Chapter 3
Server Load Balancing

Training Team

HCSA-ADC Official Training

Integrative Cybersecurity

Visionary. AI-powered. Accessible.

Agenda

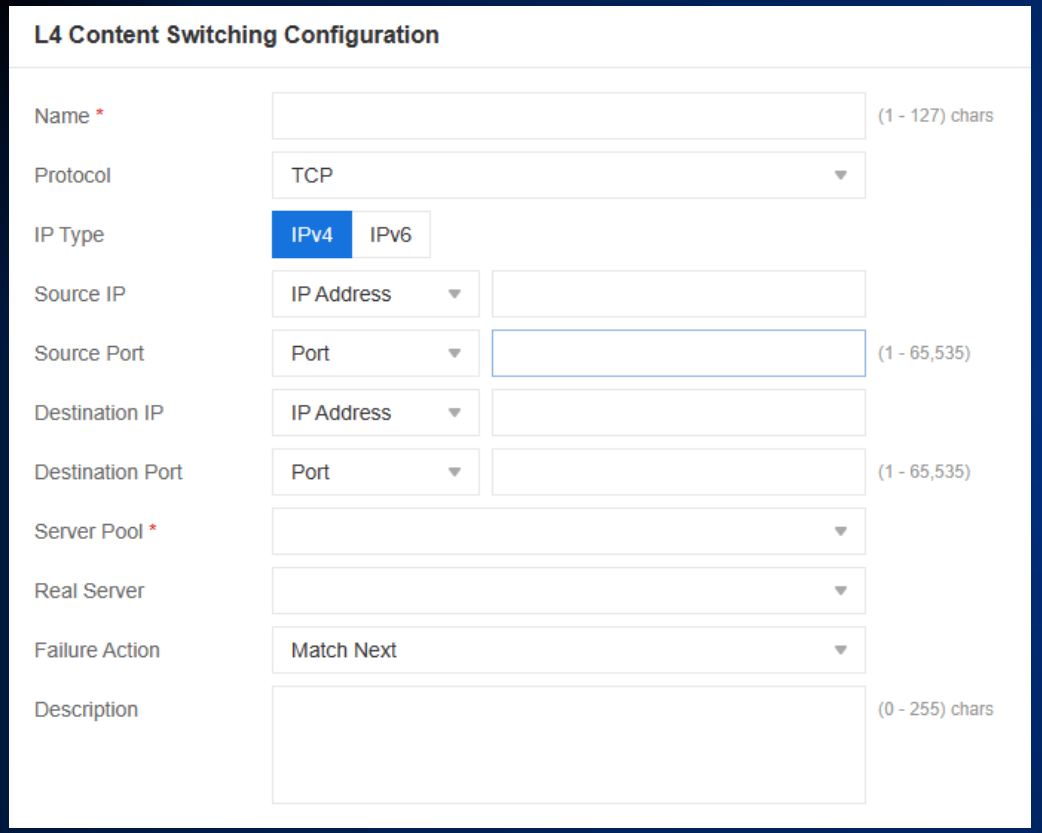
- Rule
- Application Module
- SSL Module

1 Rule

Content Switching

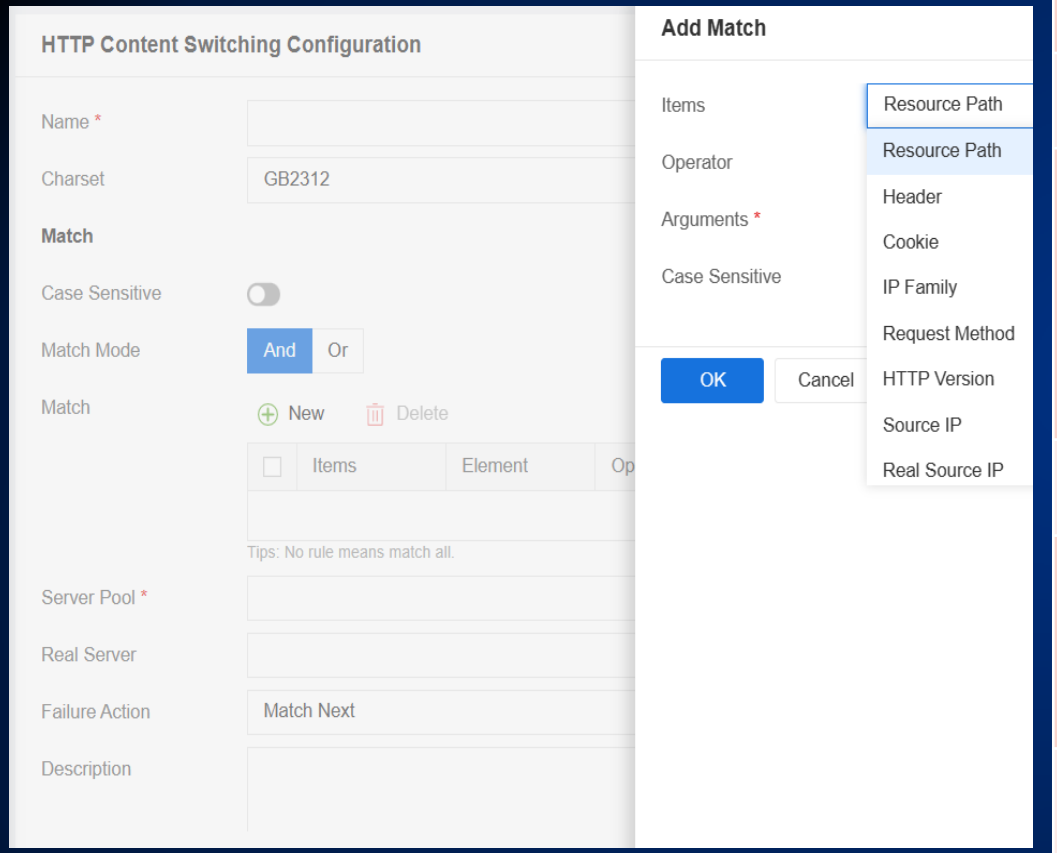
- In the chapter Load Balancing Algorithm, we introduced several modules that will schedule business, including content switching, load balancing algorithms, and session persistence. Content switching is a scheduling method that takes priority over load balancing algorithms. Taking routing as an example, routing also performs business scheduling. When there are multiple default routes, if we understand the default route as a service pool, multiple next hops are equivalent to multiple RSs. How to select a certain next hop among multiple next hops is equivalent to load balancing algorithms, while precise routing or policy routing corresponds to content switching.
- Content switching is an effective scheduling method that is superior to the balanced scheduling algorithm.
- Matching condition: The Hillstone ADC (Application Delivery Controller) is divided into two types of content switching: layer 4 content switching and HTTP content switching. The matching conditions for the layer 4 content switching are based on information such as IP, protocol, and port; for HTTP content switching, matching can be done based on content such as HTTP headers and URLs.
- Matching method: The matching method is similar to policy based routing. The rules are matched in sequence, and once the first matching rule is found, the result is returned without further matching.
- Matching results: After matching, you can choose either a pool or a specific RS in the pool. For the former, after the content switching selects a pool, it will then performs session persistence and balancing algorithms within the pool to ultimately select a specific RS in the pool; for the latter, it will directly selects a specific RS.

Layer 4 Content Switching – L4CS



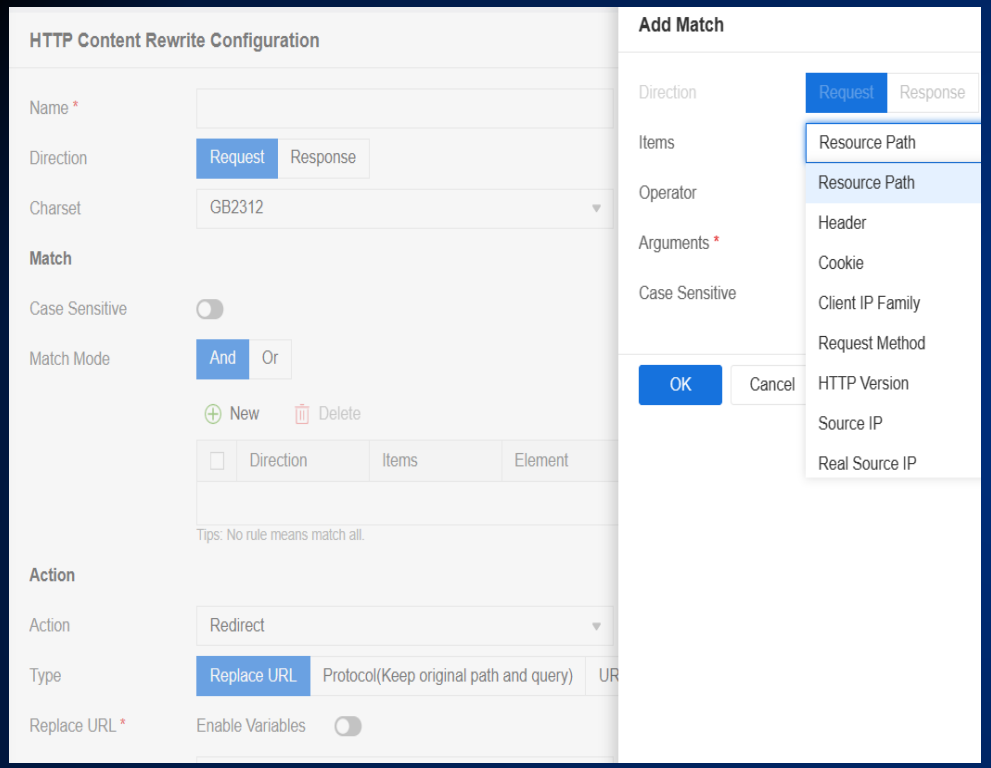
Config Item	Description
Protocol(*)	TCP or UDP
IP Type(*)	IPv4 or IPv6
Source IP(*)	Source IP or range, Not configured as “Any”
Source Port	Source Port or range, Not configured as “Any”
Destination IP	Destination IP or range, Not configured as “Any”
Destination Port	Destination port or range, Not configured as “Any”
Server Pool(*)	Business will be scheduled within the server pool based on matching conditions
Real Server	Same as service pool, it is a result of matching. If a real server is specified, the business is directly scheduled to that RS. If no RS is specified, session persistence and load balancing algorithms are used to schedule the service from the server pool.
Failure Action	When a service pool is specified but it is not available, or when an RS is specified but it is not available, it is considered to be a failure. At this point, you can choose to continue to matching or discard the request (respond with an error).

