3.4

<URLRewrite>
  <Pattern>example.com/img/(.*\.(jpg|png).*)</Pattern>
  <Replace>example.com/#1/STON/composite/watermark1</Replace>
</URLRewrite>
// Pattern : example.com/img/image.jpg?date=20140326
// Replace : example.com/image.jpg?date=20140326/STON/composite/watermark1

<URLRewrite>
  <Pattern>example.com/preview/(.*)\.(mp3|mp4|m4a)$</Pattern>
  <Replace><![CDATA[example.com/#1.#2?&end=30&boost=10&bandwidth=2000&ratio=100]]></Replace>
</URLRewrite>
// Pattern : example.com/preview/audio.m4a
// Replace : example.com/audio.m4a?end=30&boost=10&bandwidth=2000&ratio=100

<URLRewrite>
  <Pattern>example.com/(.*)\.mp4\.m3u8$</Pattern>
  <Replace>example.com/#1.mp4/mp4hls/index.m3u8</Replace>
</URLRewrite>
// Pattern : example.com/video.mp4.m3u8
// Replace : example.com/video.mp4/mp4hls/index.m3u8

<URLRewrite>
  <Pattern>example.com/(.*)_(.*)_(.*)</Pattern>
  <Replace>example.com/#0/#1/#2/#3</Replace>
</URLRewrite>
// Pattern : example.com/video.mp4_10_20
// Replace : example.com/example.com/video.mp4_10_20/video.mp4/10/20
```

If one of the five special XML characters are used, then the pattern must be surrounded with a `<![CDATA[ ... ]]>` tag. If configured using [Chapter 13. WM \(Web Management\)](#), all patterns are processed as CDATA.

5.1.2 Facade Virtual Host

Because `<Alias>` is just a nickname for the virtual host, it will not provide separate statistics and logs. If you want to use the same virtual host but obtain different [Client Statistics](#) and [Access Log](#) depending on the domain, a Facade Virtual Host can be configured.

```
# vhosts.xml - <Vhosts>

<Vhost Name="example.com">
    ...
</Vhost>

<Vhost Name="another.com" Status="facade:example.com">
    ...
</Vhost>
```

This can be done by inputting `facade: + virtual host` into the `Status` property. In the previous example, the [Client Statistics](#) and [Access Log](#) will be recorded for clients that request `another.com`, not `example.com`.

5.1.3 Sub-Path

A single virtual host can have different sub-paths. These sub-paths can be configured to be handled by separate virtual hosts.

```
# vhosts.xml - <Vhosts>

<Vhost Name="sports.com">
  <Sub Status="Active">
    <Path Vhost="baseball.com">/baseball/<Path>
    <Path Vhost="football.com">/football/<Path>
    <Path Vhost="photo.com">/*.jpg<Path>
  </Sub>
</Vhost>

<Vhost Name="baseball.com" />
<Vhost Name="football.com" />
<Vhost Name="photo.com" />
```

- If the page path or pattern matches the `<Sub>` input, then it will be sent to the corresponding virtual host. If they do not match, then the page will be handled by the current virtual host.
 - `Status` (default: `Active`) Sub-paths are ignored when inactive.
 - `<Path>` If the URI requested by the client and the path match, the request will be sent to `Vhost`. Only paths or patterns are allowed.

```
<Path Vhost="baseball.com">baseball<Path>
<Path Vhost="photo.com">*.jpg<Path>
```

If input as above, they will be parsed as `/baseball/` and `/*.jpg`, respectively.

For example, if the client requests the following, the request will be sent to the `football.com` virtual host.

```
GET /football/rank.html HTTP/1.1
Host: sports.com
```

5.1.4 Redirect Tracing

When the origin server responds with the Redirect responses (301, 302, 303, 307), the location header is tracked to request the content.

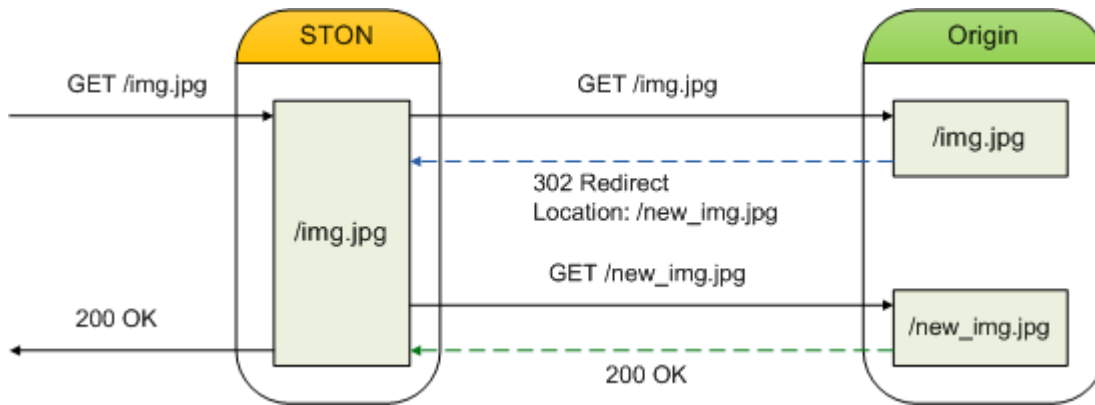


Fig. 5.2: The redirection is hidden from the client.

```
# server.xml - <Server><VHostDefault><OriginOptions>
# vhosts.xml - <Vhosts><Vhost><OriginOptions>

<RedirectionTrace>OFF</RedirectionTrace>
```

- <RedirectionTrace>
 - OFF (default) Saved as 3xx responses.
 - ON Downloads content from the address specified in the location header.

Works only if matched to the format or with a valid Location header. Traced only once to prevent infinite redirects.

5.1.5 Virtual Host Link

Even if the content is distributed across multiple origins, the service is still operable as if the content were integrated using virtual host links. It is particularly useful in environments where content is scattered due to on-premise to cloud storage migration, storage capacity, and cost.

```
# vhosts.xml - <Vhosts><Vhost>

<VhostLink Condition="...">...</VhostLink>
```

- <VhostLink> The virtual host name to delegate requests to. If the original response to the content satisfies “Condition”, the request is delegated to the specified virtual host. Only one can be set.
 - Condition HTTP response code / pattern (1xx, 2xx, 3xx, 4xx, 5xx), fail (If failed to cache from source)

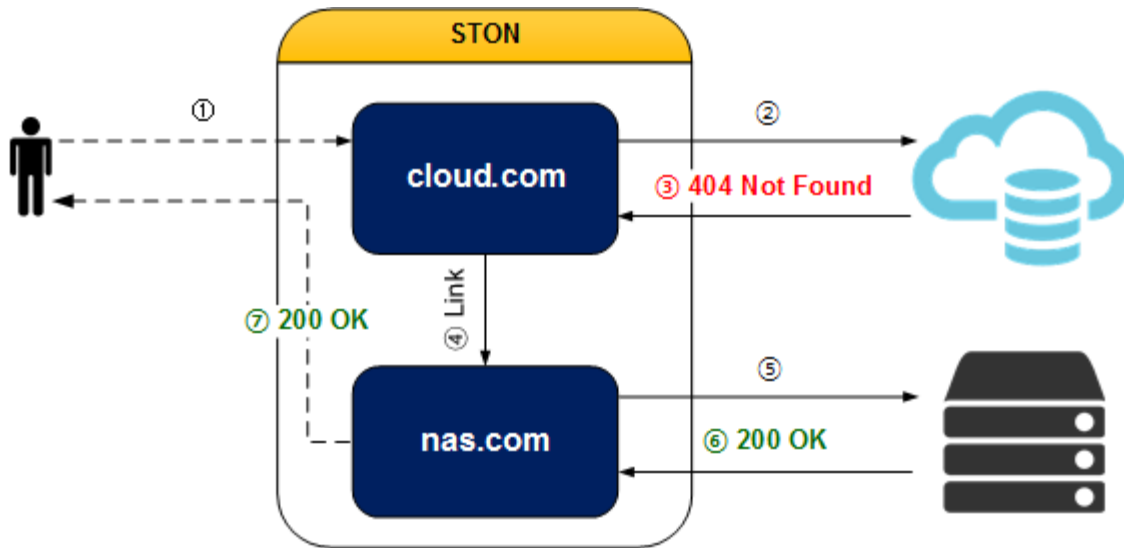


Fig. 5.3: Content missing from cloud.com is delivered by nas.com.

Even if the client request is delegated to another virtual host, the: ref: *monitoring_stats_vhost_client* and: ref: 'admin-log-access' are recorded in the virtual host accessed by the client.

Note: Please note that if the virtual hosts' configurations in the link are different from each other, it may operate in unintentional ways.

If the virtual host link is concatenated with A (simple caching) -> B (image compression) Images processed in A are not compressed, but images processed in B are compressed.

For example, if you are moving content from nas.com to cloud.com, you can only send requests to nas.com for content not on cloud.com (= 404 Not Found). In the following cases: ref: *monitoring_stats_vhost_client* and: ref: 'admin-log-access' are recorded on cloud.com, even if the request is handled by nas.com.

```
# vhosts.xml - <Vhosts>

// Content not on cloud.com (= 404 Not Found) will be served on nas.com.
<Vhost Name="cloud.com">
  <VhostLink Condition="404">nas.com</VhostLink>
</Vhost>

<Vhost Name="nas.com">
</Vhost>
```

The vhostlink field of: ref: *admin-log-access* will tell you which virtual host the client request was processed on. "-" means the request is not linked, and "nas.com" means that the request has been linked and processed at nas.com.

```
#Fields: date time s-ip cs-method cs-uri-stem ...(...)... vhostlink
2016.11.24 16:52:24 220.134.10.5 GET /web/h.gif ...(...)... -
2016.11.24 16:52:26 220.134.10.5 GET /favicon.ico ...(...)... nas.com
```

If the link has occurred multiple times, all virtual hosts linked with "+" delimiters are specified. In this case, the last virtual host is the virtual host that processed the last request.

You can link multiple virtual hosts to different conditions as follows:

```
# vhosts.xml - <Vhosts>

// When the origin server responds with 5xx or fails to cache (= fail), it delegates the request to bar.com.
<Vhost Name="foo.com">
  <VhostLink Condition="5xx,fail">bar.com</VhostLink>
</Vhost>

// Delegates the request to helloworld.com when the origin server responds with 4xx.
<Vhost Name="bar.com">
  <VhostLink Condition="4xx">helloworld.com</VhostLink>
</Vhost>

// When the origin server responds with 403, 404, or 5xx, it delegates the request to example.com.
<Vhost Name="helloworld.com">
  <VhostLink Condition="403,404,5xx">example.com</VhostLink>
</Vhost>

// Does not delegate any more.
<Vhost Name="example.com">
</Vhost>
```

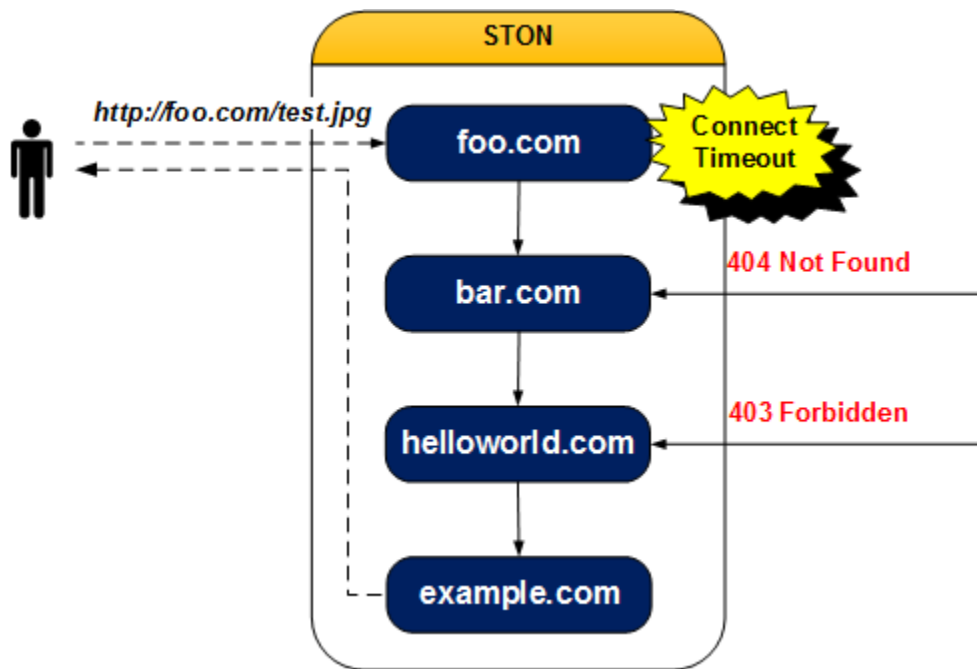


Fig. 5.4: It might look awkward but not impossible.

In the above example, the ref: *admin-log-access* of foo.com looks like this:

```
#Fields: date time s-ip cs-method cs-uri-stem ...(...)... vhostlink
2016.11.24 16:52:24 220.134.10.5 GET /test.jpg ...(...)... bar.com+helloworld.com+example.com
```

In the following cases, the link is immediately terminated.

- If the target virtual host does not exist (foo.com ->?)
- If you specified yourself as the destination virtual host (foo.com -> foo.com)
- If a recursive link occurs (foo.com -> bar.com -> foo.com)

5.2 Chapter 14. Access Control

This chapter will explain how to deny access to unwanted clients. Denying access can be done by setting up a blacklist in the Access Control List (ACL), but for the sake of convenience, a whitelist can also be configured.

Access control can be divided into server access control, which occurs during connection, and virtual host access control, which can be configured for each virtual host. Because the access time and standards are different for each level, it is important to choose what is most effective. All access control is logged.

5.2.1 Server Access Control

When the client connects to the server, the server decides whether to deny access based on the IP address. As this occurs during connection, the process is decisive and swift. This can be configured in global settings (server.xml) and has the highest priority.

```
# server.xml - <Server><Host>

<ServiceAccess Default="Allow">
  <Deny>192.168.7.9-255</Deny>
  <Deny>192.168.8.10/255.255.255.0</Deny>
</ServiceAccess>
```

- `<ServiceAccess>` Configures ACL using the IP address. Supports the four formats of IP, IP Range, Bitmask, and Subnet. The order is important, with the highest configuration taking priority. Default (default: Allow) Configures how to process requests that do not meet any of the conditions. If set to Deny, IP addresses to be allowed should be specified in `<Allow>` tags.

Denied IP addresses are logged in the *Deny Log*.

5.2.2 GeoIP

GeoIP can be used to deny access based on geographical location. Binary databases in [GeoIP databases](#) are linked to `GEOIP_MEMORY_CACHE` and `GEOIP_CHECK_CACHE` to apply changes in real time.

```
# server.xml - <Server><Host>

<ServiceAccess GeoIP="/var/ston/geoiip/">
  <Deny>AP</Deny>
  <Deny>GIN</Deny>
</ServiceAccess>
```

The directory for GeoIP databases is configured in the `GeoIP` property of `<ServiceAccess>`. The supported country codes are http://en.wikipedia.org/wiki/ISO_3166-1_alpha-2 and [ISO 3166-1 alpha-3](#).

Note: Because GeoIP has a reserved file name, you must use the local path it is stored in. Also, because service changes go into effect automatically, there is no need to reload the configuration deliberately.

If GeoIP is configured, the following will check for the files in the given directory. If it is not configured, a 404 NOT FOUND response is returned.

```
http://127.0.0.1:10040/monitoring/geoiplist
```

The results are returned in JSON format.

```
{
  "version": "2.0.0",
  "method": "geoiplist",
  "status": "OK",
  "result":
  {
    "path" : "/usr/ston/geoip/",
    "files" :
    [
      {
        "file" : "GeoIP.dat",
        "size" : 766255
      },
      {
        "file" : "GeoLiteCity.dat",
        "size" : 12826936
      }
    ]
  }
}
```

5.2.3 Virtual Host Access Control

Controls access for each virtual host. The virtual host decides whether to deny access when the client sends an HTTP request. This is because the virtual host cannot be found without an HTTP request.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<AccessControl Default="Allow" DenialCode="401">OFF</AccessControl>
```

- <AccessControl>
 - OFF (default) The ACL is not activated. All client requests are allowed.
 - ON The ACL is activated. Denied requests are responded to with the code set in the DenialCode property. If the Default (default: Allow) property is set to Allow, the ACL is used as a blacklist. If it is set to Deny, the ACL is used as a whitelist.

Denied requests are logged in the *Access Log* as TCP_DENY.

Virtual Host ACL

Determines whether client HTTP requests are allowed/denied/redirected. You can also configure different response codes for each condition. For redirected requests, the response will be **302 Moved Temporarily**. The ACL is configured in the `/svc/{virtual host name}/acl.txt` file.

```
# /svc/www.example.com/acl.txt
# Commas (,) are delimiters, and the order is {condition},{keyword = allow|deny|redirect}.
# If set to deny, the response code can be set after the keyword.
# If not specified, the DenialCode of <AccessControl> is used.
# If set to redirect, the URL to be redirected to can be set after the keyword (stated as a value of
# Multiple conditions can be combined (AND) with an & sign.

$IP[192.168.1.1], allow
$IP[192.168.2.1-255]
$IP[192.168.3.0/24], deny
```

```
$IP[192.168.4.0/255.255.255.0]
$IP[AP] & !HEADER[referer], allow
$IP[GIN], redirect, /page/illegal_access.html
$HEADER[cookie: *ILLEGAL*], deny, 404
$HEADER[via: Apache]
$HEADER[x-custom-header]
$HEADER[referer:], redirect, http://another-site.com
!HEADER[referer] & !HEADER[user-agent] & !HEADER[host], deny
$URL[/source/public.zip], allow
$URL[/source/*]
/profile.zip, deny, 500
/secure/*.dat
```

A condition can be formatted as an IP, GeoIP, Header, or URL.

- **IP** Represented as \$IP[...], in which an IP, IP Range, Bitmask, or Subnet can be input.
- **GeoIP** Represented as \$IP[...] and can only be used if GeoIP is configured.
- **Header** Represented as \$HEADER[Key : Value]. The Value can recognize specific or patterned expressions. If Value is an empty string but the colon is there (\$HEADER[Key:]), then it refers to an empty header value. If the Key is specified without the colon (\$HEADER[Key]), then the condition will refer to any request that has the corresponding header.
- **URL** Represented as \$URL[...] and can be omitted. Specific and patterned expressions can be used.

\$ means “if the condition is met, perform the action”, while ! means “if the condition is not met, perform the action”. The following negative conditions are supported.

```
# If the country is not KOR, deny.
!IP[KOR], deny

# If the referer header is missing, deny.
!HEADER[referer], deny

# If not under the /secure/ path, allow.
!URL[/secure/*], allow
```

When redirecting, the URI that the client requests might be needed. This can be obtained with the #URI keyword.

```
# If the referer header is missing, the URI is appended to example.com and redirected.
# Client requests begin with /, so there's no need to add a / to the redirect URL.
!HEADER[referer], redirect, http://example.com#URI
```

5.3 Chapter 15. Bandwidth

This chapter will explain various ways to set bandwidth limits for each virtual host. In the past, the objective was generally to prevent bandwidth from exceeding a fixed limit. However, the idea has changed into regulating bandwidth to be effective. It is now possible to analyze content in real time to optimize the use of bandwidth.