HTTP Content Switching – L7CS



Config Item	Description
Charset	When matching non-ASCII characters (such as Chinese), it is necessary to specify the Charset.
Match Mode(*)	"And" or "or", can be used to configure multiple matching conditions.
Match(*)	It is possible to configure matching conditions for resource paths, HTTP headers, IP addresses, ports, X509 certificate information, SSL information (version number, SNI, encryption suite), and other criteria. Within each matching condition, it is possible to individually configure whether case sensitivity should be ignored. Multiple matching conditions can be configured and combined.
Server Pool(*)	The matching conditions are used for business scheduling within the server pool.
Real Server	Same as service pool, it is a result of matching. If a real server is specified, the business is directly scheduled to that RS. If no RS is specified, session persistence and load balancing algorithms are used to schedule the service from the server pool.
Failure Action	When a service pool is specified but it is not available, or when an RS is specified but it is not available, it is considered to be a failure. At this point, you can choose to continue to matching or discard the request (respond with an error).

HTTP Content Rewrite



- HTTP Content Rewriting is a module that allows for the modification of the content of HTTP requests and responses. Content rewriting is a common scenario in application delivery products, where modifications may be required to accommodate for load balancing or other factors. For example, if a web page or backend program was not designed to work with load balancing restrictions, then content rewriting may be necessary after deploying an ADC. Another common scenario is during SSL offloading, where the Scheme seen by the browser is HTTPS, but the server uses HTTP. If the code contains absolute paths using HTTP as the Scheme, the page will not be loaded, then content rewriting is required. Similarly, during IPv6 migration, if the client accesses the website by an IPv6 address, the Host field will also be IPv6. If sent directly to the backend server, access may be restricted, which also require to configure content rewriting.
- Redirects are also a form of content rewriting, such as during HTTPS migration, redirecting HTTP requests to HTTPS.
- Redirection: Content rewriting is more complex than content exchange. It can be divided into "request rewriting" and "response rewriting". Matching conditions for request rewriting can only be for requests, while matching conditions for response rewriting can be based on both request and response contents.
- Match: From matching conditions' perspective, when rewriting a response, it is possible to use a combination of the request and response content as matching conditions for rewriting. Content rewriting uses line-by-line matching. Unlike content rewriting, when a request matches a rewriting rule, it will continue to match the subsequent rules.
- Action: Actions can include replace header, replace body, replace cookie, delete/insert headers or cookies.
- Original content can be considered a secondary matching condition, and content can only be rewritten if the original content matches the primary matching condition.
- New content can also support predefined variables in the Hillstone ADC system. This is useful for information such as client source IP, which may not be fixed and requires the use of variables for retrieval.

HTTP Content Rewrite

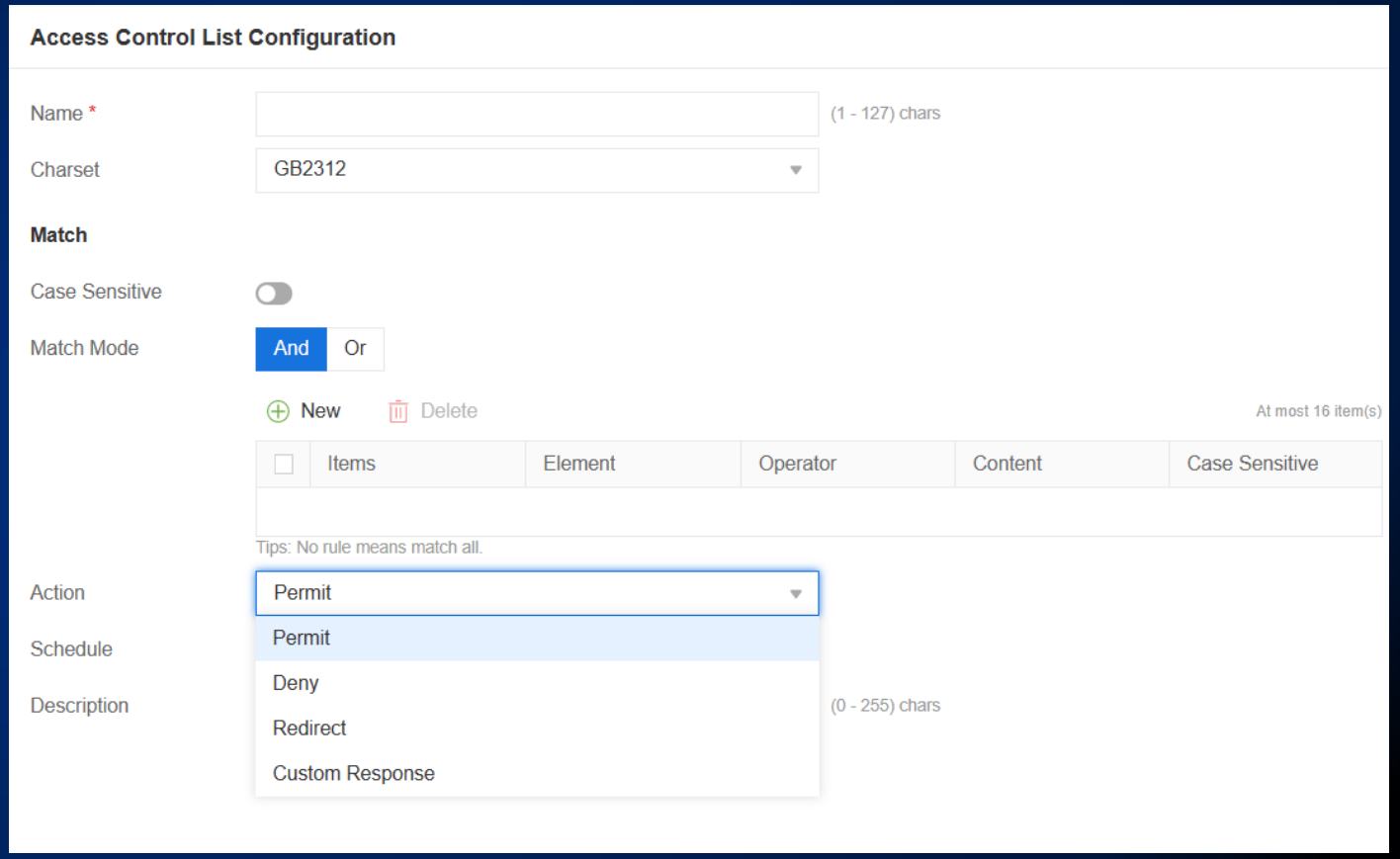
- Variables are used to describe some mutable objects that cannot be described with fixed values or ranges. They can currently be used in content rewriting rules.

- For all the variables you can use, please refer to the following link.

  https://www.hillstonenet.com/support/5.5R6/ADC/2.8/cn/Default.htm#18_SLB/SLB_rule.htm

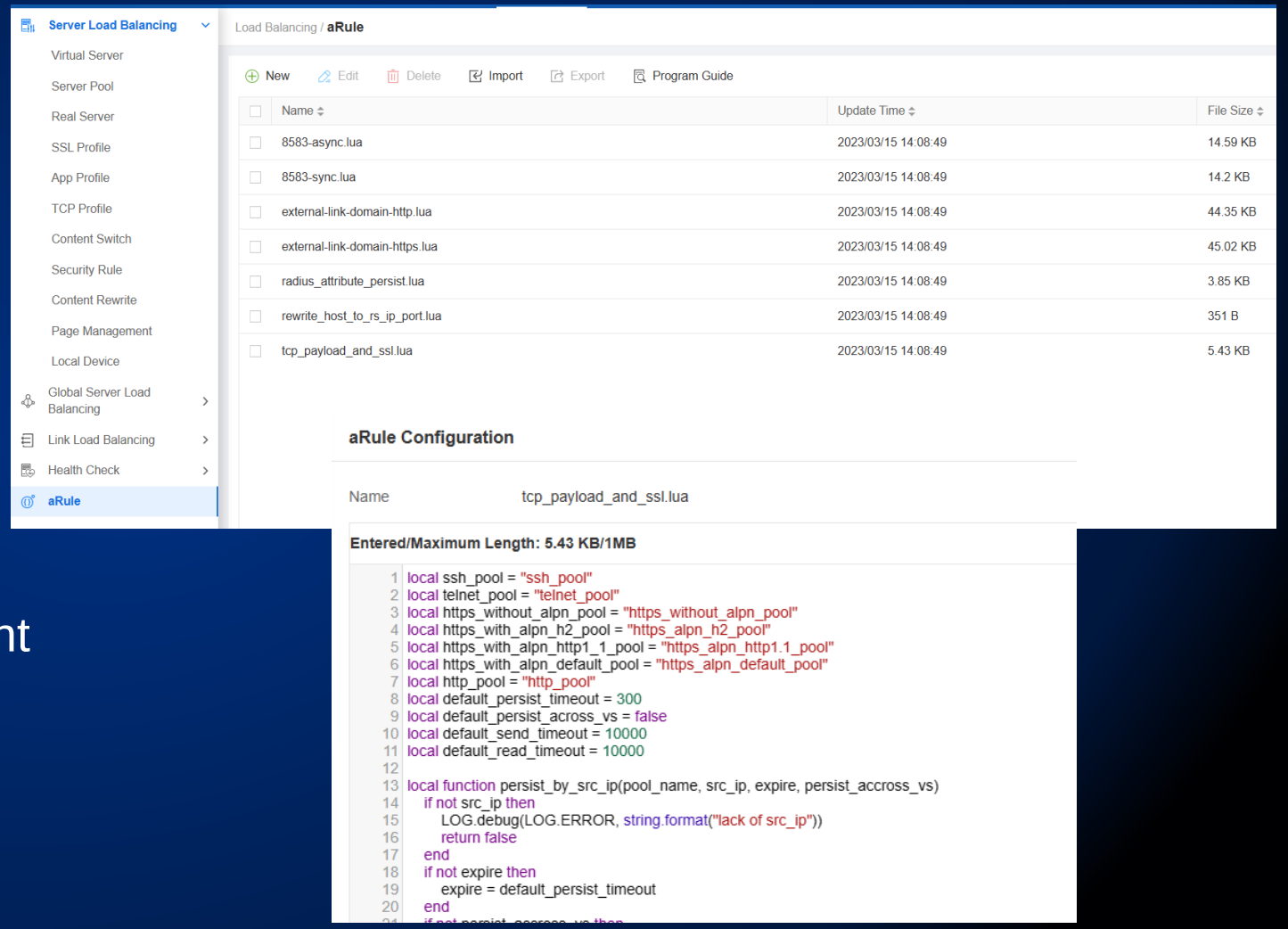
Access Control List - ACL

- The term 'access control' specifically refers to 'HTTP/HTTPS access control', which is used for permission management, etc.
- All matching content is essentially the same as 'HTTP content exchange'; the difference is that the result is not scheduling, but allowing, denying, redirecting, directly responding with information, and so on.

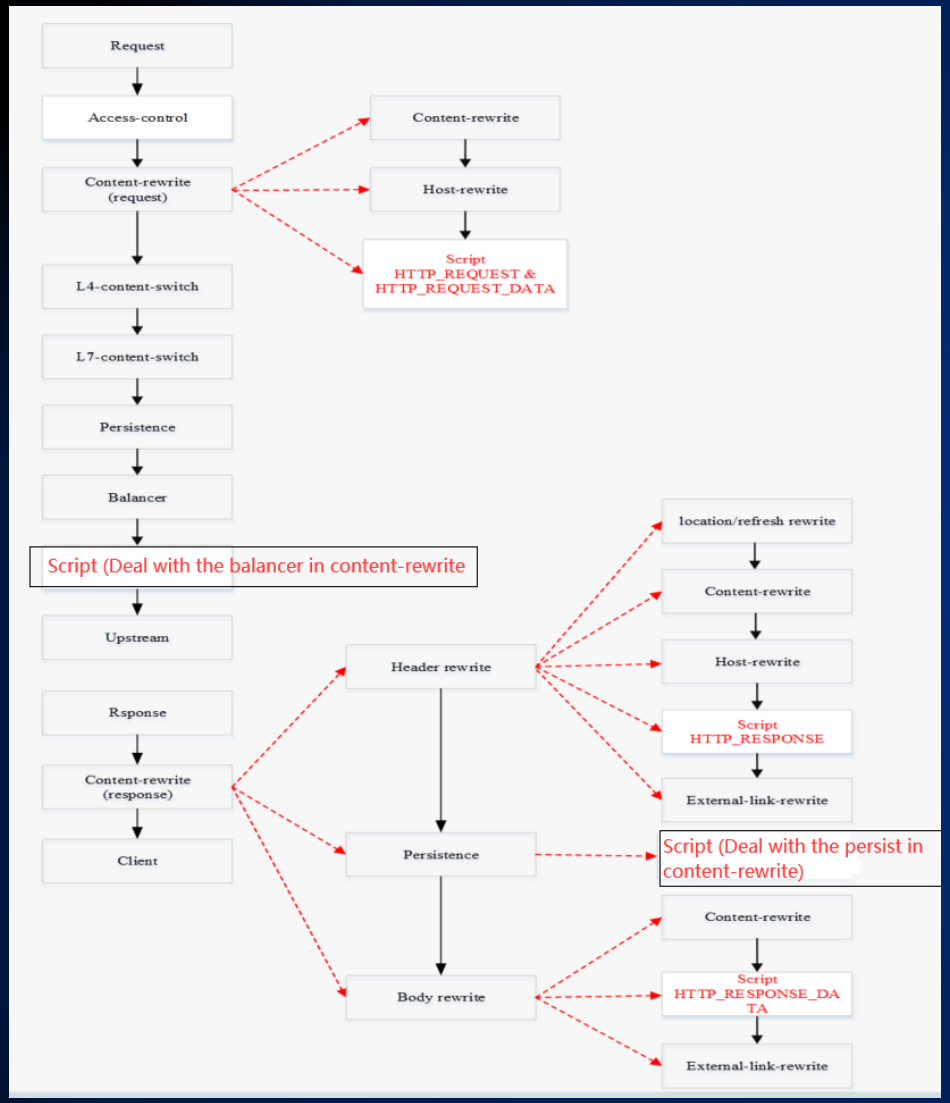


Programmable Scrip Function - aRules

- The programming language used for ADC programmable scripts is LUA, which can be edited and created online. Some other vendors only support importing, but not support for online viewing or editing.
- Programmable scripts are suitable for some complex personalized requirements, such as SSL business based on ALPN load balancing, financial industry requirements based on message load balancing, or frequent customer business changes. By modifying the script, the workload of modifying the configuration can be reduced.



SLB Business Process



2 Application Profile

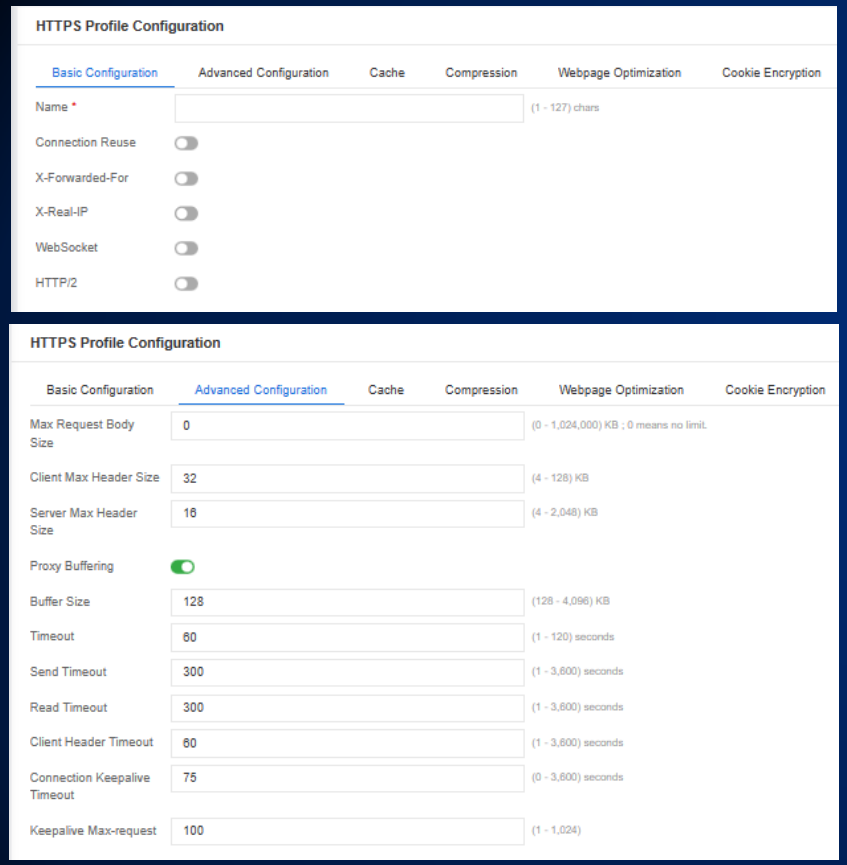
APP Profile

- For a specific application or protocol, there are specific or fine-tuned parameters. If all of these parameters are configured on a virtual server, the virtual server will become very complex and difficult to understand, and these configurations cannot be reused.
- By summarizing some of the parameters of a specific application, a set of parameter configurations is formed, which constitutes an application profile.
- An application profile can be called by multiple virtual servers.

APP Profile Types	Description
HTTP Profile	Configuring parameters for HTTP-based applications, such as connection reuse, X-Forwarded-For, various timeouts, caching, compression, WebSocket, etc.
HTTPS Profile	Similar to configuring HTTP profiles, configure parameters for HTTPS-based applications, as well as HTTP2 (which can only be used with HTTPS).
Fast HTTP Profile	HTTP-based applications, used for "Fast HTTP" virtual services, only TCP MSS can be configured.
HTTP Proxy Profile	A subset of HTTP profile, used for HTTP-based applications, specifically for "HTTP proxy" virtual servers.
Stream Profile	Configure some timeout parameters for "SSL stream proxy" and "SSL-OVER-TCP" virtual servers.
IMAP Profile	IMAP-based applications, configure cache size and read/write timeout.
POP3 Profile	POP3-based applications, configure cache size and read/write timeout.
SMTP Profile	SMTP-based applications, configure cache size and read/write timeout.

HTTP/HTTPS/HTTP Proxy Profile

The following three timeout parameters are the most frequently adjusted parameters in actual deployment, especially the "Read Timeout". The vast majority of 504 Timeouts encountered in practice are due to this reason.