5.3.1 Virtual Host Bandwidth Limits

Limits the maximum bandwidth of a virtual host. This is a concrete method that has the highest priority.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>
```

```
<TrafficCap Session="0">0</TrafficCap>
```

- `<TrafficCap>` (default: 0 Mbps) Configures the maximum bandwidth of a virtual host in Mbps. A value of 0 will not limit the bandwidth. The “Session (default: 0 Kbps)” property configures the maximum bandwidth that can be transferred in each client session.

For example, setting `<TrafficCap>` to 50 (Mbps) will have the same effect as installing 50 Mbps NIC. The total bandwidth of all clients that connect to the virtual host will be unable to exceed 50 Mbps.

Session behaves as follows.

1. Even if Session is configured, the total client bandwidth cannot exceed `<TrafficCap>`.
2. Even if *Bandwidth Throttling* is configured, the maximum speed of each client session cannot exceed the Session value.

5.3.2 Bandwidth Throttling

Bandwidth Throttling (BT) is a function that can dynamically regulate the client transfer bandwidth for each session. Media files are generally comprised of Video (V) and Audio (A) headers, as seen below.

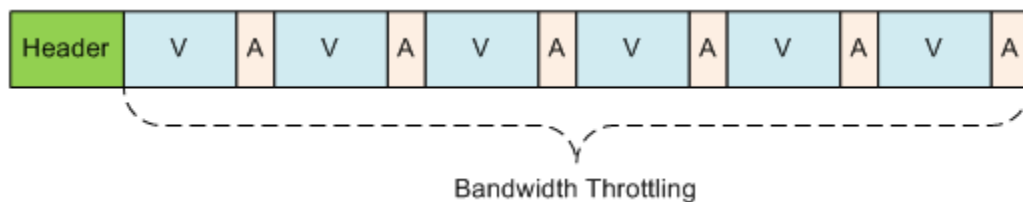


Fig. 5.5: BT is not concerned with headers.

The headers get bigger when the play-time is longer or the key frame cycle is shorter. Therefore, if the media file can be recognized, the headers are transferred without a bandwidth limit for smoother playback. As seen in the following image, it is after the headers are fully transferred that BT starts.

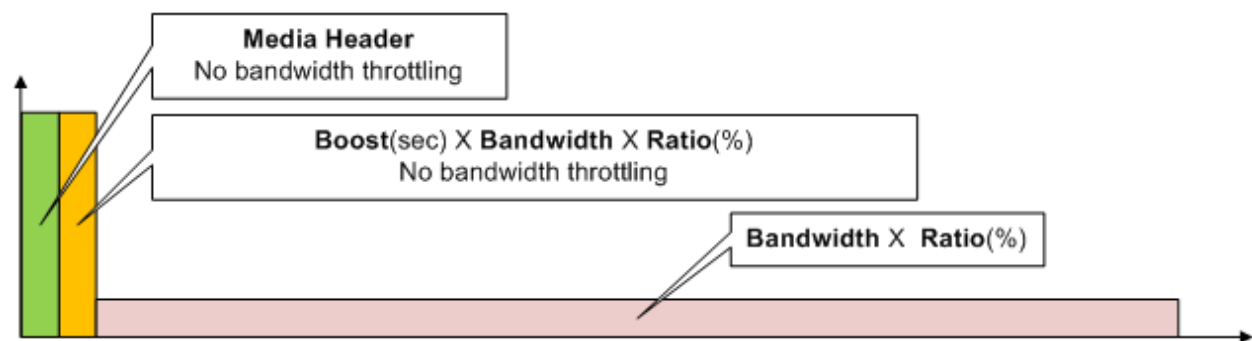


Fig. 5.6: Operation scenario

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<BandwidthThrottling>
  <Settings>
    <Bandwidth Unit="kbps">1000</Bandwidth>
```

```

    <Ratio>100</Ratio>
    <Boost>5</Boost>
  </Settings>
  <Throttling>OFF</Throttling>
</BandwidthThrottling>

```

<BandwidthThrottling> Default operation is configured in the subtags.

- <Settings> Configures default operation.
 - <Bandwidth> (default: 1000 Kbps) Configures client transfer bandwidth. The Unit property is used to configure the default unit (kbps, ``mbps, bytes, kb, mb).
 - <Ratio> (default: 100%) Configures bandwidth by applying the ratio to the <Bandwidth> setting.
 - <Boost> (default: 5 s) Transfers data to clients with unlimited speed for the set amount of time. The amount of data can be calculated with the equation <Boost> x <Bandwidth> x <Ratio>.
- <Throttling>
 - OFF (default) Does not apply BT.
 - ON Applies BT if the conditions are met.

Bandwidth Throttling Condition List

Configures the BT condition list. Conditions must be met for BT to be applied. The list is checked in order to see if any conditions are met. This transfer policy is saved in the file /svc/{virtual host name}/throttling.txt.

```

# /svc/www.example.com/throttling.txt
# Commas (,) are delimiters, and the order is {condition},{bandwidth},{ratio},{boost}.
# All fields except for {condition} can be omitted.
# Omitted fields use the default values set in <Settings>.
# All condition formats are identical to settings in acl.txt.
# The unit for {bandwidth} is the same as the Unit property of <Bandwidth> under <Settings>.

# Transfers data with unlimited speed for 3 seconds, and then limits to to 3 Mbps (3000 Kbps = 2000 Kbps)
$IP[192.168.1.1], 2000, 150, 3

# Only defines bandwidth. Transfers data with unlimited speed for 5 seconds (default), and then limits to 800 Kbps
!HEADER[referer], 800

# Only defines boost. Transfers data with unlimited speed for 10 seconds, and then limits to 1000 Kbps
HEADER[cookie], , , 10

# Does not apply BT for files with the m4a extension.
$URL[*m4a], no

```

By analyzing media files (MP4, M4A, MP3), bandwidth can be obtained from the encoding rate. The file extension must be one of .mp4, .m4a, or .mp3. In order to dynamically extract the bandwidth, you can add an x to the bandwidth value, as shown below.

```

# Finds the bandwidth for /vod/*.mp4 files. If the bandwidth cannot be found, 1000 is used instead.
$URL[/vod/*.mp4], 1000x, 120, 5

# Finds the bandwidth if the user-agent header is missing. If the bandwidth cannot be found, 500 is used.
!HEADER[user-agent], 500x

```

```
# Finds the bandwidth for /low_quality/* files. If the bandwidth cannot be found, the default value is
$URL[/low_quality/*], x, 200
```

QueryString Priority Condition

Uses a predetermined QueryString to dynamically configure <Bandwidth>, <Ratio>, and <Boost>. This configuration takes priority over BT conditions.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<BandwidthThrottling>
  <Settings>
    <Bandwidth Param="mybandwidth" Unit="mbps">2</Bandwidth>
    <Ratio Param="myratio">100</Ratio>
    <Boost Param="myboost">3</Boost>
  </Settings>
  <Throttling QueryString="ON">ON</Throttling>
</BandwidthThrottling>
```

- Param in <Bandwidth>, <Ratio>, <Boost>
 - Configures the QueryString key with a different meaning for each tag.
- QueryString in <Throttling>
 - OFF (default) Does not redefine conditions as QueryStrings.
 - ON Redefines conditions as QueryStrings.

The above configuration allows BT to be dynamically configured based on the URLs requested by clients, as seen below.

```
# Transfers data with unlimited speed for 10 seconds, and then limits to 1.3 Mbps (1 Mbps x 130%).
http://www.winesoft.co.kr/video/sample.wmv?myboost=10&mybandwidth=1&myratio=130
```

Not all parameters need to be specified.

```
http://www.winesoft.co.kr/video/sample.wmv?myratio=150
```

If some conditions are omitted as in the above example, the remaining fields (in this example, bandwidth and boost) are chosen using a condition list. If there is no condition that matches, then the default values in <Settings> are used. Even if a QueryString is specified, if the condition list is set to not apply (no), then BT will not be applied.

There is potential for confusion with *QueryString Differentiation* when using QueryStrings. If *QueryString Differentiation* is set to ON, the QueryStrings in URLs requested by clients will all be recognized except for BandwidthParam, BoostParam, and RatioParam.

```
GET /video.mp4?mybandwidth=2000&myratio=130&myboost=10
GET /video.mp4?tag=3277&myboost=10&date=20130726
```

For example, the above QueryStrings are only used to decide BT and are removed when creating a Caching-Key or sending a request to the origin server. Therefore, they will be recognized as the following.

```
GET /video.mp4
GET /video.mp4?tag=3277&date=20130726
```

5.4 Chapter 16. Media

This chapter will explain how to intelligently handle media in the service. Due to there being various client environments and service diversity, there will be many cases where content must be processed in various ways. As such, identical content will exist on the origin server in different forms. This method will not just require more processing time and storage space, it is also hard to maintain.

5.4.1 Reordering MP4/M4A Header

Normally, the header of an MP4 file cannot be completed during the encoding process, and so is attached at the end of the file after encoding is done. An extra step is necessary to move the header to the front. If the header is at the end, Pseudo-Streaming will be unavailable for media players that do not support that format. However, reordering the header can easily allow for Pseudo-Streaming support.

The header reordering only occurs during the transfer stage and will not modify the original format. It will also not take up extra storage space.

```
# server.xml - <Server><VHostDefault><Media>
# vhosts.xml - <Vhosts><Vhost><Media>

<UpfrontMP4Header>OFF</UpfrontMP4Header>
<UpfrontM4AHeader>OFF</UpfrontM4AHeader>
```

- <UpfrontMP4Header>
 - OFF (default) Nothing happens.
 - ON If the extension is .mp4 and the header is at the end of the file, the header will be moved to the front of the file before transfer.
- <UpfrontM4AHeader>
 - OFF (default) Nothing happens.
 - ON If the extension is .m4a and the header is at the end of the file, the header will be moved to the front of the file before transfer.

If content with headers that need to be moved are being requested for the first time, the parts necessary for reordering will be downloaded first. This method is not only smart, but also occurs quickly. The client will receive the file as if the header had always been in the front, without having to see any of the complicated processes occurring backstage.

Note: If the file is broken or unable to be analyzed, it will be transferred as is.

5.4.2 Trimming

Extracts desired parts of a media file using time values. Trimming only occurs during the transfer stage and will not modify the original format. It will also not take up extra storage space.

```
# server.xml - <Server><VHostDefault><Media>
# vhosts.xml - <Vhosts><Vhost><Media>

<MP4Trimming StartParam="start" EndParam="end" AllTracks="off">OFF</MP4Trimming>
<M4ATrimming StartParam="start" EndParam="end" AllTracks="off">OFF</M4ATrimming>
<MP3Trimming StartParam="start" EndParam="end">OFF</MP3Trimming>
```

- <MP4Trimming> <MP3Trimming> <M4ATrimming>

- OFF (default) Nothing happens.
- ON If the extension matches (.mp4, .mp3, .m4a), the file will be trimmed to the desired section. The section to be trimmed can be configured with the `StartParam` and `EndParam` properties.
- `AllTracks` property
 - * OFF (default) Trims only audio/video tracks (Mod-H264 format).
 - * ON Trims all tracks. Media player compatibility must be checked before using this setting.

Parameters can input via client `QueryStrings`. For example, to trim a certain section of a 10-minute video clip, you would specify the desired times (unit: seconds) in `QueryStrings`.

```
http://vod.wineosoft.co.kr/video.mp4 // 10 minutes: trim entire video
http://vod.wineosoft.co.kr/video.mp4?end=60 // 1 minute: trim from start to 1 minute (60 seconds)
http://vod.wineosoft.co.kr/video.mp4?start=120 // 8 minutes: trim from 2 minutes (120 seconds) to 10 minutes
http://vod.wineosoft.co.kr/video.mp4?start=3&end=13 // 10 minutes: trim from 3 seconds to 13 seconds
```

If the `StartParam` value is larger than the `EndParam` value, the section will be considered undefined. This function was developed to facilitate the `Skip` function for media players that support HTTP Pseudo-Streaming. As such, STON will recognize keyframes and times to extract the sections so the video can play properly, instead of trimming the file based on the offset, as when processing a `Range` request.

The file delivered to the client is in the form of a complete MP4 file with a recreated MP4 header, as shown below.

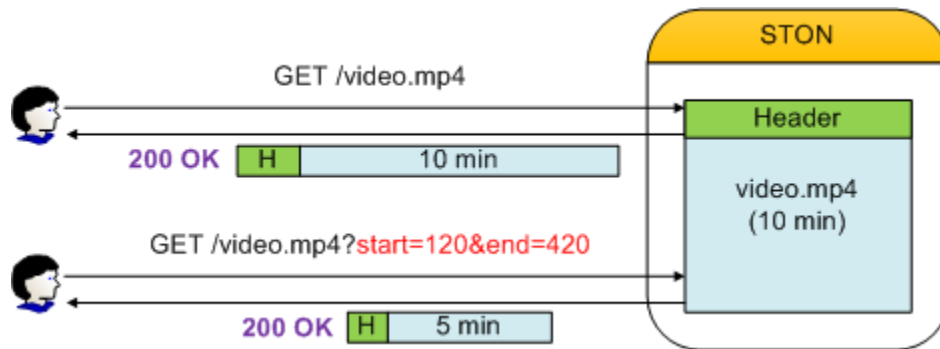


Fig. 5.7: The file is transferred in the form of a complete file.

The extracted section is recognized as a separate file, so a `200 OK` response is returned. If the `Range` header is specified as shown below, the `Range` will be calculated from the extracted file and a `206 Partial Content` response is returned.

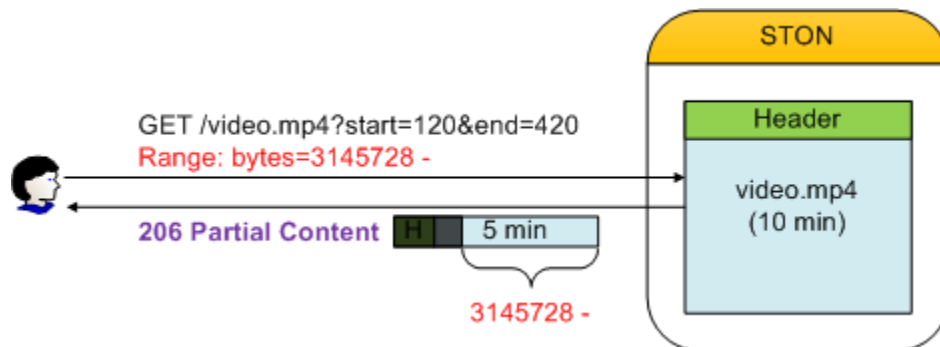


Fig. 5.8: It is processed like a normal `Range` request.

Because trimming parameters use the QueryString format, there is a possibility for confusion with *QueryString Differentiation*. If `<ApplyQueryString>` is set to ON, the QueryStrings of URLs requested by clients will be recognized, but `StartParam` and `EndParam` will be removed.

```
GET /video.mp4?start=30&end=100
GET /video.mp4?tag=3277&start=30&end=100&date=20130726
```

In the above example, if **start** and **end** are entered for `StartParam` and `EndParam`, respectively, the values will only be used to extract a section and are removed when creating a Caching-Key or sending a request to the origin server. They will then be recognized as below.

```
GET /video.mp4
GET /video.mp4?tag=3277&date=20130726
```

Expansion modules and CDN solutions can also affect how QueryStrings are used.

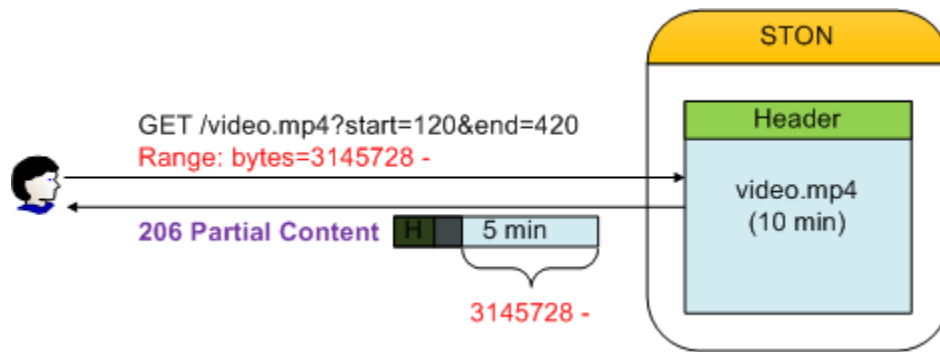


Fig. 5.9: Module/CDN references provided by the JW Player

In addition, the `ngx_http_mp4_module` in nginx and the `Mod-H264-Streaming-Testing-Version2` in lighttpd use **start** as a QueryString.

5.4.3 Multi-Trimming

Multiple sections of a video clip are stitched into a single one using time values as a basis.

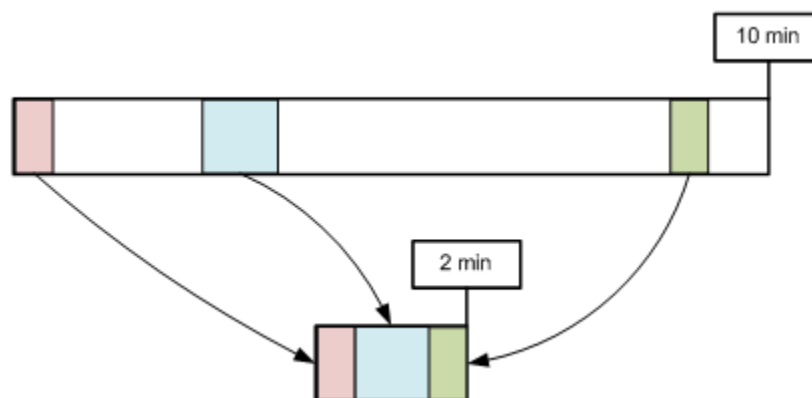


Fig. 5.10: `/video.mp4?trimming=0-30,210-270,525-555`

The function behaves in much the same way as *Trimming* except for how the sections are chosen.

```
# server.xml - <Server><VHostDefault><Media>
# vhosts.xml - <Vhosts><Vhost><Media>

<MP4Trimming MultiParam="trimming" MaxRatio="50">OFF</MP4Trimming>
<M4ATrimming MultiParam="trimming">OFF</M4ATrimming>
```

- `<MP4Trimming>` `<M4ATrimming>`
 - `MultiParam` (default: `"trimming"`) The set name is used for the `QueryString` key to determine the sections to be extracted. Each section is represented by “start - end”, and separated by a comma (,).
 - `MaxRatio` (default: `50%`) The video trimmed by Multi-Trimming can only be as large as the `MaxRatio` (max: `100%`) of the original video. Segments that exceed the `MaxRatio` are ignored.

For example, the following call will create a 3-minute video.

```
http://example.com/video.mp4?trimming=10-70,560-620,1245-1305
```

Videos can also be made with the same clip repeating, or the front and back switched.

```
http://example.com/video.mp4?trimming=17-20,17-20,17-20,17-20
http://example.com/video.mp4?trimming=1000-1200,500-623,1900-2000
http://example.com/video.mp4?trimming=600-,400-600
```

If the time values are not input, it will refer to the start and end times of the original video.

Note: *Multi-Trimming* takes priority over *Trimming*. If the *Multi-Trimming* key is seen in the `QueryString`, the *Trimming* key will be ignored.

5.4.4 MP4 HLS

MP4 files can be provided in the service with HTTP Live Streaming (HLS). The origin server no longer needs to split files for the HLS service. Regardless of the location of the MP4 file header, the file can be converted in real time to .m3u8/.ts while downloading the file.

Note:

MP4HLS is not a transcoding that converts elementary streams (video or audio). Therefore, smooth playback is only possible for MP4 files encoded for HLS. If a file is not encoded for HLS, the video may freeze or the audio may not play. The video/audio encoding format given by Apple at the time of writing (2014/2/20) is shown below.

What are the specifics of the video and audio formats supported? Although the protocol specification does not limit the video and audio formats, the current Apple implementation supports the following formats:

[Video] H.264 Baseline Level 3.0, Baseline Level 3.1, Main Level 3.1, and High Profile Level 4.1.

[Audio] HE-AAC or AAC-LC up to 48 kHz, stereo audio MP3 (MPEG-1 Audio Layer 3) 8 kHz to 48 kHz, stereo audio AC-3 (for Apple TV, in pass-through mode only)

Note: iPad, iPhone 3G, and iPod touch (2nd generation and later) support H.264 Baseline 3.1. If your app runs on older versions of iPhone or iPod touch, however, you should use H.264 Baseline 3.0 for compatibility. If your content is intended solely for iPad, Apple TV, iPhone 4 and later, and Mac OS X computers, you should use Main Level 3.1.

For existing methods, the following origin files are necessary for Pseudo-Streaming and HLS. In this case, STON will copy the origin files as is and provide them to the clients. However, as the play time gets longer, more derivative files will be created, making it harder to manage.

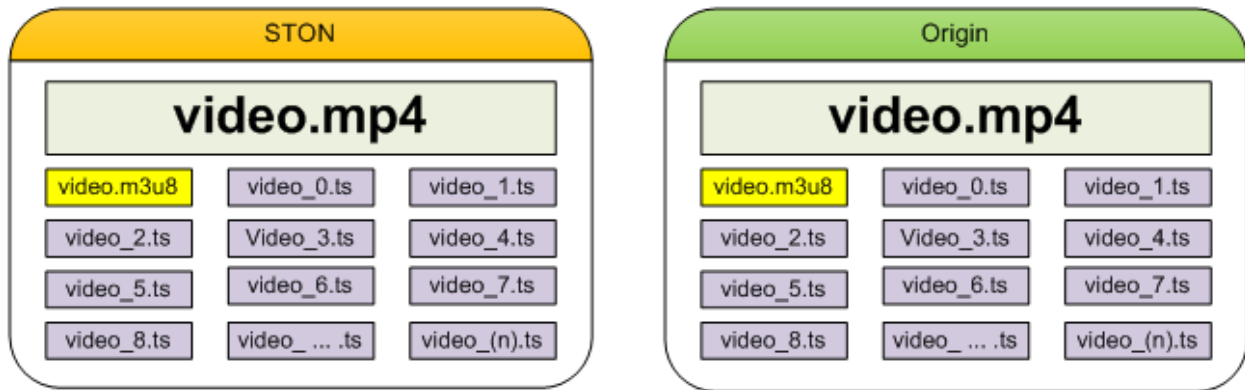


Fig. 5.11: A laborious way to provide HLS

Meanwhile, <MP4HLS> can dynamically create the necessary files from the original for the HLS service.

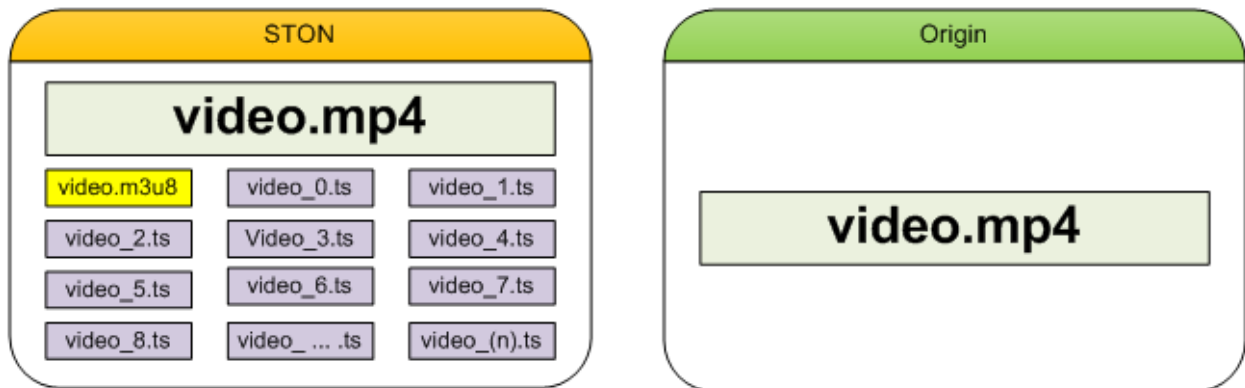


Fig. 5.12: A smarter way to provide HLS

All .m3u8/.ts files will be derived from the original file and will not consume any extra storage space. The file will only temporarily be created in memory when in use, and automatically discarded when not in use.

```
# server.xml - <Server><VHostDefault><Media>
# vhosts.xml - <Vhosts><Vhost><Media>

<MP4HLS Status="Inactive" Keyword="mp4hls">
  <Index Ver="3" Alternates="off">index.m3u8</Index>
  <Sequence>0</Sequence>
  <Duration>10</Duration>
  <AlternatesName>playlist.m3u8</AlternatesName>
</MP4HLS>
```

- <MP4HLS>
 - Status (default: Inactive) Active only when set to Active.
 - Keyword (default: mp4hls) HLS service keyword.

- `<Index>` (default: `index.m3u8`) HLS index file name (.m3u8).
 - `Ver` (default: `3`) Index file version. If the version is 3, the header will be specified as `#EXT-X-VERSION:3` and the time value of `#EXTINF` will be given to the third decimal. If the version is 1, there is no `#EXT-X-VERSION` header, and the time value of `#EXTINF` is rounded up to a whole number.
 - `Alternates` (default: `OFF`) Turns Stream Alternates on or off.

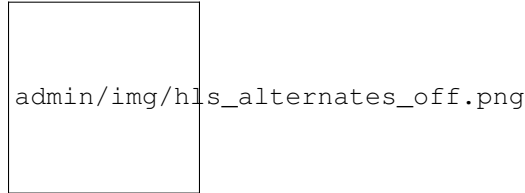


Fig. 5.13: OFF. The TS list is provided by `<Index>`.

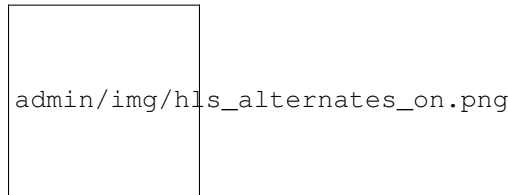


Fig. 5.14: ON. The TS list is provided by `<AlternatesName>`.

- `<Sequence>` (default: `0`) The starting number for .ts files. The number will start with the given value and increment by 1 with every file.
- `<Duration>` (default: `10 s`) Splits the MP4 file for HLS with the given time (in seconds). The basis for splitting are the keyframes in video/audio. Because the keyframes can be uneven, equal splitting is not always possible. If you attempt to split using 10 seconds but either 9 or 12 seconds is possible due to the keyframes, the closer value (9 seconds) will be chosen.
- `<AlternatesName>` (default: `playlist.m3u8`) Stream Alternates file name.

```
http://www.example.com/video.mp4/mp4hls/playlist.m3u8
```

For example, if the service address is as follows, Pseudo-Streaming can be performed with this address.

```
http://www.example.com/video.mp4
```

The virtual host will use the configured Keyword in `<MP4HLS>` to perform the HLS service. If the following URL is called, an `index.m3u8` file will be created from `/video.mp4`.

```
http://www.example.com/video.mp4/mp4hls/index.m3u8
```

If the `Alternates` property is set to ON, the `<Index>` file will provide the `<AlternatesName>` file in the service.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=200000,RESOLUTION=720x480
/video.mp4/mp4hls/playlist.m3u8
```

The Bandwidth and Resolution in `#EXT-X-STREAM-INF` are provided dynamically by analyzing the video.

Note: Though Stream Alternates is provided, index.m3u8 in the current version will always provide only one sub-index file (playlist.m3u8). This is because the cache can't tell that video_1080.mp4 and video_720.mp4 are the same video (with only the encoding being different).

The final .ts list (version 3) that is created is as follows.

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:11.637,
/video.mp4/mp4hls/0.ts
#EXTINF:10.092,
/video.mp4/mp4hls/1.ts
#EXTINF:10.112,
/video.mp4/mp4hls/2.ts

... (omitted) ...

#EXTINF:10.847,
/video.mp4/mp4hls/161.ts
#EXTINF:9.078,
/video.mp4/mp4hls/162.ts
#EXT-X-ENDLIST
```

There are three policies for splitting.

- **If the keyframe interval is smaller than <Duration>** If the keyframe is 3 seconds and <Duration> is 20 seconds, the largest multiple of the 3 (the keyframe) that does not exceed 20 (the duration) will be used to split the video (in this case, 18).
- **If the keyframe interval is close to <Duration>** If the keyframe is 9 seconds and <Duration> is 10 seconds, the largest multiple of the 9 (the keyframe) that does not exceed 10 (the duration) will be used to split the video (in this case, 9).
- **If the keyframe interval is larger than <Duration>** The keyframe interval is used to split the video.

Let's understand how STON will behave according to the following client request.

```
GET /video.mp4/mp4hls/99.ts HTTP/1.1
Range: bytes=0-512000
Host: www.winesoft.com
```

1. STON Initial loading. (Nothing has been cached yet.)
2. Client HTTP Range request. (Requests the first 500 KB of the 100th file.)
3. STON Creates a caching object of the /video.mp4 file.
4. STON Downloads from the origin server the portion necessary to analyze the /video.mp4 file.
5. STON Downloads from the origin server the portion necessary to provide the 100th file (99.ts).
6. STON Creates the 100th file (99.ts) and proceeds with the Range service.
7. STON Discards the 99.ts file after completing the service.

Note: If MP4Trimming is set to ON, MP4 files that have been trimmed can also be converted to HLS. (HLS video files cannot be trimmed. Note that HLS is not MP4 but MPEG2TS.) The most natural way of converting a trimmed video to HLS is the following.

```
/video.mp4?start=0&end=60/mp4hls/index.m3u8
```

Though this will not cause any problems, it is HTTP policy to put the QueryString at the end. To adhere to this policy, the following forms will produce the same effect.

```
/video.mp4/mp4hls/index.m3u8?start=0&end=60
/video.mp4?start=0/mp4hls/index.m3u8?end=60
```

5.4.5 MP3 HLS

MP3 files can also be provided in the service with HLS.

```
# server.xml - <Server><VHostDefault><Media>
# vhosts.xml - <Vhosts><Vhost><Media>

<MP3HLS Status="Inactive" Keyword="mp3hls">
  <Index Ver="3" Alternates="off">index.m3u8</Index>
  <Sequence>0</Sequence>
  <Duration>10</Duration>
  <AlternatesName>playlist.m3u8</AlternatesName>
</MP3HLS>
```

All settings and behaviors are identical to *MP4 HLS*.

Note: If *MP4 HLS* and *MP3 HLS* are configured to use the same Keyword, then *MP3 HLS* will not operate.

5.4.6 DIMS

The Dynamic Image Management System (DIMS) is a function that can process images into various other forms. DIMS is an expansion based on `mod_dims`. A total of seven forms (optimize, crop, thumbnail, resize, reformat, quality, composite) are available and can also be combined.

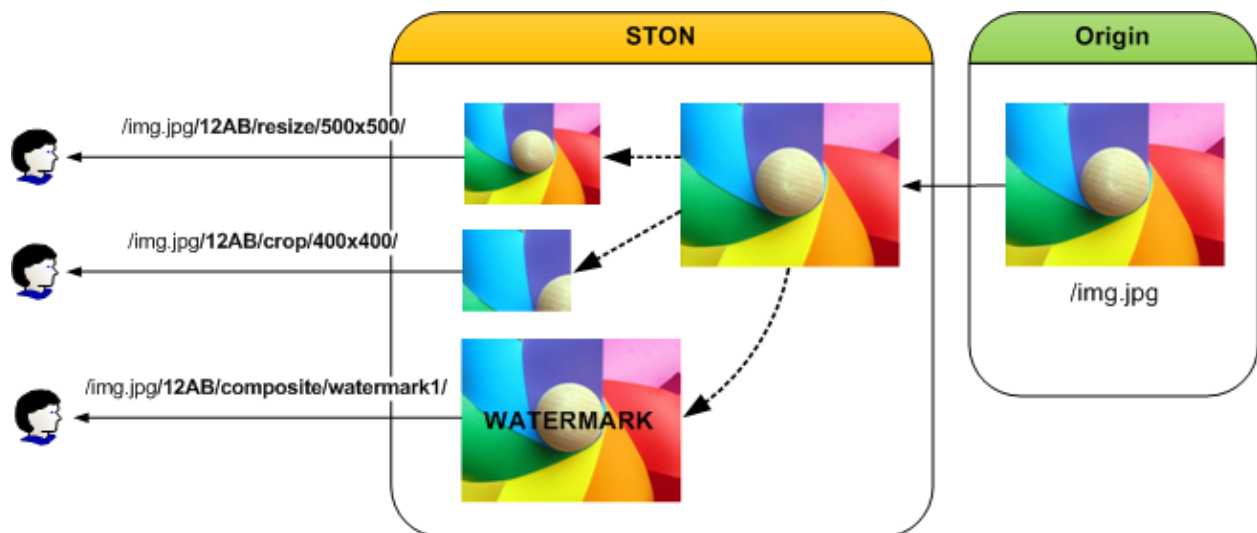


Fig. 5.15: Various ways of image processing

The image is generated dynamically and can be called by placing the keyword and the processing option at the end of the URL. The processed image will be cached and will not be reprocessed as long as the original image is not changed.

For example, if the original file is /img.jpg, the following formats can be used to process the image. (“12AB” is the configured keyword.)

```
http://image.example.com/img.jpg // Original image
http://image.example.com/img.jpg/12AB/optimize
http://image.example.com/img.jpg/12AB/resize/500x500/
http://image.example.com/img.jpg/12AB/crop/400x400/
http://image.example.com/img.jpg/12AB/composite/watermark1/
```

If <Dims> is not specifically configured, none of the following will be activated.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<Dims Status="Active" Keyword="dims" MaxSourceSize="10" OnFailure="message" />
```

- <Dims>
 - Status DIMS activation (Active or Inactive).
 - Keyword The keyword to differentiate between the original and DIMS.
 - MaxSourceSize (default: 10 MB) Maximum original image size (in MB).
 - OnFailure Behavior when image conversion fails.
 - * message (default) Responds with a 500 Internal Error. A more specific error message will be included.
 - The original file was not successfully downloaded.
 - The original file size is too large.
 - The original file loading failed.
 - Image converting failed or invalid DIMS command.
 - * redirect Performs a 302 Redirect to the original image URL.

Optimization

Optimization reduces the size of the image file while maintaining its quality. Only JPEG, JPEG-2000, and Lossless-JPEG file formats are supported. Images that have already been optimized by another tool will most likely be unable to be optimized further.

```
http://image.example.com/img.jpg/dims/optimize
```

Optimization does not require any extra options besides the keyword. As such, it is suggested to place it at the very end when combining with other options.

```
http://image.example.com/img.jpg/dims/resize/100x100/optimize
```

All the other DIMS functions use a lot of system resources, but optimization is the most resource-intensive process. The following is the result of a performance test with various images with a HitRatio of 0%.

- OS CentOS 6.2 (Linux version 2.6.32-220.el6.x86_64 (mockbuild@c6b18n3.bsys.dev.centos.org) (gcc version 4.4.6 20110731 (Red Hat 4.4.6-3) (GCC)) #1 SMP Tue Dec 6 19:48:22 GMT 2011)
- CPU Intel(R) Xeon(R) CPU E3-1230 v3 @ 3.30GHz (8 processors)

- RAM 16GB
- HDD SMC2108 SAS 275GB X 3EA

Size	Conversion	Response time (ms)	Client traffic (Mbps)	Origin traffic (Mbps)	Traffic reduction (%)
16KB	720	19.32	46.32	92.62	49.99
32KB	680	20.68	86.42	165.08	47.65
64KB	285	50.16	80.67	150.96	46.56
128KB	274	57.80	164.35	276.52	40.56
256KB	210	80.74	99.42	432.35	77.00
512KB	113	156.18	160.54	436.04	63.18
1MB	20	981.07	90.62	179.88	49.62

With about a 50% reduction in traffic, the function is very effective. As stated before, optimization is a very resource-intensive process. As can be seen in the above table, the size of the file is the most important variable.

Because of this, applying optimization without careful consideration can cause huge problems. A situation with a good *Request hit ratio* is recommended; otherwise, you must make sure that there are enough CPU resources in accordance with the scale of the service.

Crop

With the upper left corner as the origin, the image is cropped to the desired dimensions. The dimensions are formatted as **widthxheight{+-}x{+-}y{ %}**. The following is an example of cropping a section of width=100, height=200 starting from x=20, y=30.

```
http://image.example.com/img.jpg/dims/crop/100x200+20+30/
```

Generating Thumbnails

This function generates thumbnails. The size and options are formatted as **widthxheight{ % } { @ } { ! } { < } { > }**. In general, the width and height of the original image are used as the maximum values for the function. Whether the image is being enlarged or shrunk, the aspect ratio will always be preserved. To resize an image to a specific size, an exclamation point (!) can be added to the end of the size. For example, **640X480!** means that a thumbnail of exactly size 640x480 will be generated. If either the width or the height is omitted, the omitted value will be automatically calculated based on the aspect ratio.

For example, **/thumbnail/100/** will determine the height of the thumbnail using the width, while **/thumbnail/x200/** will determine the width using the height. The thumbnail size can also be expressed as a percentage (%) of the original image. To enlarge the image, a value larger than 100 (e.g. 125%) is used, while a value lower than 100 will shrink the image. It is important to keep in mind that the % character is encoded as %25 in the URL Encoding policy.

For example, 50% is encoded as 50%25. The following example generates a thumbnail with width=78, height=110.

```
http://image.example.com/img.jpg/dims/thumbnail/78x110/
```

Resizing

This function changes the size of the image. The new size is formatted as **width x height**. Even if the image is changed, the aspect ratio will be preserved. The following example resizes the original image to a size of width=200, height=200.

```
http://image.example.com/img.jpg/dims/resize/200x200/
```

Converting Format

This function converts the format of the image. The supported formats are “png”, “jpg”, and “gif”. The following example converts from a JPG to a PNG.

```
http://image.example.com/img.jpg/dims/format/png/
```

Adjusting Image Quality

This function adjusts the image quality. This function is effective because it can reduce the volume of the transferring image. The allowed range is from 0 to 100. The following example adjusts the quality of an image to 25%.

```
http://image.example.com/img.jpg/dims/quality/25/
```

Image Composition

This function composites two or more images. Unlike the previous functions, the settings for this function must be configured in advanced. This function is helpful when placing a watermark on an image.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<Dims Status="Active" Keyword="dims" port="8500">
  <Composite Name="water1" File="/img/small.jpg" />
  <Composite Name="water2" File="/img/medium.jpg" Gravity="se" Geometry="+0+0" Dissolve="50" />
  <Composite Name="water_ratio" File="/img/wmark_s.png" Gravity="s" Geometry="+0+15%" Dissolve="100" />
</Dims>
```

- <Composite>

Configures the settings for image composition. The function is configured by its properties, and will take no extra values.

- Name Designates the name of the image to be called. The “/” character cannot be used. This option will be located after “/composite/” in the URL.
- File Designates the path of the image to be composited.
- Gravity (default: c) Starting from the upper left, there are nine points (nw, n, ne, w, c, e, sw, s, se) where composition can be applied.

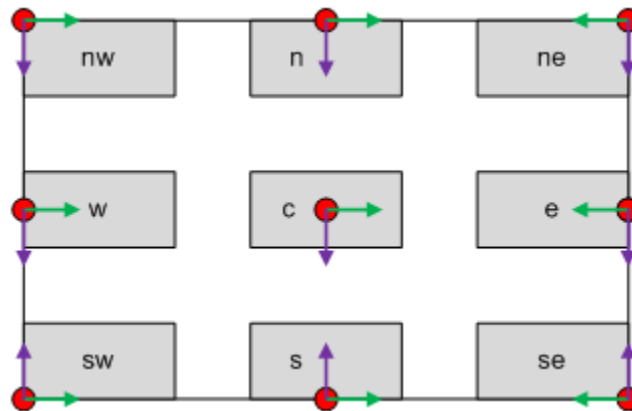


Fig. 5.16: Gravity reference points

- **Geometry (default: +0+0)** Uses Gravity as the origin to determine the location where composition is applied. The format is {+}x{+}y. The red circles are the points specified by the Gravity property and +0+0, and can be moved towards the center of the image as the values of +x+y increase. The green arrow indicates +x, and the purple arrow indicates +y. If -x-y is used, it will refer to a point outside of the image dimensions, and the composited image will not show up. This property may seem rather convoluted, but because it can automatically calculate the size of images, it is effective in creating consistent results. Furthermore, the % option can be used as in +x%+y% to use ratios as values.
- **Dissolve (default: 50)** Opacity of the image to be composited (0~100).

If <Composite> is configured, the Name property can be called to composite the images.

```
http://image.example.com/img.jpg/dims/composite/water1/
```

Original Image Conditions

Options will be dynamically applied in different ways based on the conditions of the original image. For example, if you want to lower the quality to 50% for images that are smaller than 1024x768 and resize images to 1024x768 for images that are larger, you can configure <ByOriginal> as follows.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<Dims Status="Active" Keyword="dims" port="8500">
  <ByOriginal Name="size1">
    <Condition Width="1024" Height="768">/quality/50/</Condition>
    <Condition>/resize/1024x768/</Condition>
  </ByOriginal>
</Dims>
```

- <ByOriginal> Called with the Name property. Various conditions can be configured below with <Condition>.
- <Condition> Changes will be applied to images that satisfy the conditions.
 - Width If the width is smaller than the set value, the change is applied.
 - Height If the height is smaller than the set value, the change is applied.

If no conditions are set, changes will be applied regardless of the image size.

<Condition> tags are applied in the order they are configured. Therefore, the condition for smaller images should be set first. This can be called with the following.

```
http://image.example.com/img.jpg/dims/byoriginal/size1/
```

For another example, different <Composite> conditions can be made for different image sizes. In this case, the following example uses the different Name's of ``<Composite>.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<Dims Status="Active" Keyword="dims" port="8500">
  <Composite Name="water1" File="/img/small.jpg" />
  <Composite Name="water2" File="/img/medium.jpg" Gravity="se" Geometry="+0+0" Dissolve="50" />
  <Composite Name="water3" File="/img/big.jpg" Gravity="se" Geometry="+10+10" Dissolve="50" />
  <ByOriginal Name="size_water">
    <Condition Width="400">/composite/water1/</Condition>
    <Condition Width="800">/composite/water2/</Condition>
```

```
<Condition>/composite/water3/</Condition>
</ByOriginal>
</Dims>
```

When called with the following, a different composite will be made depending on the image size.

```
http://image.example.com/img.jpg/dims/byoriginal/size_water/
```

Animated GIF

All DIMS conversions apply identically to animated GIFs. The order of processing is as follows.

1. The frames of the GIF are loaded as multiple images.
2. Each image is converted.
3. The converted frames are put back together in a single GIF.

The more frames there are, the more processing there needs to be done, which can lower service quality. In this case, you can configure the process to convert only the first frame, lowering the cost of processing.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<Dims FirstFrameOnly="OFF" />
```

- `FirstFrameOnly` (default: OFF) When set to ON, only the first frame of the animated GIF is converted.

If the following URL is called, the `FirstFrameOnly` can be specifically designated as well.

```
http://image.example.com/img.jpg/dims/firstframeonly/on/resize/200x200/
http://image.example.com/img.jpg/dims/firstframeonly/off/resize/200x200/
```

If called with the URL above, it will take precedence over the configuration.