Config Items	Description
Connection Reuse	Used to control whether ADC and RS use HTTP1.1 or HTTP1.0. The default is HTTP1.0, but when enabled, HTTP1.1 is used instead.
X-Forwarded-For	Insert X-Forwarded-For header, if the original request already contains X-Forwarded-For, add the source IP content to the right of its value.
X-Real-IP	Insert X-Real-IP header
Timeout	If ADC and RS cannot establish a TCP connection successfully within this time, it is considered a timeout. Generally, as long as the network is not unreachable, a connection can be established within the time configured by default. This situation rarely occurs.
Send Timeout	When ADC cannot successfully write data to RS for more than this time, it is considered a timeout. This is commonly seen when ADC sends content to RS, but due to some reasons, the program on RS does not read the data, causing the data to accumulate until there is no more available buffer, and ADC cannot continue to write successfully.
Read Timeout	When ADC cannot read any data from RS during this time, it is considered a timeout. Read timeout is the most common timeout encountered in practice. This often occurs when the program on the RS side needs a long time to generate a file, during which RS cannot respond to ADC, resulting in a read timeout.
HTTP/2	Enabling support for the HTTP/2.0 protocol, only supported by HTTPS templates because HTTP/2 must work on HTTPS.
WebSocket	Enabling WebSocket service allows HTTP/HTTPS and WebSocket services to be provided on a port at the same time.

HTTP/HTTPS/HTTP Proxy Profile

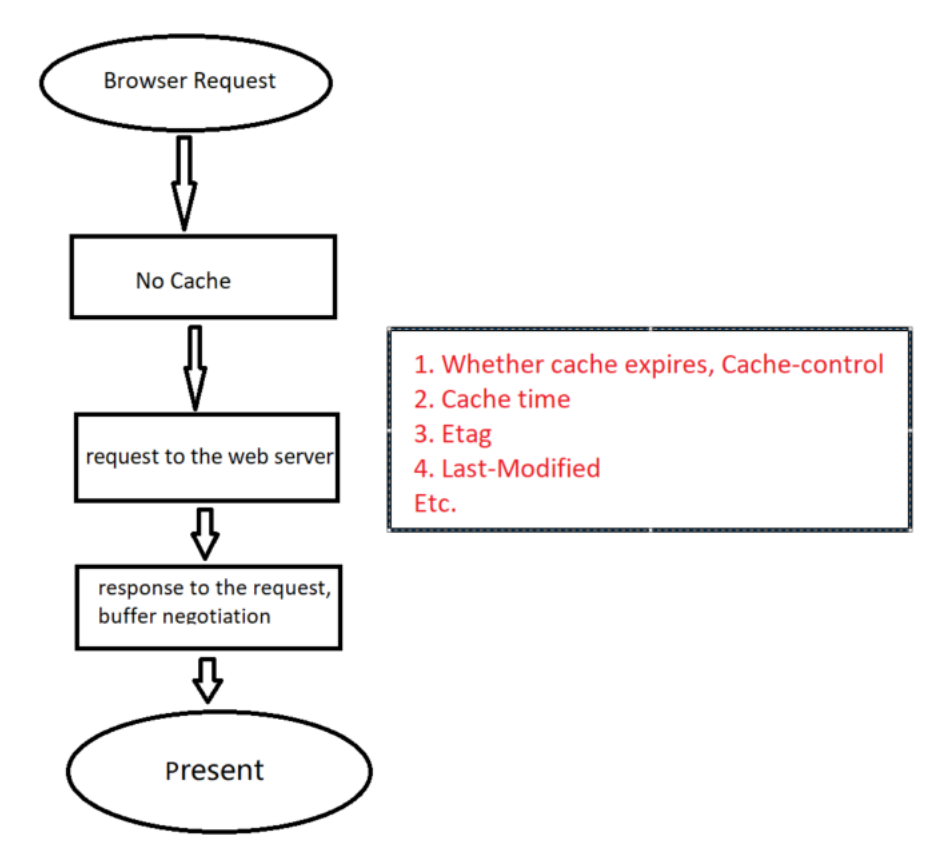
Config Item	Description
Connection Keepalive Timeout	When the data between client and ADC are transferred by HTTP1.1, if there is no new request within this time, the TCP connection will be closed.
Keepalive Max-request	When the data between client and ADC are transferred by HTTP1.1, it limit the number of HTTP requests/responses can be made in one TCP connection.
Max Request Body Size	This is used to limit the maximum size of the request body, generally used to control the size of uploaded files. The default value is 0, which means that there is no control. If a value is set and the request body size exceeds the configured value, the ADC will return a 413 error.
Proxy Buffering, Buffer Size	These are mainly related to the buffer size when the ADC acts as a proxy. Larger buffers can cache more content in memory, which can lead to higher performance, but also higher overhead. In general, there is no need to modify these parameters.

Introduction of Cache

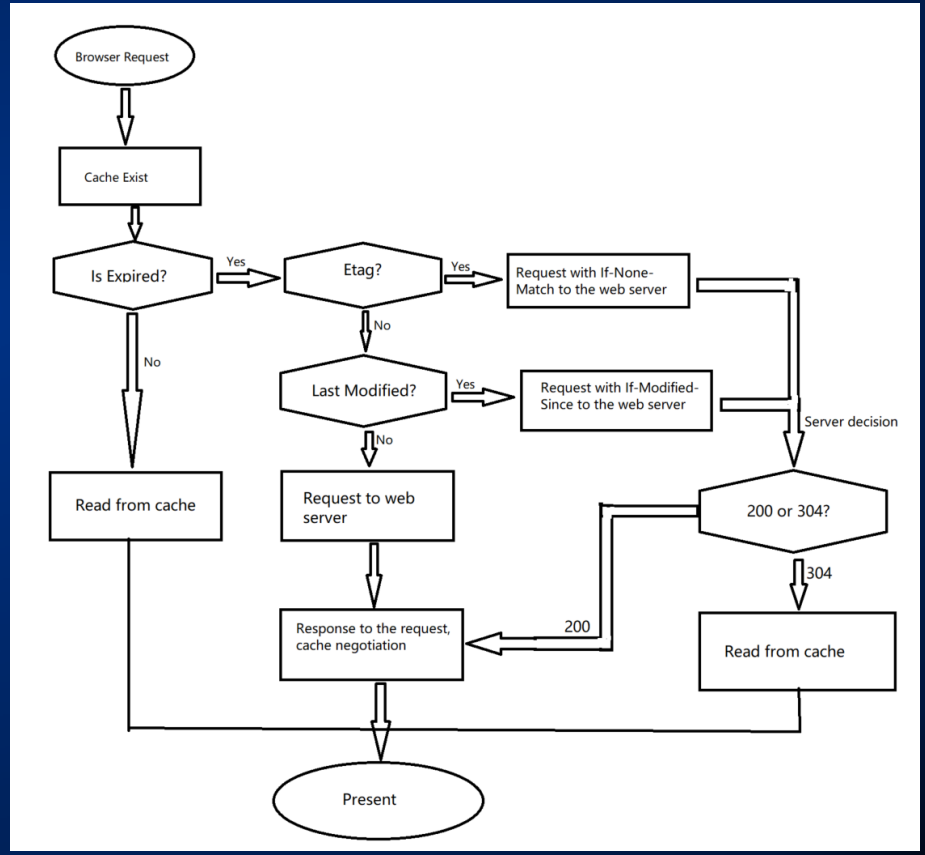
- Caching is the process of storing server content on the ADC after a client accesses a service through the ADC. When the client requests the same URL or resource again, the ADC can directly return the cached content to the client, without requesting it from the backend server again. This reduces server load and saves on interactions between ADC and server.
- Generally, dynamic content (such as PHP, JSP, and other program-generated content) cannot be cached because the content changes for each request. However, static files such as images, HTML, CSS, and videos do not change frequently and are suitable for caching.
- Cache expiration time: Although static files change less frequently, they still have possibility to be updated occasionally. So how ADC judge whether there is a update? HTTP defines a cache protocol, and ADCs can use cache timeouts to detect content changes.
- Caching can be classified into forced caching and comparison caching. Forced caching mainly uses time rules to determine whether a cache is valid, while comparison caching is more accurate and can use both time and content to determine cache expiration. HTTP headers related to forced caching include Expires and Cache-Control, which control how long the cache is valid and whether caching is allowed. Headers related to comparison caching include Etag, If-None-Match, Last-Modified/If-Modified-Since, etc.
- ADCs support both memory caching and disk caching. Reading data from memory is faster than reading data from disk. Therefore, when the demand for response speed of page or resources is high, memory caching is preferred. When both memory and disk caching configurations meet the requirements, the device prioritizes memory caching.

Cache Principle

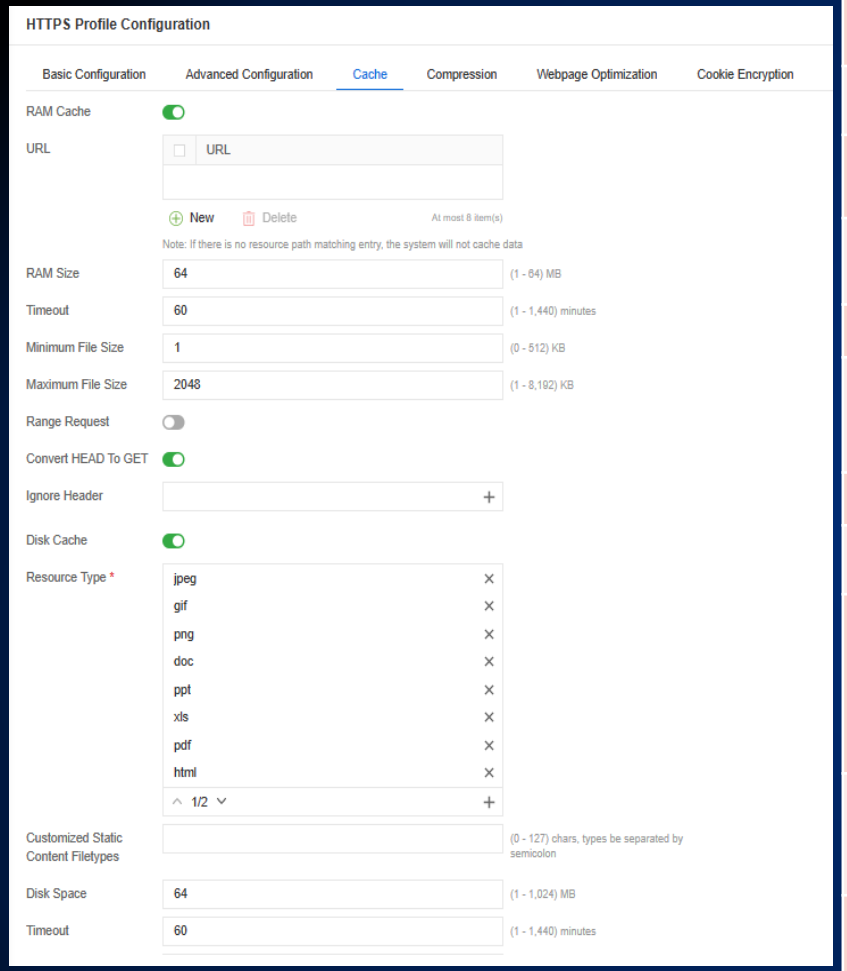
First Request



Request Again



Cache



Config Item	Description
URL Match	Match client requests through regular expression matching of resource paths and enable memory caching. (Memory caching is supported.)
Resource Type	Because only cache static resource, the type of resources that need to be cached is configured here.
Ignore Header	Some pages may instruct the client (proxy) not to cache. By configuring this option, these headers can be ignored and forced caching can be performed.
Customized Static Content Filetypes	In addition to the predefined selectable resource types, users can also specify other types of files to be cached based on their application needs.
Disk Space	The maximum cache space, when it is exceeded, older items will be deleted.
Timeout	When the cache exceeds this time, it expires. When the ADC receives a new request and finds that the data has expired, it will request an update from the RS. If there have been no new requests during this time, the old data will be deleted.
Minimum File Size	For pages smaller than this size will not be cached.
Maximum File Size	For pages bigger than this size will not be cached, the main reason is the cache space is limited.
Default Page	For the default page, which means accessing the "/" path, it is not known whether the file type can be cached. Therefore, users need to determine whether caching is necessary based on their actual situation. For example, if the default page is a static page like index.html which rarely changes, it can be cached. However, if the default page is index.php and is dynamically generated each time with unique content, it is not suitable to enable caching for the default page.
Range Request	Some applications, such as P2P download clients, use the HTTP Range method to retrieve part of the content in a resource, and finally splice it into a complete content through multiple requests. Enabling this function will also cache the form of accessing resources using HTTP Range.
Convert Head to Get	Specify the caching method of HEAD requests for ADC. By default, this feature is enabled, which converts HEAD requests to GET requests and caches them in ADC, while responding to both HEAD and GET requests from clients. Disabling this feature will cache HEAD and GET requests separately and respond to clients accordingly.

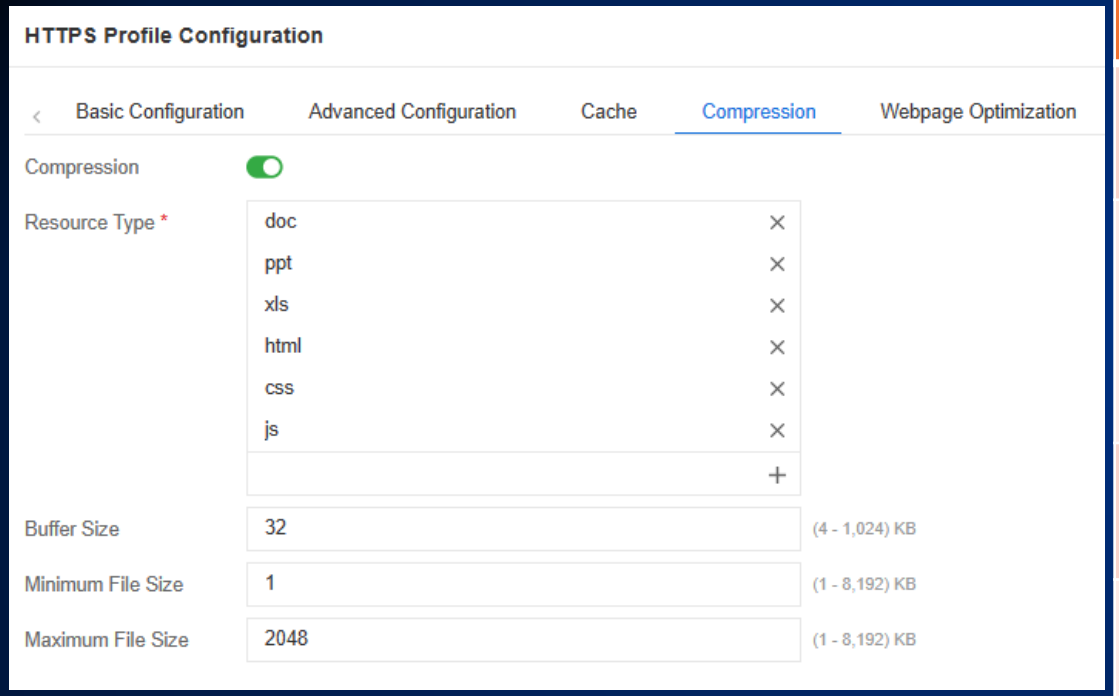
Introduction of Compression

- The HTTP protocol itself supports compressed transmission. When the server compresses the content, the data becomes smaller, which can speed up the transmission. After the client (such as a browser) receives the response, it decompresses the content and then displays it.

- For ADC devices, when compression is enabled, the uncompressed content returned by the server will be compressed and transferred to the client. After the client receives the compressed content, it will be decompressed and displayed. The benefits of compression are obvious. When the client's network bandwidth is insufficient, significant results can be achieved. However, the downside is also apparent. Compression requires a lot of computing resources on the ADC. If the client's ADC device has a high CPU usage rate, compression is not recommended.

- In addition, for multimedia files such as images and audio, the file format itself has already been compressed. If compression is performed through ADC, the compression rate is very low, which will consume ADC's CPU resources without economic benefits. In reality, only text types such as CSS, HTML, and JS have a relatively high compression rate and are more meaningful to compress.

Compression



Config Item	Description
Minimum File Size	Content that is smaller than this size will not be compressed, as compressing such small content has little benefit.
Maximum File Size	Content that exceeds this size will not be compressed. Typically, document content is not very large, and compressing very large content can consume too much CPU resources on the ADC and result in slow decompression speed on the client side, affecting the user experience.
Resource Type	File type that will be compressed
Buffer Size	Size of memory allocated during compression. Larger sizes lead to higher efficiency, but also result in greater memory usage. Typically, this parameter is not adjusted.

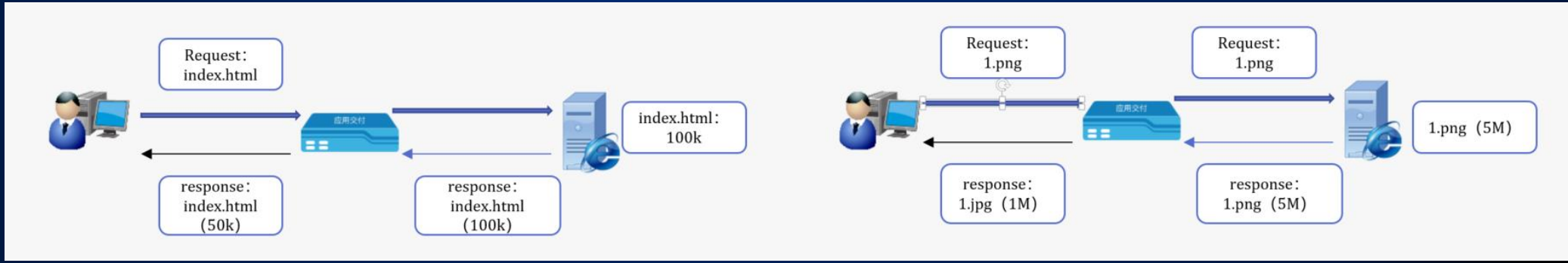
Introduction of Webpage Optimization

Web developers often add a lot of comments to their code to make it easier for others to read, or they may have some bad habits like typing in a lot of spaces or line breaks in the code or between lines, which causes the page to be bloated, which occupying a lot of bandwidth and affecting data transmission speed.