Other

The above default functions can be combined to process images in more complex ways. For example, you can generate a thumbnail (78x110), convert the format from JPG to PNG, and change the quality to 50% all in one step.

```
http://image.example.com/img.jpg/dims/thumbnail/78x110/format/png/quality/50/
```

DIMS can process images using URL calls. As such, it is important to take note of other options that can affect URLs and cause unwanted results.

- If *QueryString Differentiation* is set to OFF, QueryStrings found after keywords are ignored.

```
http://image.example.com/img.jpg?session=5234&type=37/dims/resize/200x200/
```

If it is set to ``ON``, it will be understood as is, but if it is set to ``OFF``, the URL will be

```
http://image.example.com/img.jpg/dims/resize/200x200/
```

- If *Case Sensitivity* is set to OFF, all URLs will be converted to lowercase and then processed. Therefore, if DIMS keywords contain uppercase letters, they will not be recognized when called. It is recommended to always use only lowercase letters for keywords.

5.5 Chapter 17. File System

This chapter will explain how to utilize STON as if it were a local disk. STON is based on **FUSE** and is mounted on the Linux VFS (Virtual File System). All files in the mounted directory will be cached the moment they are accessed, but other processes will not notice. You can consider this system as a **ReadOnly disk with a Caching function**.

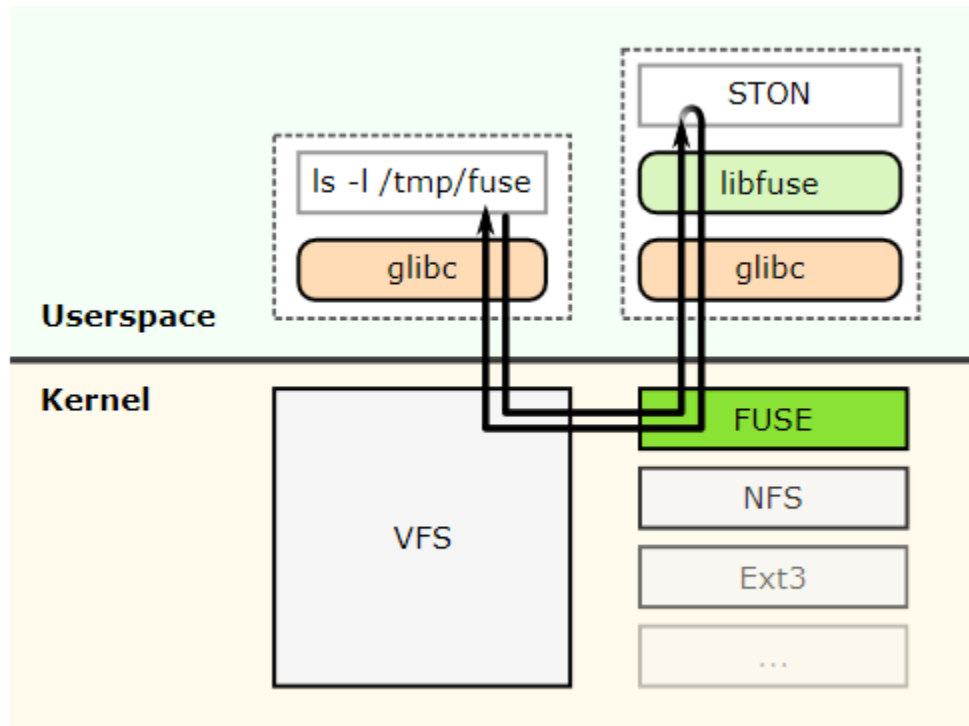


Fig. 5.17: Fuse structure.

If the Linux Kernel delivers structural File I/O function calls directly to STON, no other elements (e.g. physical File I/O, socket transmission) can interfere with the process. This architecture makes extremely high performance possible. Using STON's memory caching, you can expect performance that's better than physical disk access.

5.5.1 Mount

Mount is configured in global settings (server.xml).

```
# server.xml - <Server><Cache>
<FileSystem Mount="/cachefs" DotDir="OFF" Separator="^">OFF</FileSystem>
```

- `<FileSystem>`
 - OFF (default) Does nothing.
 - ON STON will be mounted onto the path given by the Mount property.

This was developed in such a way that the existing HTTP structure is preserved, but a file system that can access the cache module is added. As such, regardless of where the access comes from, caching occurs only once and is given service by either HTTP or File I/O. The file system is a new bridge added that allows access to the cache module.

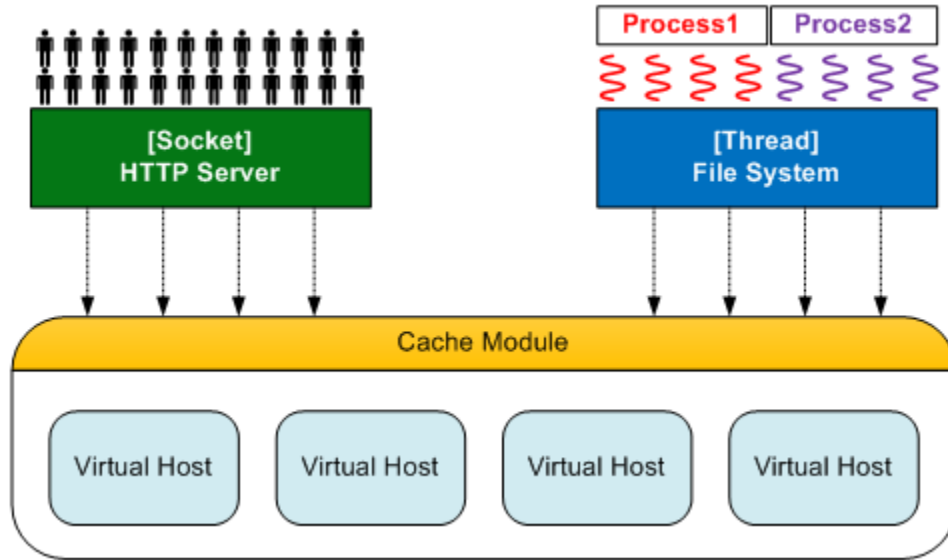


Fig. 5.18: HTTP and File I/O share a cache module.

The content on the origin server can be accessed not only by HTTP but also by File I/O. Using this, you can increase the availability of solutions that are based on local files.

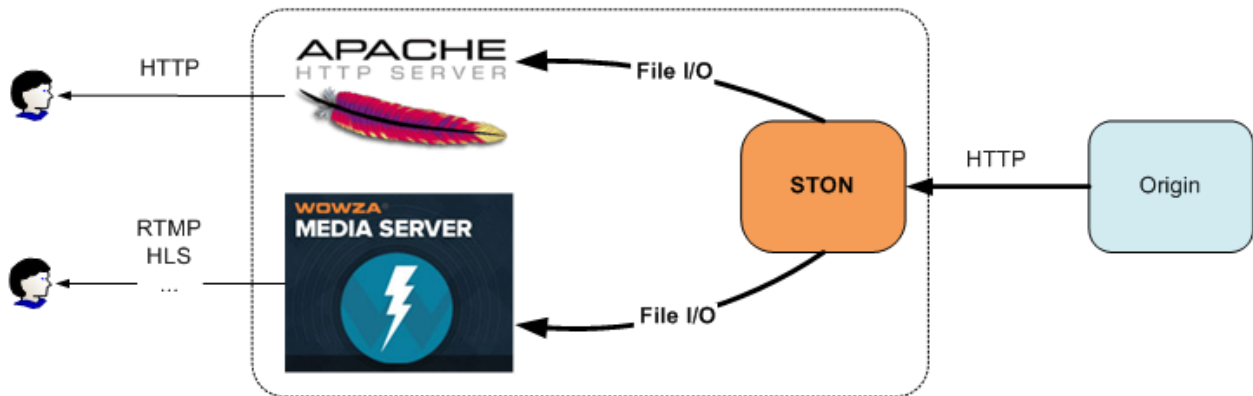


Fig. 5.19: Any server is OK.

The current list of functions that support the STON File System is as follows.

FUSE	C	LINUX
open	fopen	open
release	fclose	close
read	fseek, fread	seek, read
getattr	fstat	stat
unlink	remove	unlink

File I/O goes through several internal steps. It is important to understand what goes on at each step to obtain the best performance.

5.5.2 Searching for Virtual Hosts

The first step is searching for the virtual host to be accessed. In an HTTP header, the Host header is specified as below, making it easy to find the virtual host.

```
GET /ston.jpg HTTP/1.1
host: example.com
```

This can be done in the file system using its first directory. For example, if STON is mounted on the /cachefs directory, local files can be accessed with the following path.

```
/cachefs/example.com/ston.jpg
```

Discovering a Virtual Host will work in the same way. If the <Alias> of example.com is set to *.example.com, then the following paths will access the same file.

```
/cachefs/example.com/ston.jpg
/cachefs/img.example.com/ston.jpg
/cachefs/example.example.com/ston.jpg
```

For example, in order to link example.com to the Apache server, you must set the DocumentRoot to /cachefs/example.com/.

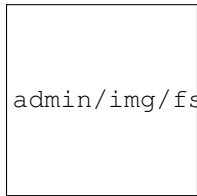
5.5.3 File/Directory

The file system can be configured for each virtual host. Alternatively, a default virtual host can be configured to give all virtual hosts the same settings.

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<FileSystem Status="Active" DotDir="OFF">
  <FileTime>origin</FileTime>
  <FileStatus>200</FileStatus>
  <DirStatus>301, 302, 400, 401, 403</DirStatus>
  <Unlink>Purge</Unlink>
</FileSystem>
```

- <FileTime> (default: Origin) Returns the Last-Modified time from the origin server when set to Origin, or the local cached time when set to Local. If the origin server does not return a Last-Modified time when set to Origin, then the file time will be returned as the Unix epoch as seen below.



admin/img/fs_filetime.png

- <FileSystem> The file system cannot be accessed if Status is Inactive. It must be set to Active.
- <FileStatus> (default: 200) Configures the origin server HTTP response code that will be recognized as a file. Generally, 200 is used, but there are no specific restrictions.
- <DirStatus> (default: 301, 302, 400, 401, 403) Configures the origin server HTTP response code that will be recognized as a directory. The default values are usually 301, 302, 400, 401, or 403.

- <Unlink> (default: Purge) Configures the behavior to be used for a file deletion request, choosing from Purge, Expire, or HardPurge.

Each origin server can interpret HTTP response codes in different ways. As such, it is important to configure how each HTTP response code should be interpreted.

In most cases, if a file exists on the origin server the response will be **200 OK**. If a directory is accessed, the response will be **403 Forbidden** or a redirect to another page with **302 Found**. Multiple response codes can be set with commas (,) to identify the Body of corresponding HTTP response codes as files or directories. Response codes that are not configured will be considered non-existing, and File I/O will fail.

5.5.4 File Properties

In general, the first step of File I/O is to obtain the properties of the file. It is obvious to obtain the file information before opening the file. The process of the Kernel providing the file properties as seen by STON is portrayed in the figure below. (/cache/fs is the mounted directory and is omitted by the Kernel.)

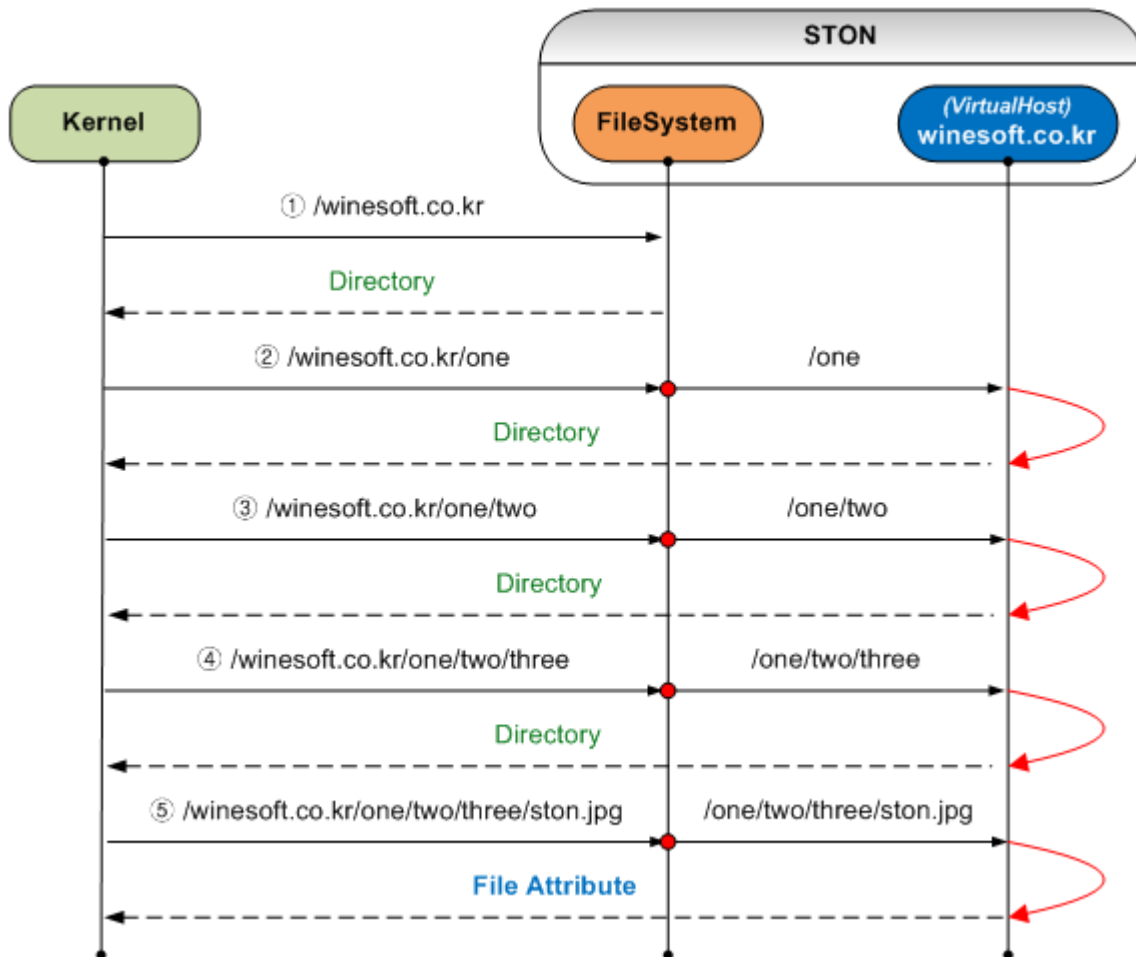


Fig. 5.20: The process of obtaining the file properties.

In Linux, files are not distinguished from directories, so obtaining the file properties can be more complicated than it seems. As can be seen from the above figure, as the number of subfolders increases, more unnecessary virtual host searches and file accesses will occur, lowering performance. In particular, requests for inaccessible directories like

/one or /one/two will be made, causing load on the origin server. Of course, if the file is cached, the origin server will not be accessed during the TTL, but it is clear that this is not an elegant solution.

A heuristic solution for this structural load is to add a `DotDir` property. `DotDir` is a function that will recognize paths without a dot (.) as a directory. The above figure is the result of `DotDir` having been set to OFF. If `DotDir` is set to ON, the following will occur.

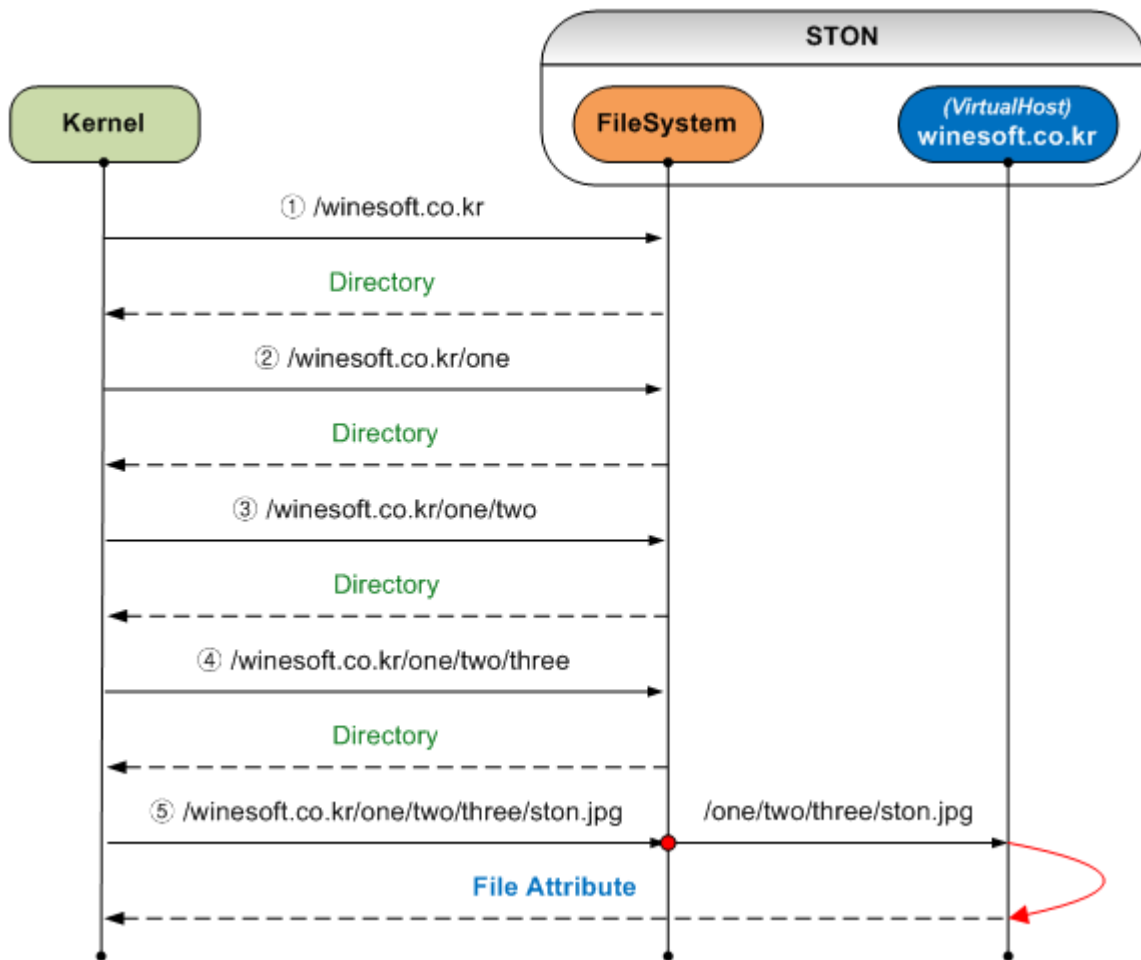


Fig. 5.21: Enabling (ON) the global `DotDir`.

There is no change in the process or number of Kernel calls. However, if the requested paths do not contain a dot (.), it will not go all the way to the virtual host and instead return immediately as a directory, allowing the virtual host and files to be accessed only when necessary. This function is based on the observation that most programmers do not assign extensions to directories. It is important that you check how directories are set up before using this function.

`DotDir` is a global property of `<FileSystem>`. In other words, if none of the virtual hosts use dots (.) for directories, it will be very effective to set `DotDir` to ON. Of course, even if `DotDir` is set to OFF, you can still configure it on each virtual host separately. Doing so can lower performance slightly, as shown below.

Virtual host searches will still occur, but files will only be accessed if there is a dot (.). As the system's performance is affected by how many times it is called, it is highly recommended to understand this function thoroughly.

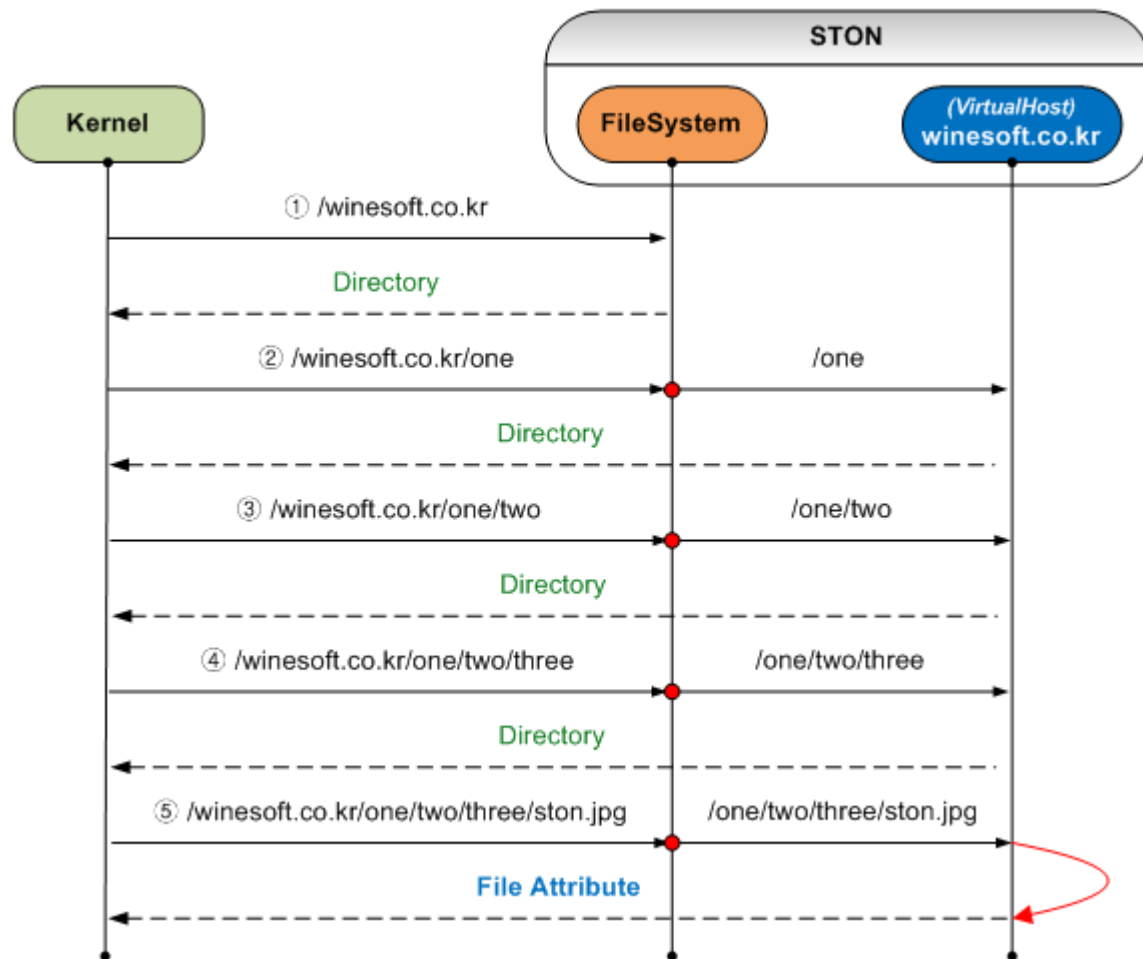


Fig. 5.22: Enabling (ON) the virtual host DotDir.

5.5.5 Reading Files

Though the process to obtain file properties is complicated, reading files is much simpler. First, the file is opened. All files will of course be read-only, so accessing a file with write permissions will fail. When a file is accessed for the first time, the file will be cached from the origin server just like the HTTP service. While downloading the requested file, the File I/O service is run concurrently so that the process is not delayed.

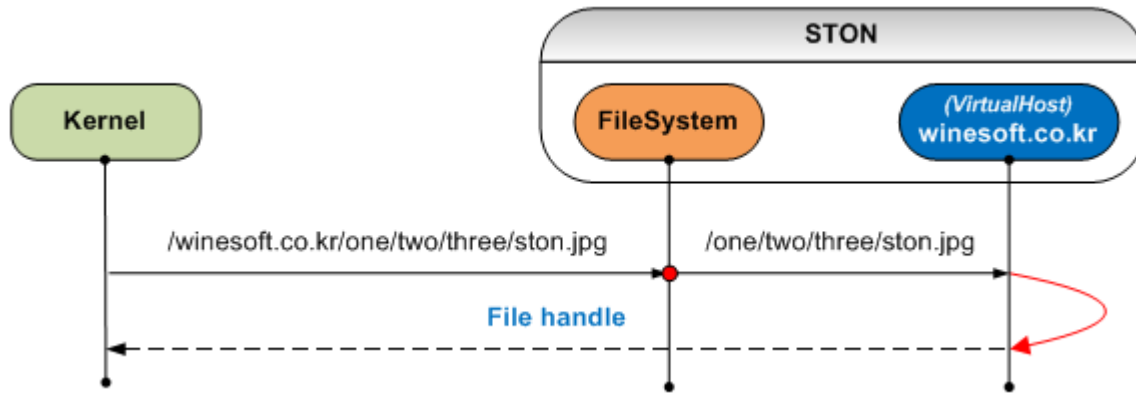


Fig. 5.23: Opening a file.

Once a file is opened, the behavior will be identical to the HTTP service. HTTP is more advantaged in file transfer because sequential file access occurs from an initially determined range. On the other hand, File I/O can generate a large number of read accesses on the scale of 1 KB regardless of file size. STON has implemented [Readahead](#) in the cache module in order to maximize performance, especially File I/O performance.

If the function to close a file (e.g. `fclose`) is called or a process is terminated, the file handle is turned in by the Kernel, which is the same as an HTTP transaction being closed.

5.5.6 Deleting Files

Cached files are managed by STON, but the process can send a request to delete a file. STON offers several [Purge](#) methods to respond to these requests.

For example, if `<Unlink>` is set to `expire`, the corresponding file will be expired upon a file deletion request. If the Kernel tries to access the file again, the file must be checked for modification on the origin server because the file is expired. If the file was not modified, it can then be provided again.

5.5.7 File Expansion

HTTP can dynamically process a file using an URL as seen below.

```
# Trims a 0-60 second section of /video.mp4 via HTTP.
http://www.example.com/video.mp4?start=0&end=60
```

This QueryString format can be used in the same way for both HTTP and the file system.

```
# Accesses the local file made from trimming a 0-60 second section of /video.mp4.
/cache/fs/www.example.com/video.mp4?start=0&end=60
```

However, putting the processing options at the end of a URL as seen in MP4HLS and DLS can cause problems in File I/O.

```
/cachefs/image.winesoft.com/img.jpg/12AB/resize/500x500/
/cachefs/www.winesoft.com/video.mp4/mp4hls/index.m3u8
```

As explained in “File Properties”, Linux asks for the properties of each directory in a path. STON is unable to tell whether additional directories are added to the end of a path, so unprocessed files will end up in the service.

To resolve this issue, STON uses the `Separator` (default: `^`) property to differentiate.

```
/cachefs/image.winesoft.com/img.jpg^12AB^resize^500x500^
/cachefs/www.winesoft.com/video.mp4^mp4hls^index.m3u8
```

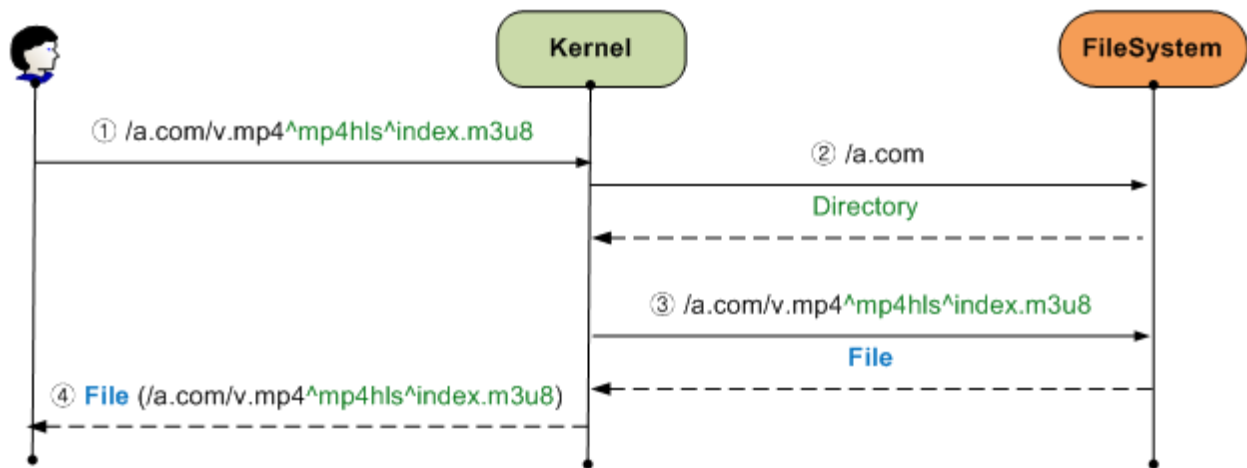


Fig. 5.24: MP4HLS access.

Within STON, the `Separator`s are switched to slashes (`/`) in order to use the HTTP call standard. Using this can eliminate unnecessary File I/O access, as shown below.

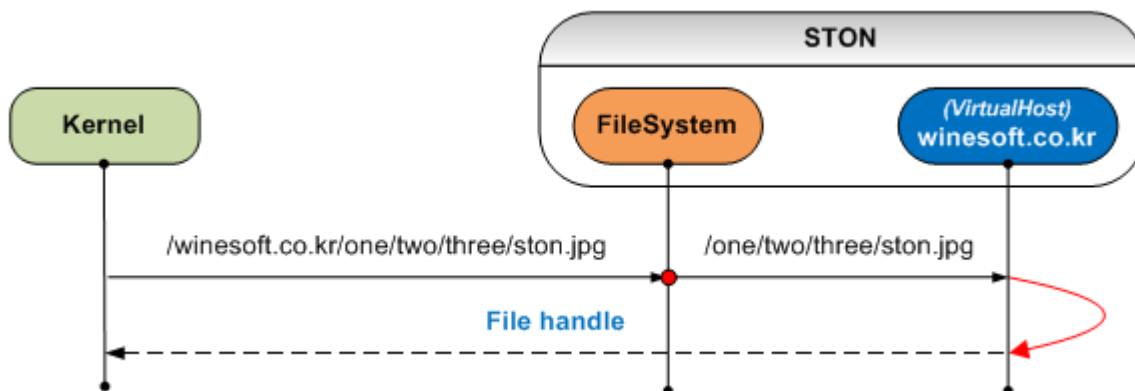


Fig. 5.25: Extremely optimized access.

5.5.8 Wowza Integration

Wowza can be integrated using the file system. All you need to do is configure the path that STON is mounted on as the file path for Wowza.

1. **[STON - Global settings] Turn on the file system configuration** Set `<FileSystem>` to ON in the global settings (server.xml). (In this example, the mount path will be set to “/cachefs”).

```
# server.xml - <Server><Cache>

<FileSystem Mount="/cachefs" DotDir="OFF" Separator="^">ON</FileSystem>
```

Alternatively, in WM, go to Global Settings -> File System and set the file system to “On”.

Global Settings – File System

File System :
 (Mount Path : , Directory Delimiter :)

✖ **STON must be restarted if File System configuration is changed.**

Fig. 5.26: STON must be restarted after configuration for mounting to be successful.

2. **[STON - Virtual host] Configure file system access permissions and response codes** File system access for virtual hosts should be set to Active. The recognition of files/directories based on origin server response codes should also be set. The following uses the virtual host default settings (server.xml) as an example, but this can also be configured individually for each virtual host (vhosts.xml).

```
# server.xml - <Server><VHostDefault><Options>
# vhosts.xml - <Vhosts><Vhost><Options>

<FileSystem Status="Active" DotDir="OFF">
  <FileStatus>200</FileStatus>
  <DirStatus>301, 302, 400, 401, 403</DirStatus>
</FileSystem>
```

Alternatively, in WM, go to Virtual Host -> Configuration (File System) and set the virtual host to “accessible”.

3. **[Wowza] Storage path configuration** In the Wowza installed path, the `/Conf/Application.xml` file should be edited to refer to the path that STON is mounted on, as shown below.

```
<Streams>
  <StreamType>default</StreamType>
  <StorageDir>/cachefs/example.com</StorageDir>
  <KeyDir>${com.wowza.wms.context.VHostConfigHome}/keys</KeyDir>
</Streams>
```

4. **[Wowza] VOD path configuration** In the Wowza installed path, the `/Conf/vod/Application.xml` file should be edited to refer to the path that STON is mounted on, as shown below.

```
<Streams>
  <StreamType>default</StreamType>
  <StorageDir>/cachefs/example.com</StorageDir>
  <KeyDir>${com.wowza.wms.context.VHostConfigHome}/keys</KeyDir>
</Streams>
```

5. **Player test** Using the Wowza test player, videos not saved in local storage (that STON must cache) can be played with RTMP.

example.com
Virtual Host – Configuration (File System)

Origin Server
Origin Server Mgmt.
Client
Log
Content Caching
Content Control
Bypass
Statistics
Media
DIMS
Bandwidth Control
File System

Virtual host accessible from FileSystem
✖ Changes are effective immediately without restart.

Either File or Directory is selected, depending on origin HTTP response code. File I/O fails for unassigned HTTP response code.

File	Directory
100 (Continue) Add 200 (OK) ✖	100 (Continue) Add 301 (Moved Permanently) ✖ 302 (Moved Temporarily) ✖ 400 (Bad Request) ✖ 401 (Authorization Required) ✖ 403 (Forbidden) ✖

Fig. 5.27: Response codes can be configured here.

5.6 Chapter 18. Optimization and More

This chapter will discuss optimization and other miscellaneous advanced topics. Optimization is a method used to obtain high performance, which is biggest merit that we are pursuing. In an enterprise environment, if hardware is high-performance, it can also mean that it uses as much resources as possible.

Among those resources is memory, the resource that is most important to all plans and policies. Memory indexing (finding requested URLs quickly) is especially important to understand, because indexing is what determines the quality of the service. The following table, displaying the default values based on the physical memory size, will be referred back to by the rest of the section.

Physical RAM	System Free	Contents	Caching Count	Sockets
1GB	409.60MB	188.37MB	219,469	5,000
2GB	819.20MB	446.74MB	520,494	10,000
4GB	1.60GB	963.49MB	1,122,544	20,000
8GB	3.20GB	2.05GB	2,440,422	20,000
16GB	6.40GB	4.45GB	5,303,733	20,000
32GB	12.80GB	9.25GB	11,030,356	20,000
64GB	25.60GB	18.85GB	22,483,603	20,000
128GB	51.20GB	38.05GB	45,390,095	20,000

5.6.1 Indexing

In order to explain indexing, you must first understand the idea of “hot” and “cold” content.

Content cached from the origin server is saved on the local disk. If that content must be read from the disk whenever it is accessed, performance will definitely decrease. As such, we can obtain higher performance by loading frequently requested content into memory. We will refer to content loaded into memory as hot content, and content located only on the disk as cold content.

Indexing refers to the process of locating hot and cold content and directly affects performance. The default is memory indexing.

Live Video Streaming

Flash RTMP Player

WIN 12,0,0,77 (Flash-AS3)

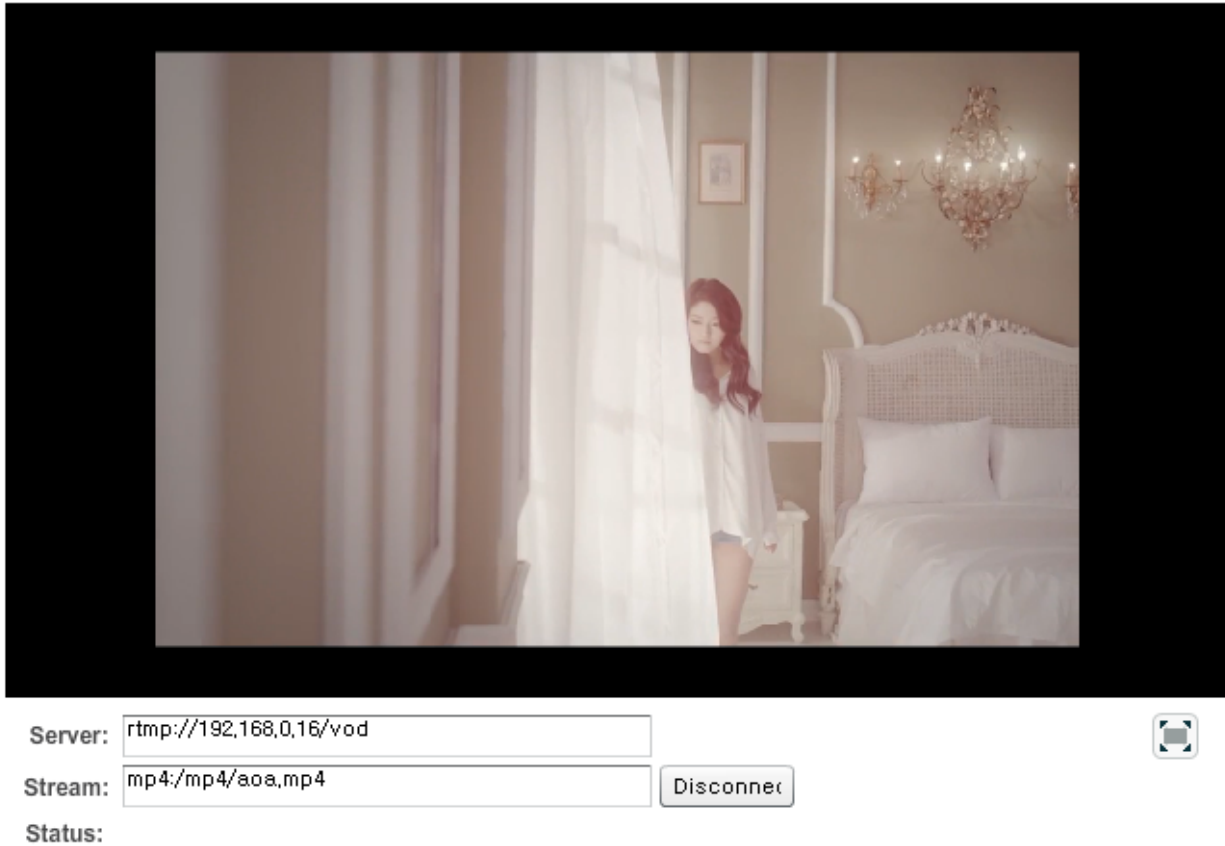
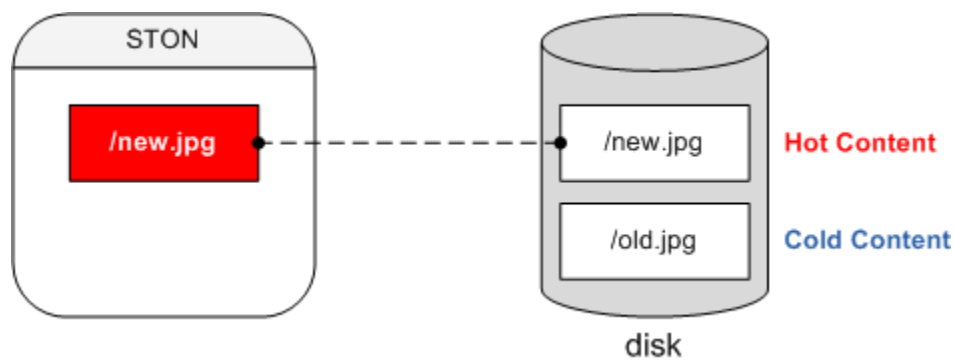


Fig. 5.28: The test needs a good video clip to play.



```
# server.xml - <Server><Cache>

<Indexing>Memory</Indexing>
```

Memory indexing does not keep a record of cold content. Information about all files is loaded into memory, so if a file cannot be found, it will be downloaded from the origin server. Because the search time is very short, we can obtain an increase in performance and service quality. However, this is limited by the memory storage size as well as the caching count, which is listed in the above table.

In disk indexing, if the requested file is not in hot content, it will look in cold content before going to the origin server.

```
# server.xml - <Server><Cache>

<Indexing>Disk</Indexing>
```

This method is not limited by memory and therefore is not limited by the caching count. It can guarantee speed if the content is hot, but it will be relatively slower if the content is slow, due to it having to use the disk. In other words, hot content is based on memory speed, and cold content is based on disk speed.

If using disk indexing, it is highly recommended that you also use a solid-state drive (SSD). Indexing is only performed on the disk that STON is installed on. Because STON is generally installed on the same disk as the OS, you can expect high performance just by using an SSD for the OS disk.

Note: SSD endurance is determined not by access frequency but by the amount that can be written. SSDs from Intel or Samsung can guarantee a write endurance of 600 TB. In other words, if 20 GB is written in a day, then the SSD can last for about 10 years. 99% of STON's writing operations is logging. Therefore, it is recommended to log on disks other than SSDs (such as SAS or SATA) to ensure the disk's endurance.

Warning: Indexing cannot be changed dynamically, and even if it is changed, it will not guarantee stability. Therefore, you must perform *Caching Reset* in order to safely proceed with the service.

5.6.2 Memory Structure

The cache server can have the same behaviors as a general web server, but their objectives are quite different. Even better service optimization is possible if you can thoroughly understand the structure and behaviors of STON. The purpose of optimization is as follows.

High throughput. Handling tens of thousands of sessions simultaneously without a drop in performance.

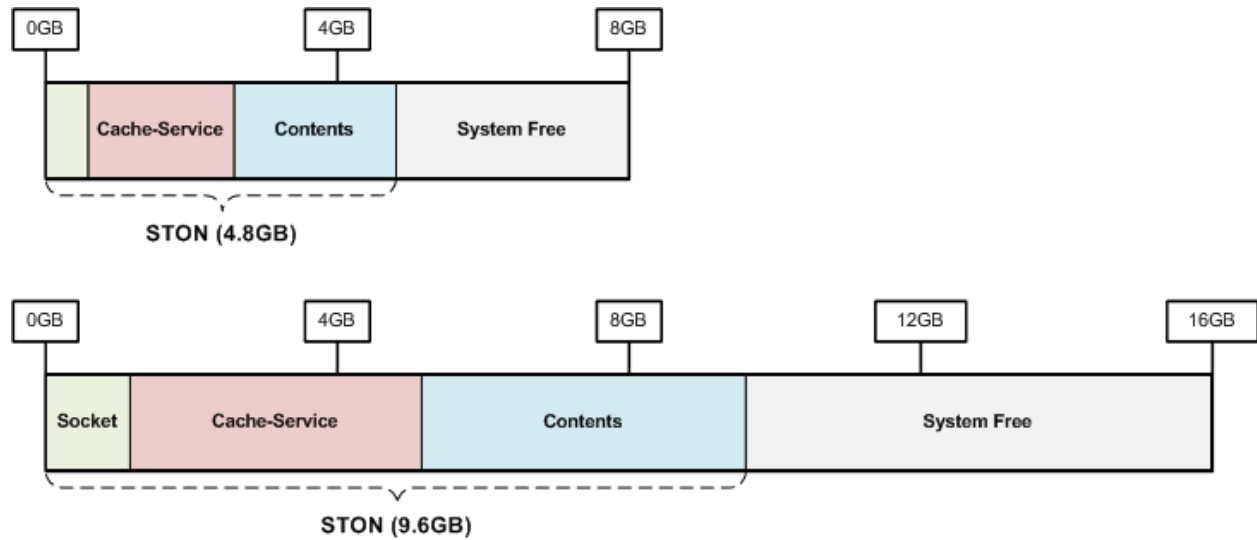
Fast responsiveness. Providing a service to clients without delay.

Reduction in origin server load. Preventing an overload on the origin server in advance.

The following figures represent the memory structure of STON with 8 GB and 16 GB memory.

Memory is divided into memory used by STON and free memory not used by STON. Like files and sockets, the memory used by STON can change based on the scale of the service.

Note: The basis of system load is disk I/O. You will have to consider how much content should be cached in order to reduce disk I/O.



5.6.3 Memory Management

Memory Structure will automatically be calculated based on the size of physical memory.