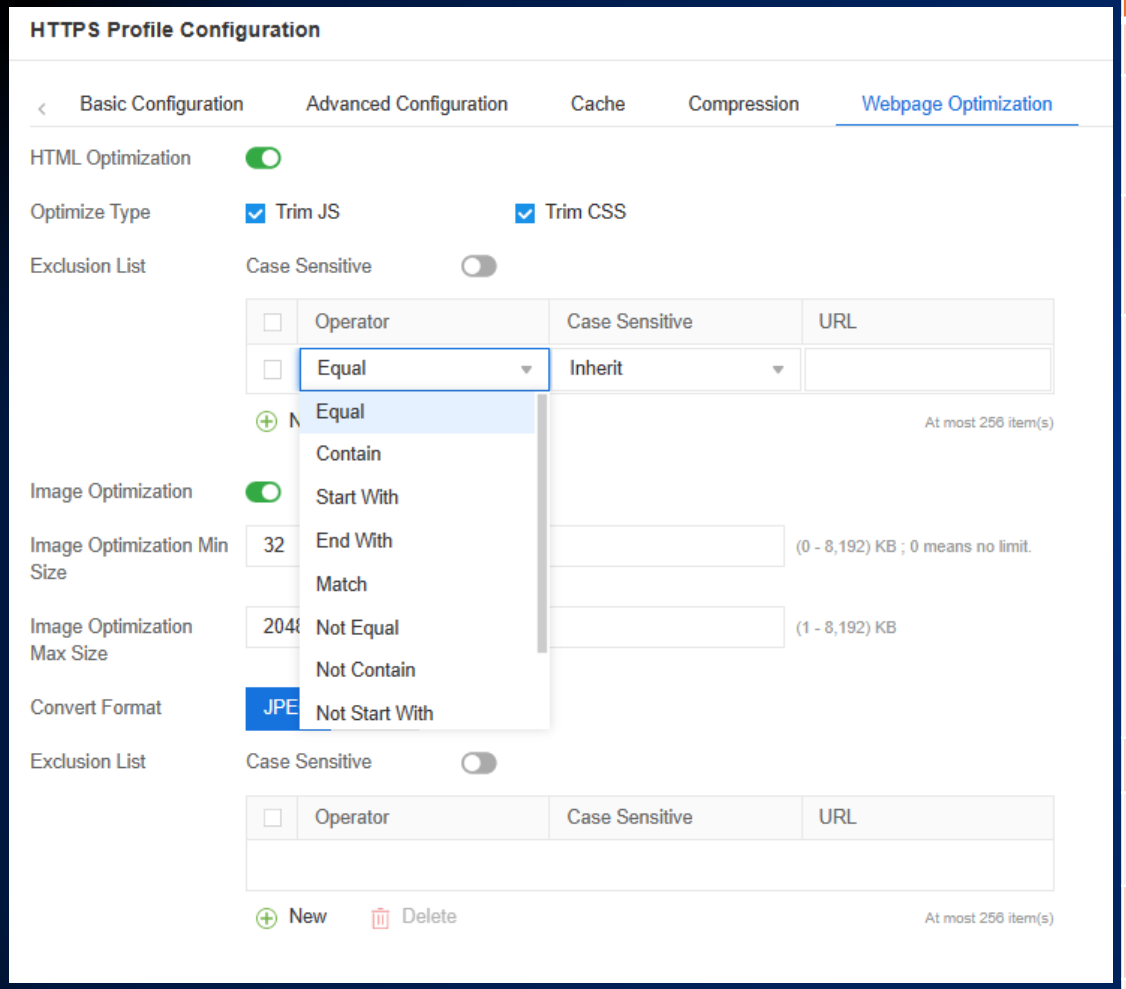
- By using webpage optimization features, you can remove large number of useless comments and extra white spaces and line breaks in the webpage source code, which can speed up the webpage, optimize bandwidth, and reduce latency.

When there are a large number of images on the webpage, and some of the images are very large. Due to the number of images to be loaded and internet speed and bandwidth issues, these images load very slow, which negatively affects customer's internet experience.

- By Optimizing image format and automatic image transcoding, we can reduce image size, improve web page loading speed, reduce bandwidth usage, and enhance the user experience of website visitors.

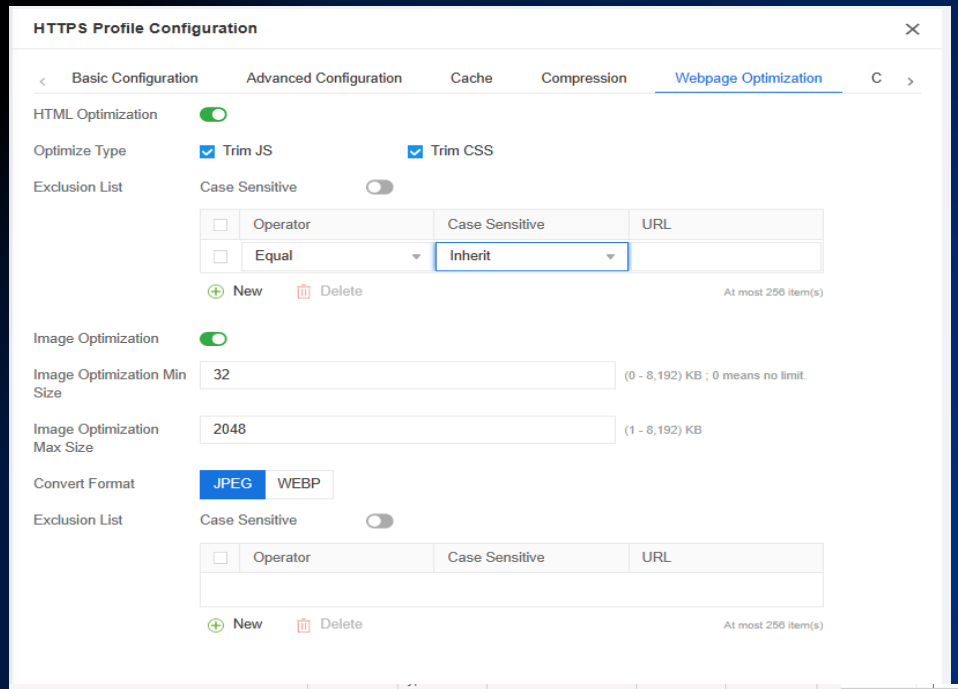


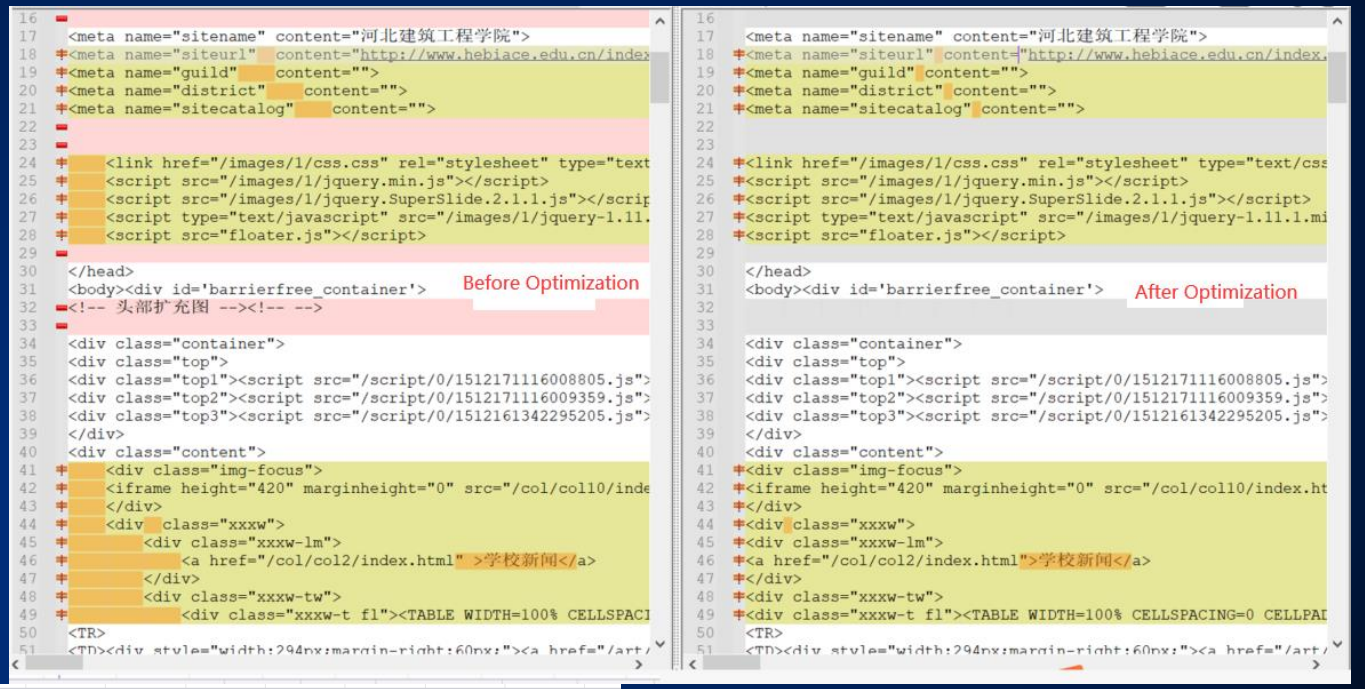
Webpage Optimization

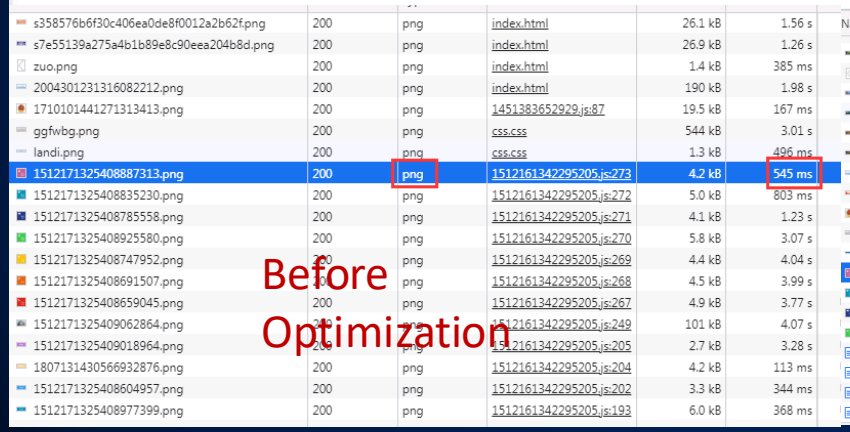


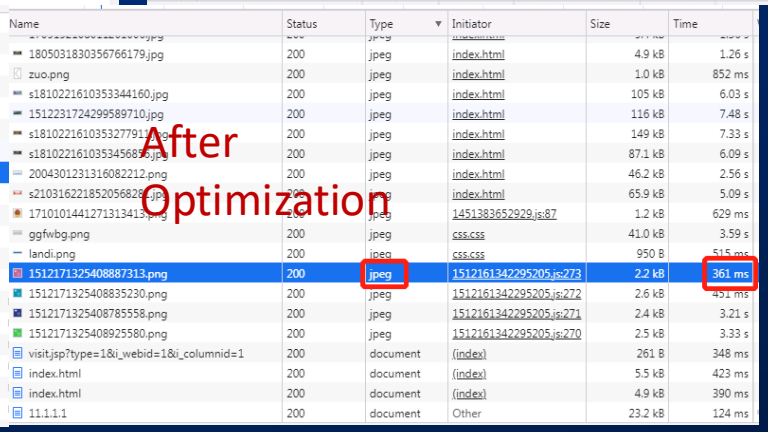
Config Item	Description
HTML Optimization	HTML page optimization enable switch
Optimize Type	JS optimization: After enabling, it can optimize the JS code in HTML pages. CSS optimization: After enabling, it can optimize the CSS code in HTML pages.
Exclusion List Case Sensitive	When matching paths is in the exclude list, is case sensitivity considered? Whether the resource paths in the exclude list are case sensitive when selecting 'inherit' is depends on this switch.
Exclusion List	By configuring the exclude list, the traffic of the matched exclude list entries will not be optimized. Operator: The way to match paths. The commonly used 'equal to' means exact matching, 'contains' means fuzzy matching, and 'match' means regular expression matching. Case sensitivity: Whether the entry matching is case sensitive can be defined. If it is not case sensitive, the case configuration in the entry and the case of the actual accessed path will be ignored. 'Inherit' can be selected to be controlled globally. URL: The path accessed by the user. For example, if the client accesses https://hillstonenet.com/image/123.jpg, then the resource path is /image/123.jpg, and it can be matched according to the operator.
Image Optimization	Image Optimization enable switch
Image Optimization Min Size	Images smaller than this size will not be optimized.
Image Optimization Max Size	Images bigger than this size will not be optimized.
Convert Format	You can choose to convert images to JPEG or WEBP formats.