```
# server.xml - <Server><Cache>
<SystemMemoryRatio>100</SystemMemoryRatio>
```

- `<SystemMemoryRatio>` (default: 100) Configures the ratio of memory used by STON using physical memory as the basis.

For example, if memory is 8 GB and `<SystemMemoryRatio>` is set to 50, it will act as if there is 4 GB of physical memory. This can be useful if STON is run alongside other processes that take up space in memory.

It can be even more effective to adjust the ratio of content stored in memory based on the specifics of the service.

```
# server.xml - <Server><Cache>
<ContentMemoryRatio>50</ContentMemoryRatio>
```

- `<ContentMemoryRatio>` (default: 50) Configures the ratio of memory used for content to the total memory used by STON.

For example, if the file count is small but the content size is huge (like a game portal), you can increase this value to reduce file I/O. Conversely, if you have a lot of very small files, decreasing this value will be more useful.

5.6.4 System Free Memory

If the operating system (OS) is slow, no program will be able to obtain good performance. STON will set aside a portion of memory for the OS. This is to maximize the performance of the OS and is called system free memory.

Physical RAM	System Free
1GB	409.6MB
2GB	819.2MB
4GB	1.6GB
8GB	3.2GB
16GB	6.4GB
32GB	12.8GB
64GB	25.6GB
128GB	51.2GB

An experienced user will be able to adjust the free memory to what's best for their service. Reducing free memory will mean loading more content into memory.

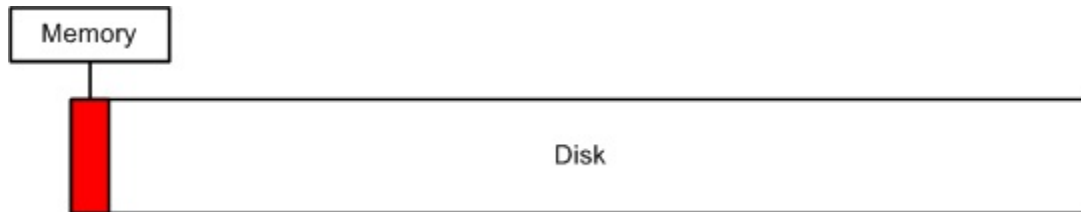
```
# server.xml - <Server><Cache>

<SystemFreeMemoryRatio>40</SystemFreeMemoryRatio>
```

- `<SystemFreeMemoryRatio>` (default: 40, max: 40) Configures the ratio of memory set aside for free memory using physical memory as a basis.

5.6.5 Caching Service Memory

This is the memory that caches content that is delivered to clients. Content loaded into memory once will continue to exist in memory, as long as there is enough space. The problem is that there will often not be enough space.



As seen in the above figure, the disk can be full of deliverable content, but the actual capacity of memory is limited. Even if you have 32 GB of physical memory, when you consider the size of game clients or HD video clips, it isn't that much. No matter how efficiently you manage memory, it will only amount to the speed of the physical disk I/O.

The most effective method is to use as much available content memory space and reduce disk I/O. The following is a table of STON's default settings for maximum content memory size based on physical memory.

Physical RAM	Contents	Caching Count
1GB	188.37MB	219,469
2GB	446.74MB	520,494
4GB	963.49MB	1,122,544
8GB	2.05GB	2,440,422
16GB	4.45GB	5,303,733
32GB	9.25GB	11,030,356
64GB	18.85GB	22,483,603
128GB	38.05GB	45,390,095

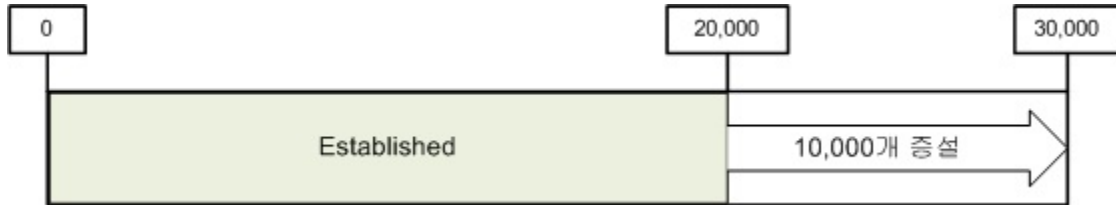
5.6.6 Socket Memory

Sockets also use memory. If you have at least 4 GB of physical memory, then STON will by default generate at least twenty thousand sockets. With one socket equaling 10 KB, ten thousand sockets will use about 97.6 MB. About 195

MB of memory will be allotted to sockets by default.

Physical RAM	Socket Count	Socket Memory
1GB	5,000	97.6MB
2GB	10,000	195MB
4GB or more	20,000	390MB

If all the sockets are used as in the following figure, more sockets will automatically created.



If, like the above figure, more sockets are installed to bring the number up to thirty thousand, that will mean about 240 MB will be allotted to sockets. There doesn't seem to be any problem with using only the number of sockets that we need. However, setting up more sockets than we need to use is just a waste of memory. For example, to guarantee 10 Mbps for each client from 10 Gbps NIC, the following equation gives us a maximum simultaneous user count of one thousand people.

$$10,000 \text{ Mbps} / 10 \text{ Mbps} = 1,000 \text{ Sessions}$$

In this case, out of the twenty thousand sockets created by STON, only nineteen thousand are used, wasting about 148 MB. This 148 MB could be used for content, increasing efficiency. By setting the smallest possible number of sockets, we can gain use memory much more efficiently.

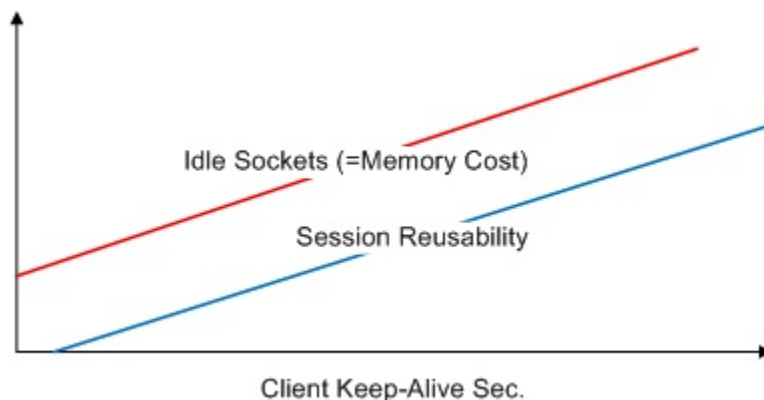
Minimum number of sockets. Refers to the number of initially allotted sockets.

Sockets that are installed later. Installs more sockets if all current sockets are established.

Another important factor is the Keep-Alive time setting for the client (see [Session Management](#)).



Not all connected sockets will be in the middle of transferring data. In browsers like IE or Chrome, sockets are maintained in an accessed state to prepare for the next HTTP transfer that will occur. Among the connected sessions in online shopping, the percentage of sessions that aren't transferring any data ranges from 50% to 80%.



If the Keep-Alive time is long, the reusability of the socket is better, but there will be more idle sockets and more memory waste. As such, it is important to configure a client Keep-Alive time that works best for the service.

5.6.7 TCP Segmentation Offload

Important: If you're using 10 Gbps NIC, it is recommended that you configure TCP Segmentation Offload (TCP) to OFF.

In TCP, packets go under segmentation; TSO configures it so that this process is done not by the CPU but by NIC. (The default setting is ON.) However, we have experienced many errors related to this in a 10 Gbps NIC service environment.

- TCP packet loss and delay
- TCP connection timeout
- Unnatural increase in the load average

In conclusion, we can assume that TSO is unable to provide the high performance we expected from it. (These problems did not occur when changing NIC to 1 Gbps.) When TSO was set to OFF, the service returned to normal. This is not a point of concern about the usage of CPU but a good benchmark for the scale of the service.

The TSO setting can be configured/checked with the following. (The K is case-sensitive.)

```
# ethtool -K ethX tso off          // TSO OFF setting
# ethtool -k ethX                  // Setting check
...
tcp segmentation offload: on
...
```

Tip: Please refer to the following links for more information.

- <http://sandilands.info/sgordon/segmentation-offloading-with-wireshark-and-ethtool>
- <http://www.linuxfoundation.org/collaborate/workgroups/networking/tso>
- <http://www.packetinside.com/2013/02/mtu-1500.html>

5.6.8 Client Request Limit

If you allow unlimited client requests, it can cause excessive load on the system. System overload is a very possible error. This can protect the system by preventing client requests over a certain number.

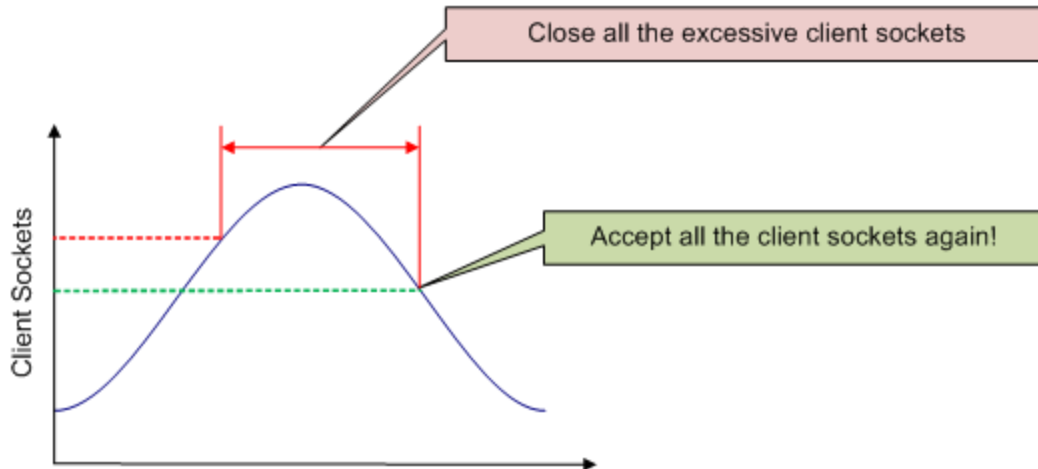
```
# server.xml - <Server><Cache>

<MaxSockets Reopen="75">80000</MaxSockets>
```

- **<MaxSockets>** (default: 80000, max: 100000) The maximum number of client sockets that will be allowed. If the socket count falls below the **Reopen** (default: 75%) ratio in **<MaxSockets>**, access will be reallocated.

Using the default settings, if the total client socket count exceeds 80,000, connections from new clients will immediately be closed. If the total client socket count falls to 60,000 (75% of 80,000), connections will be reallocated.

For example, there are thirty thousand client sessions and the origin servers have reached their limit, setting this value to thirty or forty thousand is recommended. Doing so, the available benefits are as follows.



- There is no need for a separate network setup (such as L4 session control).
- Prevents unnecessary client requests (that can't be processed due to origin load).
- Raises service credibility. There will be no need for restarting or inspection during service bursts.

5.6.9 HTTP Client Session Count

Configures the initial/additional session count to process HTTP client connections.

```
# server.xml - <Server><Cache>

<HttpClientSession>
  <Init>20000</Init>
  <TopUp>6000</TopUp>
</HttpClientSession>
```

- `<Init>` The number of sockets initially generated when STON is started.
- `<TopUp>` The number of additional sockets generated when the initial sockets are all in use.

When not specifically configured, the settings will be automatically configured based on the size of physical memory.

Physical RAM	<code><Init></code> , <code><TopUp></code>
1GB	5 thousand, 1 thousand
2GB	10 thousand, 2 thousand
4GB	20 thousand, 4 thousand
8GB or more	20 thousand, 6 thousand

If service is still possible with a smaller number of sockets in a limited environment, you can save on memory by lowering this socket count.

5.6.10 Request hit ratio

First, you must understand how client HTTP requests are processed. Caching processing results use the `TCP_*` format just like Squid, and each expression refers to how the cache server processed the request.

- `TCP_HIT` The requested resource (not expired) is cached and will respond immediately.

- **TCP_IMS_HIT** The resource requested with the If-Modified-Since (IMS) header is not expired and still cached, and will respond with 304 NOT MODIFIED. This will also apply when TTLExtensionBy4xx or TTLExtensionBy5xx is set to ON.
- **TCP_REFRESH_HIT** The requested resource is expired and will respond after checking the origin server (origin not modified, 304 NOT MODIFIED). The expiration time of the resource is extended.
- **TCP_REF_FAIL_HIT** The origin server check during the TCP_REFRESH_HIT result fails (connection failure, transfer delay) and will respond with expired content.
- **TCP_NEGATIVE_HIT** The requested resource is abnormal (origin server connection/transfer failure, 4xx response, 5xx response) and will respond with its currently cached form.
- **TCP_REDIRECT_HIT** Responds with a Redirect according to the service's allow/deny/redirect conditions.
- **TCP_MISS** The requested resource is not cached (requested for the first time) and will respond with the result of accessing the origin server.
- **TCP_REF_MISS** The requested resource is expired and will respond after an origin server check (origin change, 200 OK). The new resource is cached.
- **TCP_CLIENT_REFRESH_MISS** The request is bypassed to the origin server.
- **TCP_ERROR** The requested resource is not cached (requested for the first time). Due to an origin server error (connection failure, transfer delay, origin exclusion) the resource was not cached. Responds with a 500 Internal Error to the client.
- **TCP_DENIED** The request is denied.

The request hit ratio can be calculated using the above results, and the formula is shown below.

$$\frac{\text{TCP_HIT} + \text{TCP_IMS_HIT} + \text{TCP_REFRESH_HIT} + \text{TCP_REF_FAIL_HIT} + \text{TCP_NEGATIVE_HIT} + \text{TCP_REDIRECT_HIT}}{\text{SUM(TCP_*)}}$$

5.6.11 Byte hit ratio

The byte hit ratio is the ratio of the client outbound traffic to the origin inbound traffic. A negative number can arise if the origin inbound traffic is higher than client outbound traffic.

$$\frac{\text{Client Outbound} - \text{Origin Inbound}}{\text{Client Outbound}}$$

5.6.12 Origin Server Failure Policy

One of the goals of the development team was to have the customer be able to examine the origin server at all times. If an error occurs on an origin server, the corresponding server will be excluded and go into restoration mode. Even if the server is reactivated, normal service operation must be confirmed before it can be put back into the service.

If all origin servers somehow end up in an error status, the service continues with only the content that was cached at the time. Expired content will have their TTLs extended until the origin server is restored. Even purged content can be restored if it cannot be cached from the origin server, in order to proceed with a smooth service. The goal is to not expose the error status of the servers to the clients as much as possible. If a request for new content is made with a complete error status, the following error page and reason will be displayed.

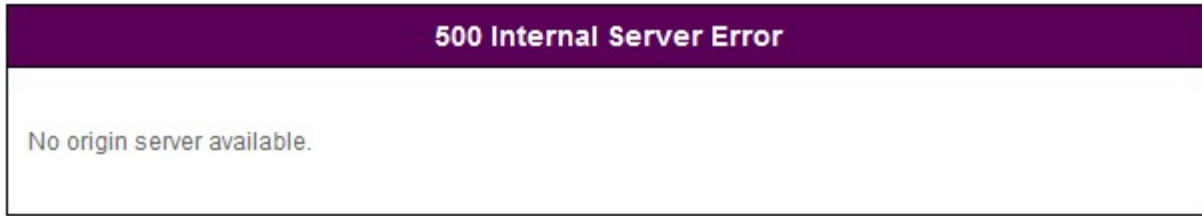


Fig. 5.29: You’d want to avoid showing this page as much as possible.

5.6.13 Time Units and Expressions

For values with a base unit of “seconds”, a string can instead be used for easier time expression. The following are the supported time expressions as well as their values converted to seconds.

Expression	Conversion
year(s)	31536000 sec (365 days)
month(s)	2592000 sec (30 days)
week(s)	604800 sec (7 days)
day(s)	86400 sec (24 hours)
hour(s)	3600 sec (60 min)
minute(s), min(s)	60 sec
second(s), sec(s), (omitted)	1 sec

The following expression, using combined units, can be used for time.

```
1year 3months 2weeks 4days 7hours 10mins 36secs
```

This can currently be used for the following values.

- Time expression of Custom TTL
- Everything but the Ratio of TTL
- ClientKeepAliveSec
- ConnectTimeout
- ReceiveTimeout
- BypassConnectTimeout
- BypassReceiveTimeout
- ReuseTimeout
- Cycle property of Recovery
- Bandwidth Throttling

5.6.14 Emergency Mode

Internally, all virtual hosts share MemoryBlocks to manage data. If new memory is necessary, old MemoryBlocks that aren’t being used can be reused as new memory. This process is called Memory-Swap. Using this structure can guarantee stability even for long periods of service.

Like in the right diagram of the above image, a situation may occur where all MemoryBlocks are in use with no reusable MemoryBlocks. In this case, Memory-Swap will be unavailable. For example, if all the clients are each downloading different parts of data at the same time, and the origin server is transferring different parts of data, then

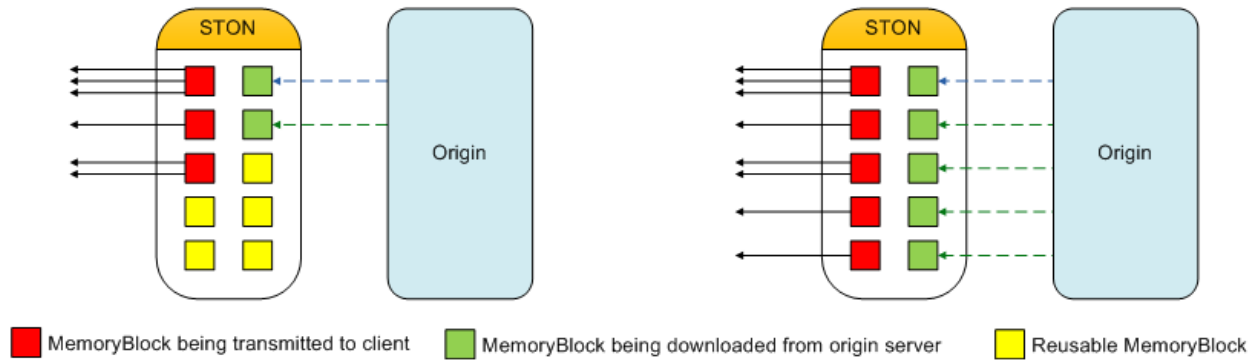


Fig. 5.30: Content is loaded onto a MemoryBlock before being delivered.

this worst-case scenario can occur. In this case, one solution is to have the system allot more memory to use. However, if the situation continues, memory usage can easily increase. An excessive use of memory can lead to a system memory swap or, at worst, the OS forcing STON to quit.

Note: Emergency mode refers to the mode where, when there is not enough memory, STON temporarily prevents new MemoryBlocks from being allotted.

STON will be put into emergency mode to prevent excessive memory usage, and the mode will be automatically lifted when enough reusable MemoryBlocks can be guaranteed.