Webpage Optimization









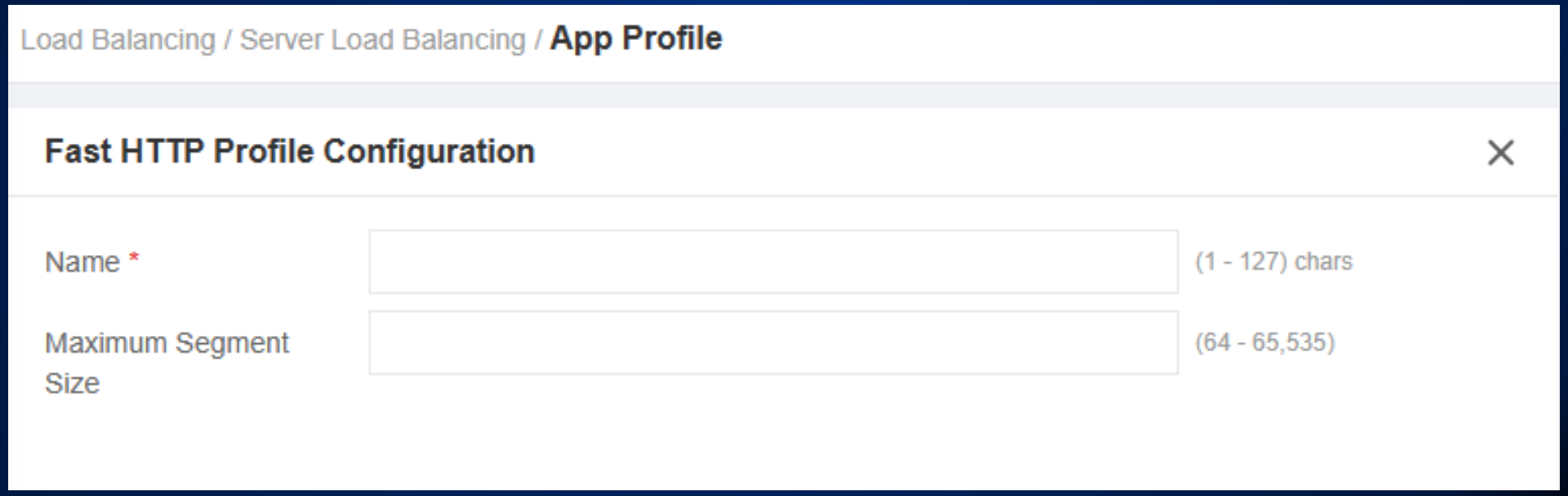
Comparison before and after enabling webpage optimization:

1. Unused blank lines and extra spaces in the webpage source code were removed.
2. PNG images were changed to JPEG format.
3. The loading speed of the same image decreased from 545ms to 361ms.

Fast HTTP Profile

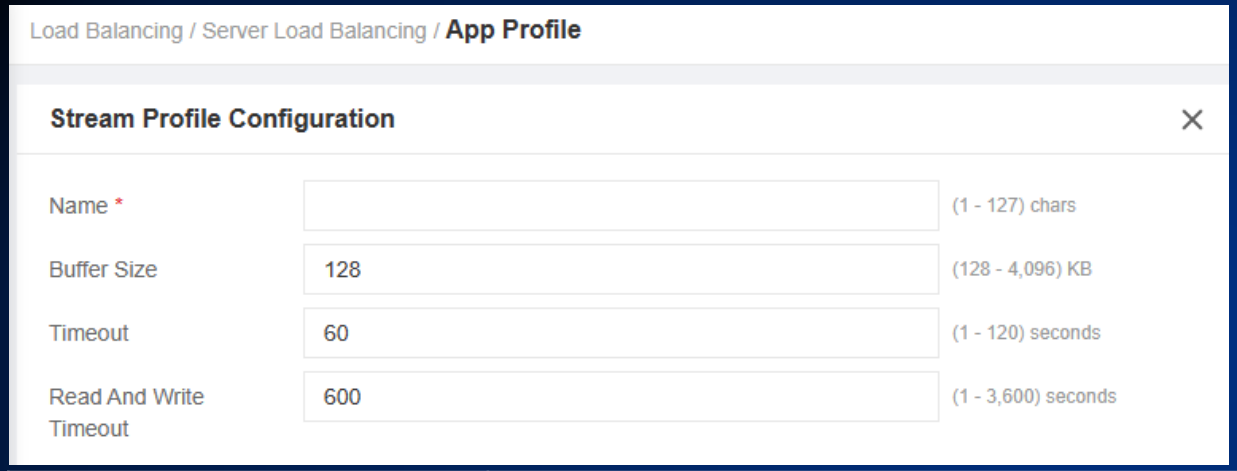
The Fast HTTP profile is designed for use with virtual servers “Fast HTTP" type virtual servers. Fast HTTP is based on a lightweight HTTP proxy and does not support content rewriting. It only performs business scheduling based on the first HTTP request.

Currently, the Fast HTTP profile has very few configurable options, and only supports configuring the "maximum segment size" (TCP MSS).



Stream Profile Configuration

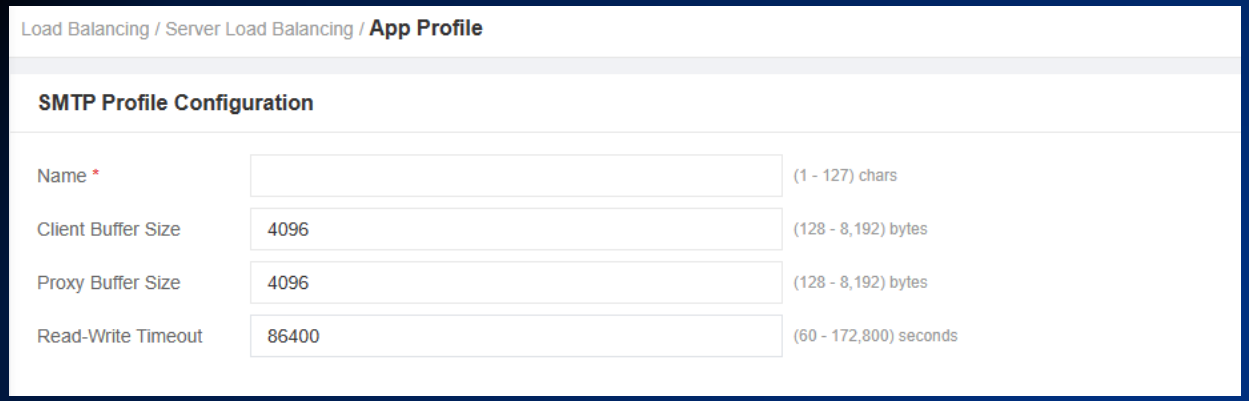
- Stream Profile is used for stream-based virtual servers such as "SSL stream proxy" and "SSL-OVER-TCP"



Config Item	Description
Timeout	If the TCP connection cannot be successfully established between ADC and RS within the specified time, it is considered as a timeout. Generally, as long as the network is reachable, the connection can be established successfully within the default configured time, and this situation rarely occurs.
Read and Write Timeout	If ADC cannot read any data from RS or cannot successfully complete any data writing (due to RS not reading and causing the buffer to be full) during this period, it is considered as a timeout.
Buffer Size	These parameters are mainly related to the buffer size when ADC acts as a proxy. A larger buffer size allows for more content to be cached in memory, which can result in higher performance, but also comes with a higher overhead. In general, there is no need to modify this parameter.

IMAP/POP3/SMTP Profile

- Used to called by the virtual server with correspondent protocol .