```
# server.xml - <Server><Cache>

<EmergencyMode>OFF</EmergencyMode>
```

- <EmergencyMode>
 - OFF (default) Emergency mode is not used.
 - ON Emergency mode is used.

In emergency mode, STON behaves as follows.

- Content that is already loaded will be provided normally.
- Bypasses will work normally.
- Content not loaded will return 503 service temporarily unavailable. TCP_ERROR statuses will increase.
- Idle client sockets will be quickly taken care of.
- New content will be unable to be cached.
- Expired content will not be renewed.
- The cache.vhost.status in SNMP and the Host.State value in XML/JSON statistics will return “Emergency”.
- The info log will record the activation/inactivation of emergency mode as below.

```
2013-08-07 21:10:42 [WARNING] Emergency mode activated. (Memory overused: +100.23MB)
... (omitted) ...
2013-08-07 21:10:43 [NOTICE] Emergency mode inactivated.
```

5.6.15 Disk Hot-Swap

Swaps the disks without stopping the service. The parameters must be the same as those in `<Disk>`.

```
http://127.0.0.1:10040/command/unmount?disk=...  
http://127.0.0.1:10040/command/umount?disk=...
```

The excluded disk is immediately inactivated and all content saved on the disk is invalidated. The status of the disk excluded by the administrator is set to “Unmounted”.

To reactivate the disk, the following should be called.

```
http://127.0.0.1:10040/command/mount?disk=...
```

All content in the reactivated disk is invalidated.

5.6.16 SyncStale

After an abnormal service termination, content that has been invalidated by *Purge*, *Expire*, or *HardPurge* may be skipped over by indexing (due to performance reasons). To make up for this, the API calls will be logged and go into effect when the service is restarted.

```
# server.xml - <Server><Cache>  
  
<SyncStale>ON</SyncStale>
```

- `<SyncStale>`
 - ON (default) Synchronizes on restart.
 - OFF Ignores.

The log can be found at `./stale.log` and is initialized upon normal termination or regular indexing.

Appendix

6.1 Appendix A: Graph

All MRTG statistics are shown as PNG-format graphs. The call format is the resource and then the unit.

```
# 5 types of CPU graphs (dash, day, week, month, year)
http://127.0.0.1:10040/graph/cpu_dash.png
http://127.0.0.1:10040/graph/cpu_day.png
http://127.0.0.1:10040/graph/cpu_week.png
http://127.0.0.1:10040/graph/cpu_month.png
http://127.0.0.1:10040/graph/cpu_year.png
```

All graphs are provided in 5 different types.

Type	Size	Time unit	Period
dash	205 X 175	5 min	12 hours
day	580 X 203	5 min	2 days (48 hours)
week	580 X 203	30 min	2 weeks (14 days)
month	580 X 203	2 hours	7 weeks
year	580 X 203	1 day	18 months

A graph can have from one to three lines. The Main line is drawn in green, while the Sub line is drawn in blue. In graphs for “Week” and above, a Peak line is also displayed. The Peak line draws the highest value from smaller units in pink.

Note: If too many graphs are made at once, CPU usage will increase sharply and affect the service quality. To prevent this, please make sure to only draw one graph at a time.

6.1.1 Global Resource

The global resource graph shows the system status or resources related to STON. Below, the asterisk can be replaced with one of five types: dash, day, week, month, or year.

CPU

```
/graph/cpu_*.png
```

- Main Kernel + User

- Sub Kernel

STON CPU

/graph/stoncpu_*.png

- Main Kernel + User
- Sub Kernel

Memory

/graph/mem_*.png

- Main Total usage
- Sub STON usage

IO Wait

/graph/iowait_*.png

- Main IO Wait

Load Average

/graph/loadavg_*.png

- Main Load Average

Server Socket Event (Client -> STON)

/graph/ssockevent_*.png

- Main Accepted
- Sub Closed

Server Socket Usage (Client -> STON)

/graph/ssockusage_*.png

- Main Total server sockets
- Sub Established server sockets

Client Socket Event (STON -> Origin Server)

/graph/csockevent_*.png

- Main Connected client sockets
- Sub Closed client sockets

Client Socket Usage (STON -> Origin Server)

/graph/csockusage_*.png

- Main Total client sockets
- Sub Established client sockets

Denied IP Accesses

/graph/acldenied_*.png

- Main Denied clients

Event Queue

/graph/eq_*.png

- Main Length of event queue

Write Pending

/graph/wf2w_*.png

- Main Number of write-pending files

Successful URL Preprocessing

/graph/urlrewrite_*.png

- Main Number of preprocessed URLs

TCP Socket

/graph/tcpsocket_*.png

■ Established, ■ Timewait, ■ Orphan, ■ Alloc, ■ Mem

6.1.2 Virtual Host

The virtual host graph shows the status of all hosts or individual hosts. The vhost parameter can be used to choose a specific virtual host, but if it is omitted then it will provide the statistics of all virtual hosts.

http://127.0.0.1:10040/graph/vhost/mem_day.png?vhost=example.com

Below, the asterisk can be replaced with one of five types: dash, day, week, month, or year.

Hit Ratio

/graph/vhost/hitratio_*.png

- Main Request Hit Ratio
- Sub Byte Hit Ratio

Amount of Content

/graph/vhost/filecount_*.png

■ ~4KB, ■ ~16KB, ■ ~32KB, ■ ~64KB, ■ ~128KB, ■ ~256KB, ■ ~1MB,
■ ~10MB, ■ ~100MB, ■ ~512MB, ■ ~1GB, ■ ~4GB, ■ 4GB~, ■ Total

Content Memory

/graph/vhost/mem_*.png

- Main Amount of content data loaded into memory

Delete Pending

/graph/vhost/wf2d_*.png

- Main Number of delete-pending files

Client Bypass

/graph/vhost/client_httpreq_bypass_*.png

- Main Bypassed client HTTP requests

Denied Client Requests

/graph/vhost/client_httpreq_denied_*.png

- Main Denied client requests

Client Sessions

/graph/vhost/client_http_session_*.png

- Main Total client sessions
- Sub Client sessions in the middle of transfer

Client Traffic

```
/graph/vhost/client_traffic_*.png
```

- Main Inbound
- Sub Outbound

Client Responses

```
/graph/vhost/client_http_res_*.png
```

- Main Number of client HTTP responses
- Sub Number of client HTTP requests

Detailed Client Responses

```
/graph/vhost/client_http_res_detail_*.png
```

■ 2xx, ■ 3xx, ■ 4xx, ■ 5xx, ■ Requests

Client Transaction Completions

```
/graph/vhost/client_http_res_complete_*.png
```

- Main Number of completed client HTTP responses
- Sub Number of client HTTP requests

Client Response Time

```
/graph/vhost/client_http_res_time1_*.png
```

- Main HTTP response time for a client request

Client Completion Time

```
/graph/vhost/client_http_res_time2_*.png
```

- Main HTTP transaction completion time for a client request

Client Caching Response

```
/graph/vhost/client_http_res_hit_*.png
```

■ TCP_HIT, ■ TCP_IMS_HIT, ■ TCP_REFRESH_HIT, ■ TCP_REF_FAILHIT, ■ TCP_NEGATIVE_HIT, ■ TCP_MISS, ■ TCP_REFRESH_MISS, ■ TCP_CLIENT_REFRESH_MISS, ■ TCP_DENIED, ■ TCP_ERROR, ■ Requests

Client SSL Traffic

```
/graph/vhost/client_traffic_ssl_*.png
```

- Main Inbound
- Sub Outbound

Origin Server Sessions

```
/graph/vhost/origin_http_session_*.png
```

- Main Total origin sessions
- Sub Origin sessions in the middle of transfer

Origin Server Traffic

```
/graph/vhost/origin_traffic_*.png
```

- Main Inbound
- Sub Outbound

Origin Server Responses

```
/graph/vhost/origin_http_res_*.png
```

- Main Number of origin HTTP responses
- Sub Number of origin HTTP requests

Detailed Origin Server Responses

```
/graph/vhost/origin_http_res_detail_*.png
```

 2xx,  3xx,  4xx,  5xx,  Requests

Origin Server Transaction Completions

```
/graph/vhost/origin_http_res_complete_*.png
```

- Main Number of completed origin HTTP responses
- Sub Number of origin HTTP requests

Origin Server Response Time

```
/graph/vhost/origin_http_res_timel_*.png
```

- Main HTTP response time for requests sent to the origin server

Origin Server Completion Time

```
/graph/vhost/origin_http_res_time2_*.png
```

- Main HTTP transaction completion time for requests sent to the origin server

6.2 Appendix B: Cacti Monitoring

This appendix will explain how to set up monitoring of multiple instances of STON using Cacti's Graph Tree. The following two prerequisites are required.

- A server with Cacti installed
- SNMP activation (see *Chapter 11. SNMP*)

6.2.1 Adding Templates

Using the Host Template provided by STON, the monitoring environment can easily be set up ([download](#)).

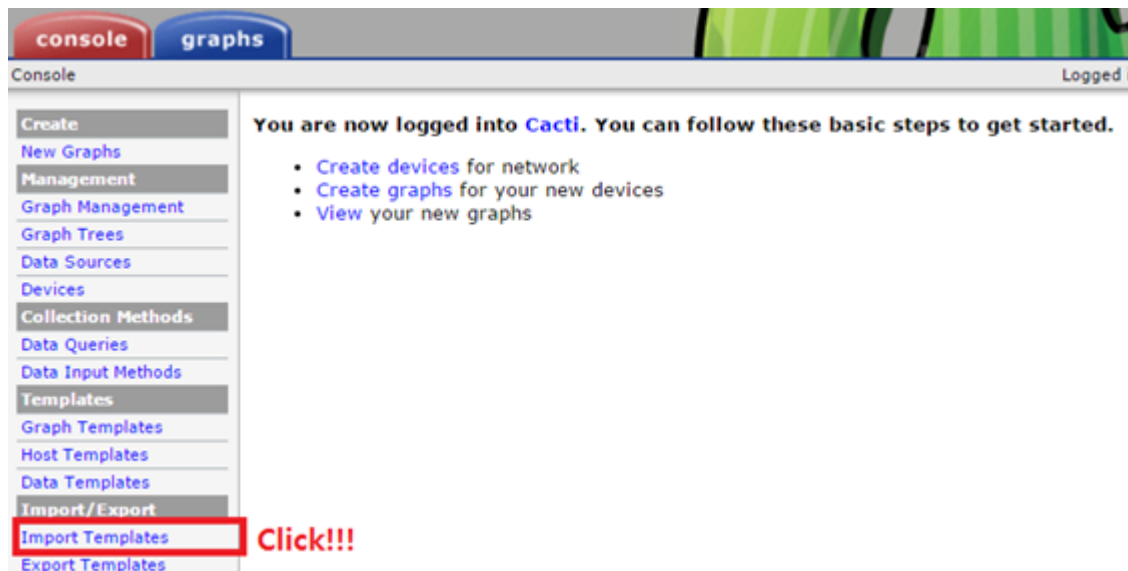


Fig. 6.1: Select [Import Templates].

6.2.2 Device Registration

Register STON as a Cacti device.

1. Input the name used for STON.
2. Input STON's IP address.
3. Select "STON".
4. Select "Public".
5. Input the default port of 161.

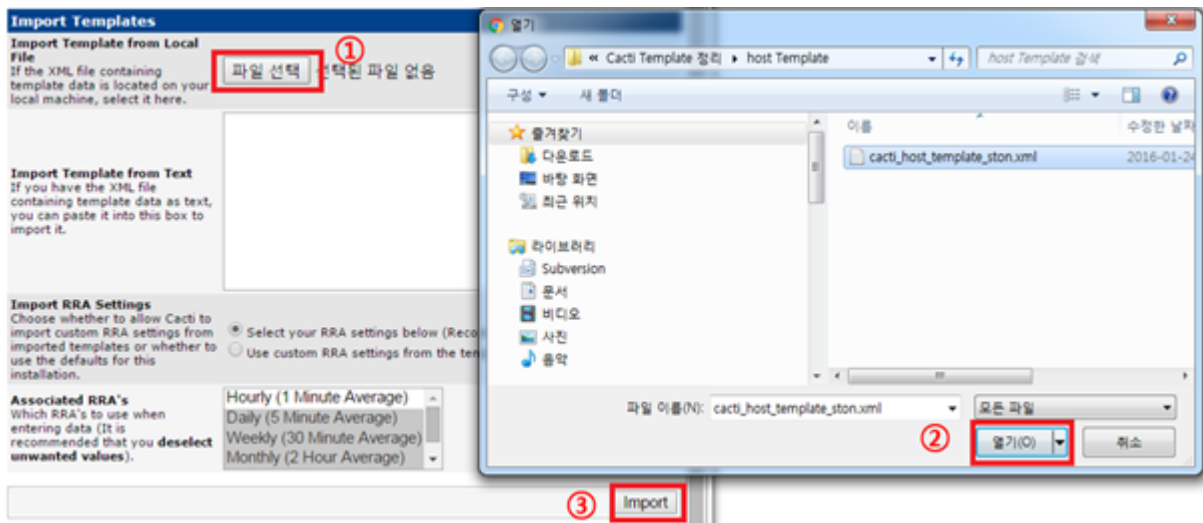


Fig. 6.2: Import cacti_host_template_ston.xml.

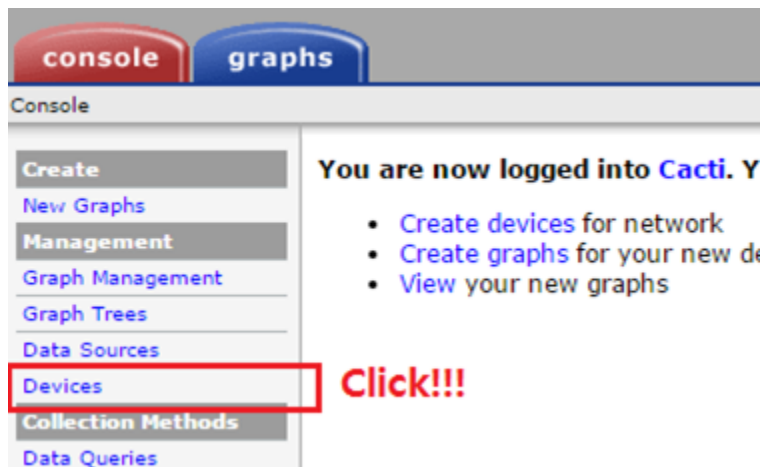


Fig. 6.3: Select [Devices].

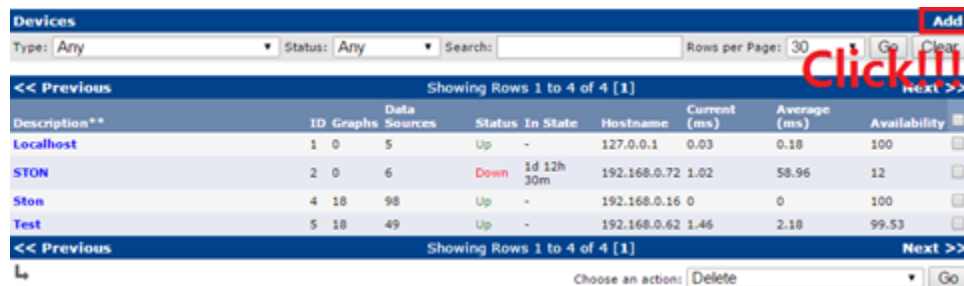


Fig. 6.4: Click the [Add] button in the [Devices] menu.

Devices [new]

General Host Options

Description
Give this host a meaningful description. ①

Hostname
Fully qualified hostname or IP address for this device. ②

Host Template
Choose the Host Template to use to define the default Graph Templates and Data Queries associated with this Host. ③

Number of Collection Threads
The number of concurrent threads to use for polling this device. This applies to the Spine poller only.

Disable Host
Check this box to disable all checks for this host. ☐ Disable Host

Availability/Reachability Options

Downed Device Detection
The method Cacti will use to determine if a host is available for polling.
NOTE: It is recommended that, at a minimum, SNMP always be selected.

Ping Timeout Value
The timeout value to use for host ICMP and UDP pinging. This host SNMP timeout value applies for SNMP pings.

Ping Retry Count
After an initial failure, the number of ping retries Cacti will attempt before failing.

SNMP Options

SNMP Version
Choose the SNMP version for this device.

SNMP Community
SNMP read community for this device. ④

SNMP Port
Enter the UDP port number to use for SNMP (default is 161). ⑤

SNMP Timeout
The maximum number of milliseconds Cacti will wait for an SNMP response (does not work with php-snmp support).

Maximum OID's Per Get Request
Specified the number of OID's that can be obtained in a single SNMP Get request.

Additional Options

Notes
Enter notes to this host.

Fig. 6.5: Fill in the device options.

Click the “Create” button to engage the device.



Fig. 6.6: Engaged successfully.

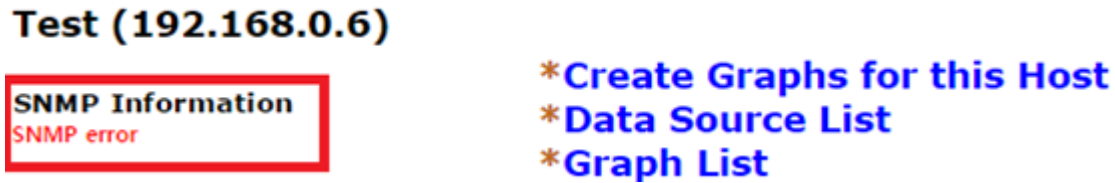


Fig. 6.7: Engagement error.

Note: If the SNMP engagement fails:

- Check STON to make sure SNMP is enabled.
 - Check to see if the SNMP port number matches STON’s SNMP port number.
-

If the device engagement succeeds, you can use 18 different types of graph provided by the STON template.

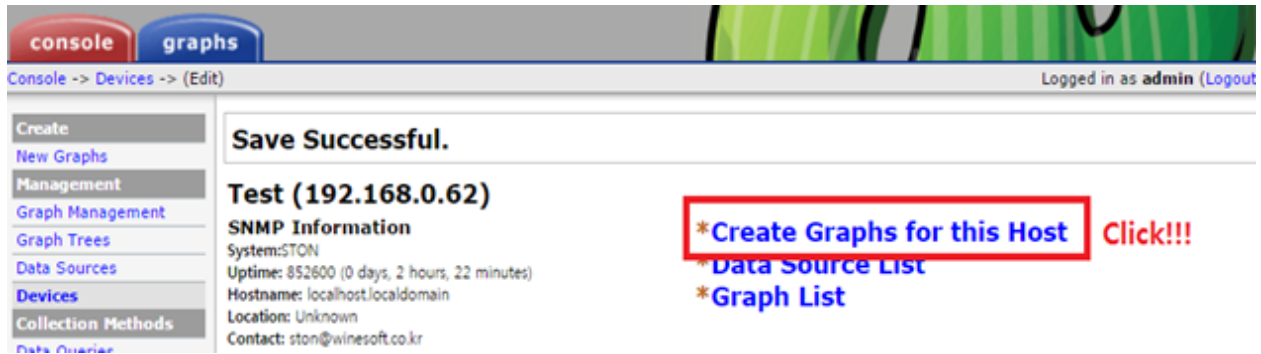


Fig. 6.8: Click “Create Graphs for this Host”.

Click the [Create] button and check the graphs that were created.

6.2.3 Graph Tree Creation

Create Graph Trees.

You can add STON to the Graph Tree.

1. Select “Host”.



Fig. 6.9: There are 18 types of graphs provided.

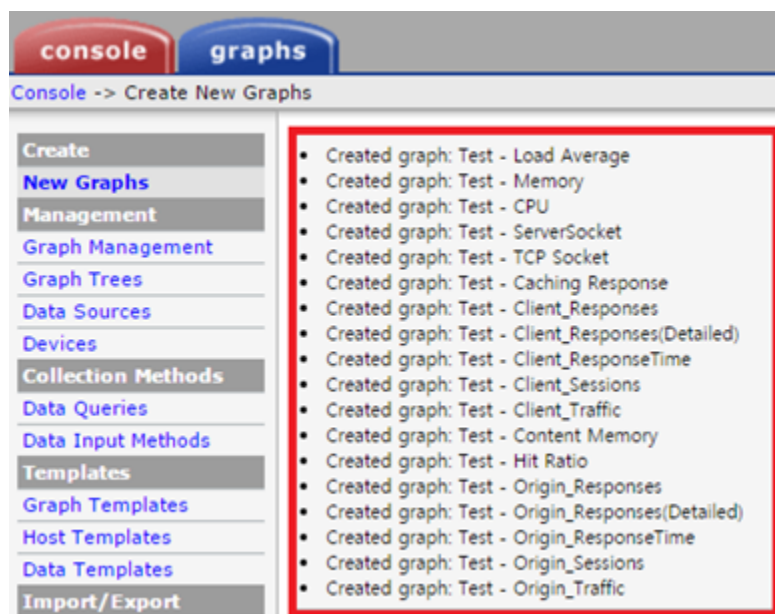


Fig. 6.10: The graphs have been created.

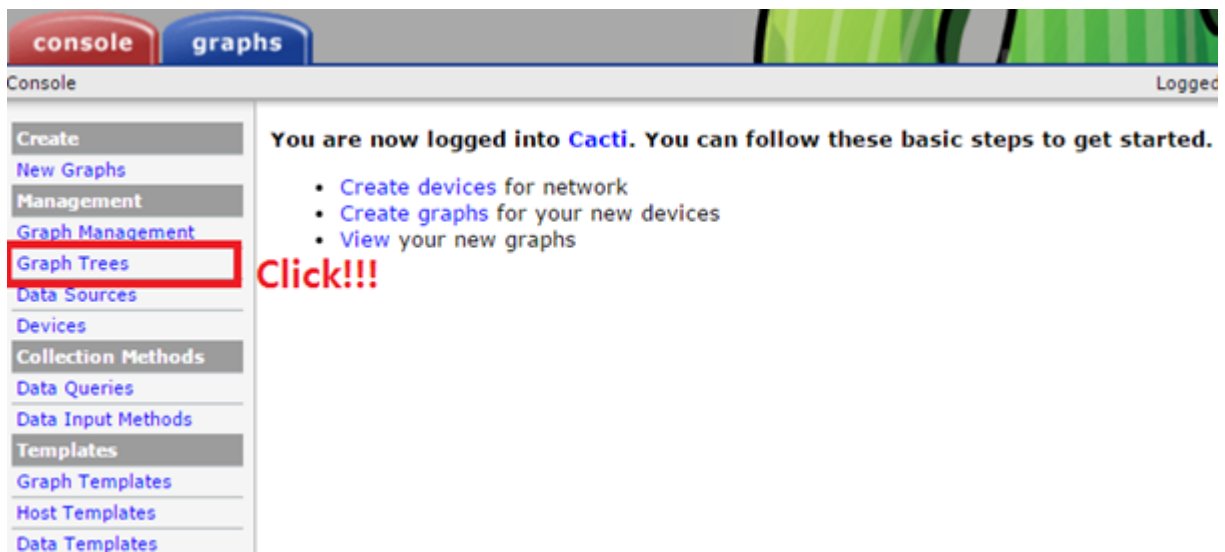


Fig. 6.11: Select [Graph Trees].

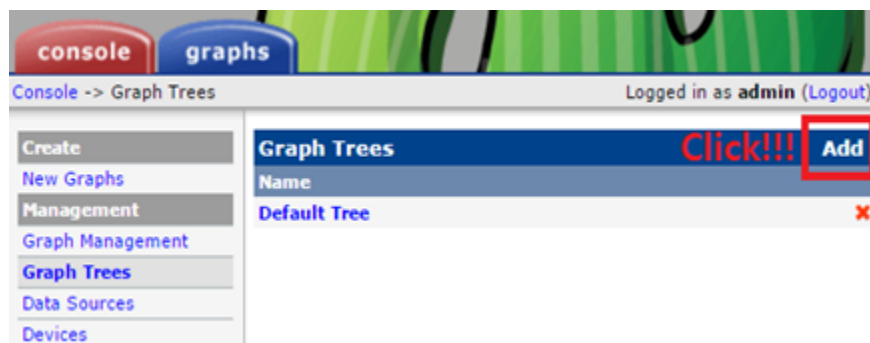


Fig. 6.12: Click the [Add] button on the right side.

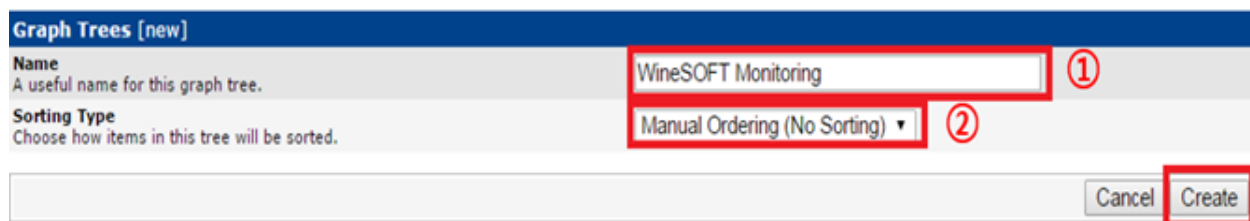


Fig. 6.13: Create Graph Trees.

Graph Trees [edit: STONTREE]

Name
A useful name for this graph tree. STONTREE

Sorting Type
Choose how items in this tree will be sorted. Manual Ordering (No Sorting) ▼

Tree Items Add Click!!!

Expand All Collapse All

Item	Value
No Graph Tree Items	

Return Save

Fig. 6.14: Click the [Add] button in the [Tree Items] menu.

Tree Items

Parent Item
Choose the parent for this header/graph. [root] ▼

Tree Item Type
Choose what type of tree item this is. Host ▼ 1

Tree Item Value

Host
Choose a host here to add it to the tree. STON (192.168.0.62) ▼ 2

Graph Grouping Style
Choose how graphs are grouped when drawn for this particular host on the tree. Graph Template ▼ 3

Round Robin Archive
Choose a round robin archive to control how Graph Thumbnails are displayed when using Tree Export. Hourly (1 Minute Average) ▼

Cancel Create

Fig. 6.15: Select the [Tree Items] options.

2. Select the devices you will add.
3. Select “Graph Template”.

6.2.4 Graph Check

Select [Graphs] in the upper left side to check if the graphs are displaying correctly.

6.3 Appendix C: Dynamic Page Exceptions

Dynamic pages are web pages developed using web programming languages (e.g. JSP, PHP), and they can change based on client requests. If the following types of dynamic pages are cached for appropriate lengths of time, it will create an exceptional decrease in the load for web servers, WAS, and databases.

- Real-time orderings (Trending search terms or rankings)
- Search results
- API for inquiries
- Detailed product pages (stock)

Pages that cannot be cached are as follows.

- **Private accounts** Caching a page means that if a person reads a page, other people can also read that page. Nobody wants their private information exposed to other people. From a technical standpoint, most web

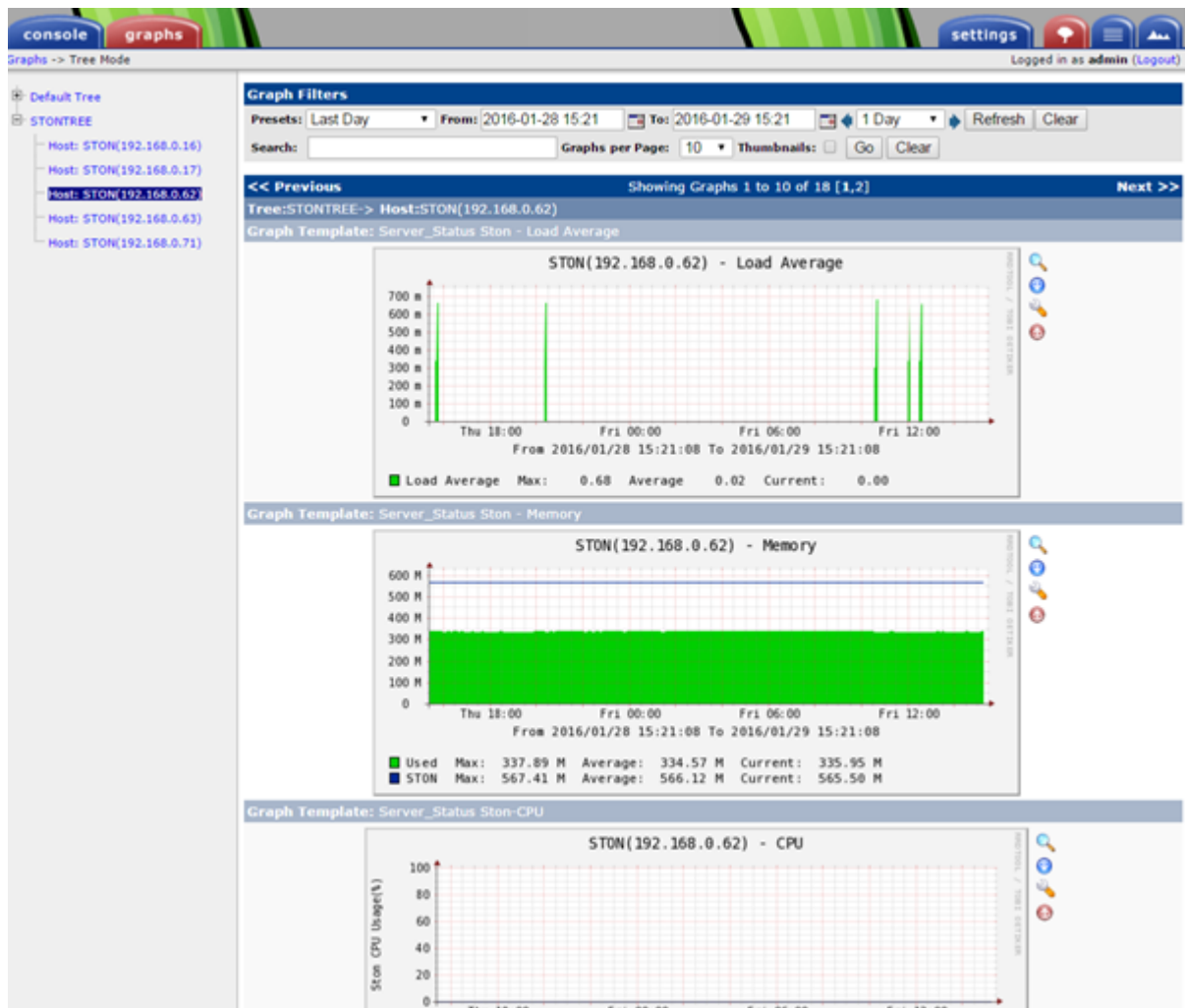


Fig. 6.16: Check regularly that the graphs are showing properly.

sites will not allow you to view private information without first logging in. The condition that checks for login status on web servers should be configured in STON's bypasses.

- **Payment** There are services that allow users to pay without logging in. All pages that are used for payment should be configured under caching exceptions.
- **Cookie creation** Cookies often contain information essential to the client. If a cookie created by User A is given to User B, a huge disaster will occur. Because of this, STON is configured to ignore all cookie headers given by the origin server.
- **Write API** The Read API can be effective with even 1 second of caching, but the Write API can malfunction if it is cached and a bad result is returned.
- **Ticket problems** Refers to reservations made in order like for concert tickets. There are many cases where this is done using WAS or a private server.

6.4 Appendix D: Release Notes

6.4.1 v2.4.x

2.4.3 (JAN 20, 2017)

Bug Fix

- Fixed infrequent Content-Encoding headers missing from compressed content responses

2.4.2 (JAN 18, 2017)

Feature/Policy Updates

- Vhost-Link feature added

Bug Fixes

- Fixed the unintended termination with negative Content-Length header value from origin servers
- Fixed the unintended termination from unstable origin server communications for MP3HLS packetizing

2.4.1 (NOV 24, 2016)

Feature/Policy Updates

- Processes origin HTTP responses with missing Reason-Phrase's
- DIMS: supports canvas resizing

Bug Fixes

- Compression integrity improved
- VLC media player compatibility for M4A HLS playback
- Abnormal termination from missing DIMS resize dimensions

2.4.0 (NOV 07, 2016)

Feature/Policy Updates

- Modify HTTP request URL to origin.
- Support M4A HLS

Bug Fix

- Enhanced processing for invalid MP4 size headers.

6.4.2 v2.3.x

2.3.9 (OCT 28, 2016)

Bug Fix

- TTL: Content was not updated few seconds in some circumstances

2.3.8 (OCT 13, 2016)

Bug Fix

- Enhanced processing for invalid MP4 size headers

2.3.7 (SEP 26, 2016)

Feature/Policy Updates

- DIMS: allocates system resource for image conversion
- Origin Health Checker: also includes stand-by origin servers

Bug Fix

- Compression on/off

2.3.6 (AUG 16, 2016)

Feature/Policy Updates

- Client socket processing policy update
- DIMS: PNG alpha compositing update for JPG conversion

Bug Fix

- Unintended termination from a Hardpurge API call in DIMS processing

2.3.5 (JUL 1, 2016)

Feature/Policy Updates

- Improved native HLS player compatibility
- DIMS image cropping in the unfixed aspect ratio

Bug Fix

- Unintended termination upon an origin status reset API call with Origin Health-Checker activated

2.3.4 (JUN 3, 2016)

Feature/Policy Updates

- Supports large MP4 files with 32-bit mdat atoms (4GB or more)
- Supports Host header value in unknown access logs
- WM installation : Apache Manual folder deleted for security
- WM installation : “winesoft” Apache account as nologin

Bug Fixes

- HLS: CPU overload upon some videos
- Termination upon bypassing HTTP requests
- Client IP shown as 0.0.0.0 in access logs
- Configuration backup failure for over 260 virtualhosts generated

2.3.3 (APR 26, 2016)

Bug Fixes

- Unintended 404 responses upon origin host, DIMS and compression configured [2.3.0 ~ 2.3.2]
- Unintended CPU overload upon SNMP View deletion
- WM - 0 value input error from SNMP GlobalMIn

2.3.2 (MAR 22, 2016)

Feature/Policy Update

- HLS index file compatibility improved

Bug Fixes

- Unintended termination from encryption/decryption with a bad SSL handshake
- Occasional termination from active ACLs

2.3.1 (FEB 23, 2016)

- Supports MP3 streaming in HLS

Feature/Policy Updates

- **Custom access log format**
 - %.y Request HTTP header size
 - %.z Response HTTP header size

Bug Fix

- WM : unintended failure in no destination port configured

2.3.0 (FEB 3, 2016)

- Supports on-the-fly compression.

Bug Fixes

- Expires Header: incorrect max-age value from modification
- DIMS Statistics: incorrect average value from faulty denominator

6.4.3 v2.2.x

2.2.5 (JAN 12, 2016)

Feature/Policy Updates

- HTTP response code update: “451 Unavailable For Legal Reasons”

Bug Fix

- TLS : unintended termination upon attacking packets

2.2.4 (DEC 11, 2015)

Bug Fix

- HLS : playback termination upon media segmentation

2.2.3 (DEC 4, 2015)

Bug Fix

- Virtualhost generation failure from Web Management in 2.2.2

2.2.2 (DEC 3, 2015)

- Modifies HTTP request header to origin

Feature/Policy Updates

- HTTP request/response header modification : ‘put’ action added, which inserts the header

2.2.1 (NOV 19, 2015)

Bug Fixes

- TLS-Handshake: overlapping ChangeCipherSpec return upon separate ChangeCipherSpec and ClientFinished messages
- *DIMS* : size ratio malfunction from Animated GIF resizing

2.2.0 (NOV 4, 2015)

- Supports TLS 1.2 (including Forward Secrecy and other security policy updates)

Bug Fixes

- Abnormal termination upon no disk information
- **TLS-Handshake version choice** **Before.** TLSPlaintext.version **After.** ClientHello.client.version

6.4.4 v2.1.x

2.1.9 (OCT 15, 2015)

Bug Fix

- *MP4 HLS* : Video playback malfunction from 2.1.7

2.1.8 (OCT 14, 2015)

Bug Fix

- Abnormal termination upon manager port access from blocked IPs (2.1.6 ~ 7)

2.1.7 (OCT 7, 2015)

- *Multi-Trimming* : Stitches multiple segments trimmed from the origin videos.

Feature/Policy Updates

- *Access Log* : Supports TrimCIP option for X-Forwarded-For header

Bug Fixes

- *MP4 HLS* : Video jittering from few profiles
- *DIMS* : B 500 Internal Error responses with zero TTLs
- Unintended space characters in X-Forwarded-For c-ip logging

2.1.6 (SEP 9, 2015)

Feature/Policy Updates

- *DIMS* : Converts only the first frames for *Animated GIF*

Bug Fixes

- *Chapter 14. Access Control* : IP access control malfunction
- *DIMS* : '+' coordinate malfunction for cropping images

2.1.5 (AUG 18, 2015)

- Virtualhost *Sub-Path* : Supports virtualhost sub-path by accessing paths
- *Facade Virtual Host*: Supports separate client traffic statistics and access logs by accessing domains

2.1.4 (JUL 31, 2015)

Feature/Policy Updates

- Less CPU usage
- https-multi-nic: listening on multiple NICs
- **URI policy change for Access Control Before.** keywords omitted (such as MP4HLS) from URIs **After.** the whole URIs

Bug Fixes

- *DIMS* : encoded strings unrecognized
- *HardPurge* : case-sensitive error
- Configuration History: POST request exception missing

2.1.3 (JUN 25, 2015)

Feature/Policy Updates

- *SyncStale* : All content control (*Purge* , *Expire* and *HardPurge*) API calls tracked and logged (synchronization with stale logs and index when restarted)
- %u expression added to *Custom Access Log Format* (full-length URIs from client requests logged)

Bug Fixes

- *DIMS* : image revalidation failure with no Last-Modified header from origin
- *Trimming* : CPU overload for >4GB trimmed MP4s
- Via header missing in error page responses

2.1.2 (MAY 29, 2015)

Web Management - English support

Feature/Policy Updates

- Single-core CPU support

Bug Fix

- Customized module malfunction in the *Indexing* mode

2.1.1 (MAY 7, 2015)

MP4 HLS : Provides bandwidth and resolution information in *StreamAlternates*

Bug Fix

- Abnormal termination caused by broken header MP4 video analysis

2.1.0 (APR 15, 2015)

Indexing added

Animated GIF *DIMS* supported

DIMS statistics supported

Feature/Policy Updates

- **Directory expression removed from *Chapter 5. Content Purge* (purge, expire, hardpurge, expireafter)**
URL by directory expression (example.com/img/) caches the returned file from the origin. Directory expression (example.com/img/) merged with pattern (example.com/img/*)
- **API expressions added**
 - /monitoring/average.xml
 - /monitoring/average.json
 - /monitoring/realtime.xml
 - /monitoring/realtime.json
 - /monitoring/fileinfo.json
 - /monitoring/hwinfo.json
 - /monitoring/cpuinfo.json
 - /monitoring/vhostslist.json
 - /monitoring/geoiplist.json
 - /monitoring/ssl.json
 - /monitoring/cacheresource.json
 - /monitoring/origin.json
 - /monitoring/coldfiledist.json
- WM - resolv.conf editing removed

6.4.5 v2.0.x

2.0.7 (JUN 25, 2015)

Bug Fixes

- *DIMS* : image revalidation failure with no Last-Modified header from origin
- *Trimming* : CPU overload for >4GB trimmed MP4s
- Via header missing in error page responses

2.0.6 (APR 28, 2015)

Feature/Policy Updates

- WM - resolv.conf editing removed

Bug Fix

- abnormal termination from MP4 analysis with broken headers

2.0.5 (APR 1, 2015)

Feature/Policy Updates

- Serves trimmed MP4 by HLS The following expressions trim the original media file (/vod.mp4) from 0 to 60 seconds and serve in HLS. | /vod.mp4?start=0&end=60/**mp4hls/index.m3u8** | /vod.mp4**/mp4hls/index.m3u8**?start=0&end=60 | /vod.mp4?start=0/**mp4hls/index.m3u8**?end=60
- HLS index file (.m3u8) update **Before.** Version 1 **After.** Version 3 (changeable back to version 1)

Bug Fixes

- abnormal termination in HLS conversion with HTTP encoded special characters
- overloaded CPU for MP4 media with broken headers
- audio/video synchronization failure while serving MP4 with uneven audio keyframe in HLS
- RRD - statistics bug: average response time shown in total
- WM - forcing formatting new disks removed

2.0.4 (FEB 27, 2015)

Feature/Policy Updates

- Hash algorithm update at *Origin Selection*
Before. hash(URL) / servers
After. *Consistent Hashing*
- Client requested URI is usable as a parameter when redirecting by *Virtual Host Access Control*.

Bug Fix

- Disk full due to unremoved caching files

2.0.3 (FEB 9, 2015)

Feature/Policy Updates

- DIMS internalization and enhancement
- WM - traffic alert messages added

Bug Fix

- WM - Virtual host generation failure

2.0.2 (Jan 28, 2015)

- Able to pass User-Agent header value from clients when requesting to the origin server.

Bug Fixes

- Failed to trim MP4 files with MDAT length 1.
- WM - failed to show other clustered servers' graph.
- WM - showing other clustered server's status as the relevant one.

2.0.1 (DEC 30, 2014)

Bug Fix

- No HitRatio graph value

2.0.0 (DEC 17, 2014)

- Disk space optimization: just as downloaded from the origins. (please refer to *Range Request*)
- *Memory Restriction* added
- TLS 1.1 support
- *SSL/TLS Acceleration* support by AES-NI.
- ECDHE CipherSuite support (please refer to *CipherSuite Selection*)
- *DNS Log* added
- Policy Update: Seprate TTLs for each IP if the origin server is configured by domain
- origin *Error Detection and Recovery* added
- origin *Health-Checker* added
- *System Free Memory* added
- etc.
 - Supported operating system updaated: CentOS 6.2 or later, Ubuntu 10.01 or later
 - NSCD daemon included in the installation package
 - *DIMS* included
 - Restart required after *Caching Reset*
 - <DNSBackup> removed
 - <MaxFileCount> removed
 - <Distribution> removed, integrated into *Origin Selection*



STON