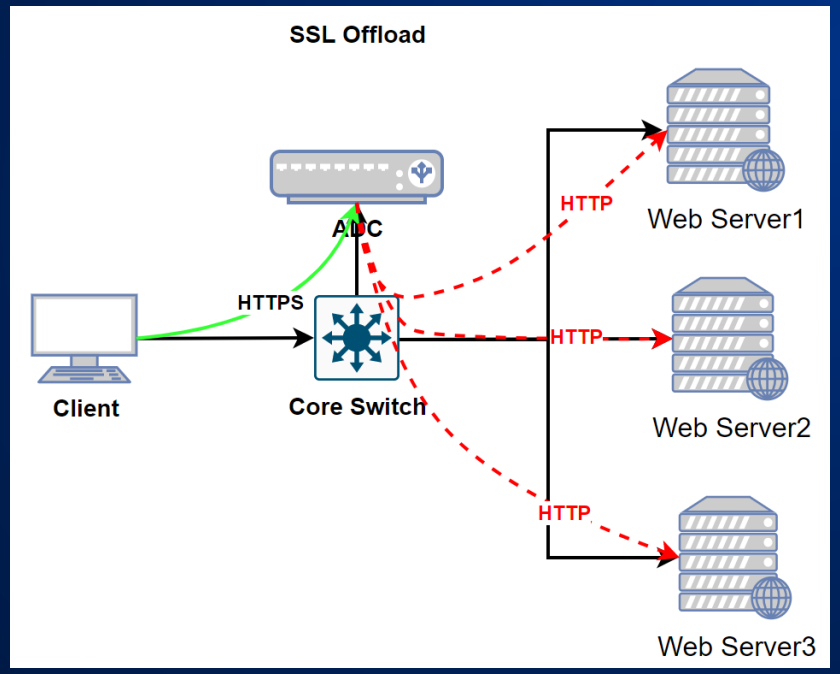
Config Item	Description
Client Buffer Size	Specifies the buffer size for receiving client request messages on the device. This item is not supported by the POP3 profile and generally does not need to be modified.
Proxy Buffer Size	The buffer size for the interaction between AX and RS. Generally, it does not need to be modified.
Read-Write Timeout	Specifies the timeout for two consecutive reads (writes) between the client and AX, as well as the timeout for two consecutive reads (writes) between AX and RS. Generally, it does not need to be modified.

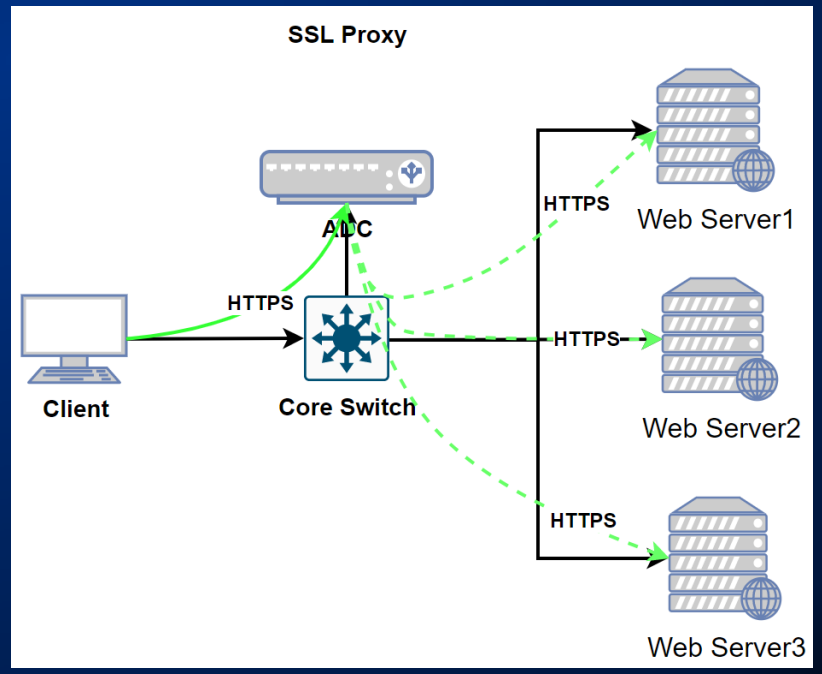
3

SSL Profile

Client SSL Profile

- The two scenarios described below require SSL decryption by the ADC, whether it's SSL offloading or SSL proxy.
- The term "client SSL profile" refers to the perspective of the client who initiates the connection, in which case the ADC is actually acting as the SSL server. The client SSL profile needs to be configured for both SSL offloading and SSL proxy. In other words, whenever the ADC needs to perform SSL decryption, the client SSL profile must be configured.





GMSSL algorithm

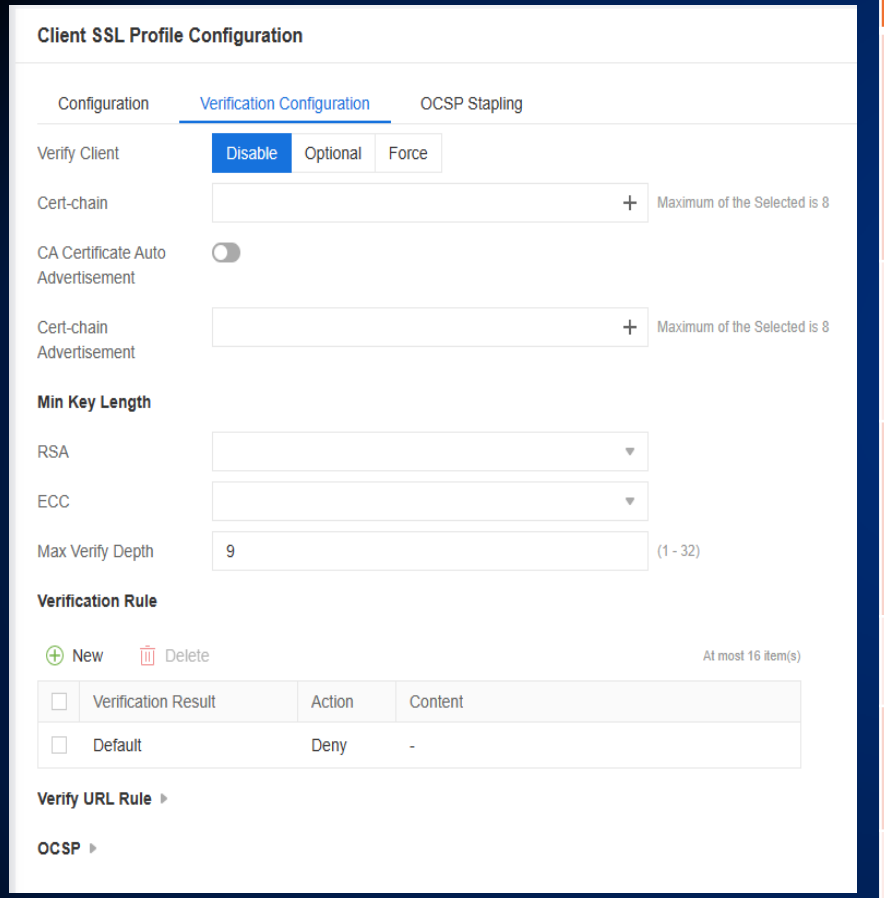
The corresponding algorithm to commercial cryptography is the GMSSL. During the process of transforming a website to support GMSSL, it is not widely supported by client-side devices. Therefore, during deployment, commercial cryptography and GMSSL are generally used together. The final decision of whether to use commercial cryptography or GMSSL is based on the protocol used by the client-side device.

The Hillstone ADC supports the configuration of RSA certificates, ECDSA certificates, and GMSSL certificates on a single virtual server (website). Based on the protocol and suite supported by the client, the final protocol and suite used can be smoothly transitioned to either GMSSL or ECDSA certificates.

Config Item	Description
Cipher Suite(*)	GMSSL cipher suite; As of the current version, ADC only supports the suite of SM2-SM4-SM3 (ECC-SM4-SM3).
Signature Cert-chain/Encryption Cert-Chain(*)	In the GMSSL protocol, the signature certificate and encryption certificate can be two separate certificates.

Verify Client

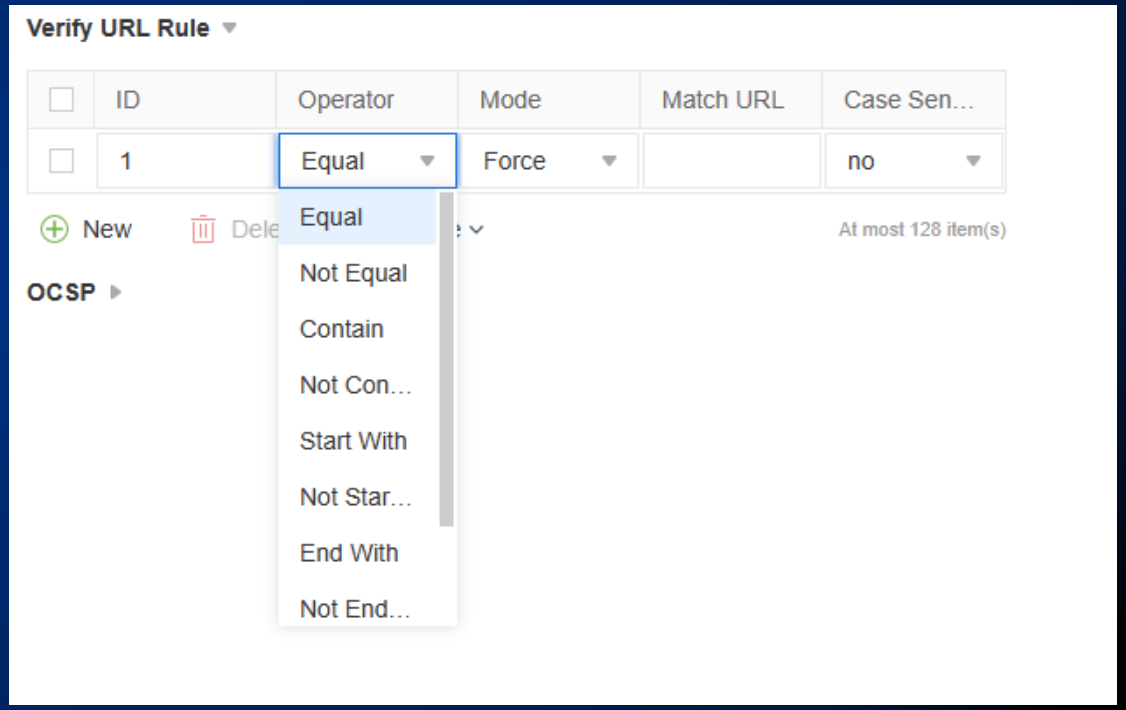
When ADC is deployed in front of SSL VPN, and client identity (certificate) validation is required, the “Verify Client" configuration needs to be set up.



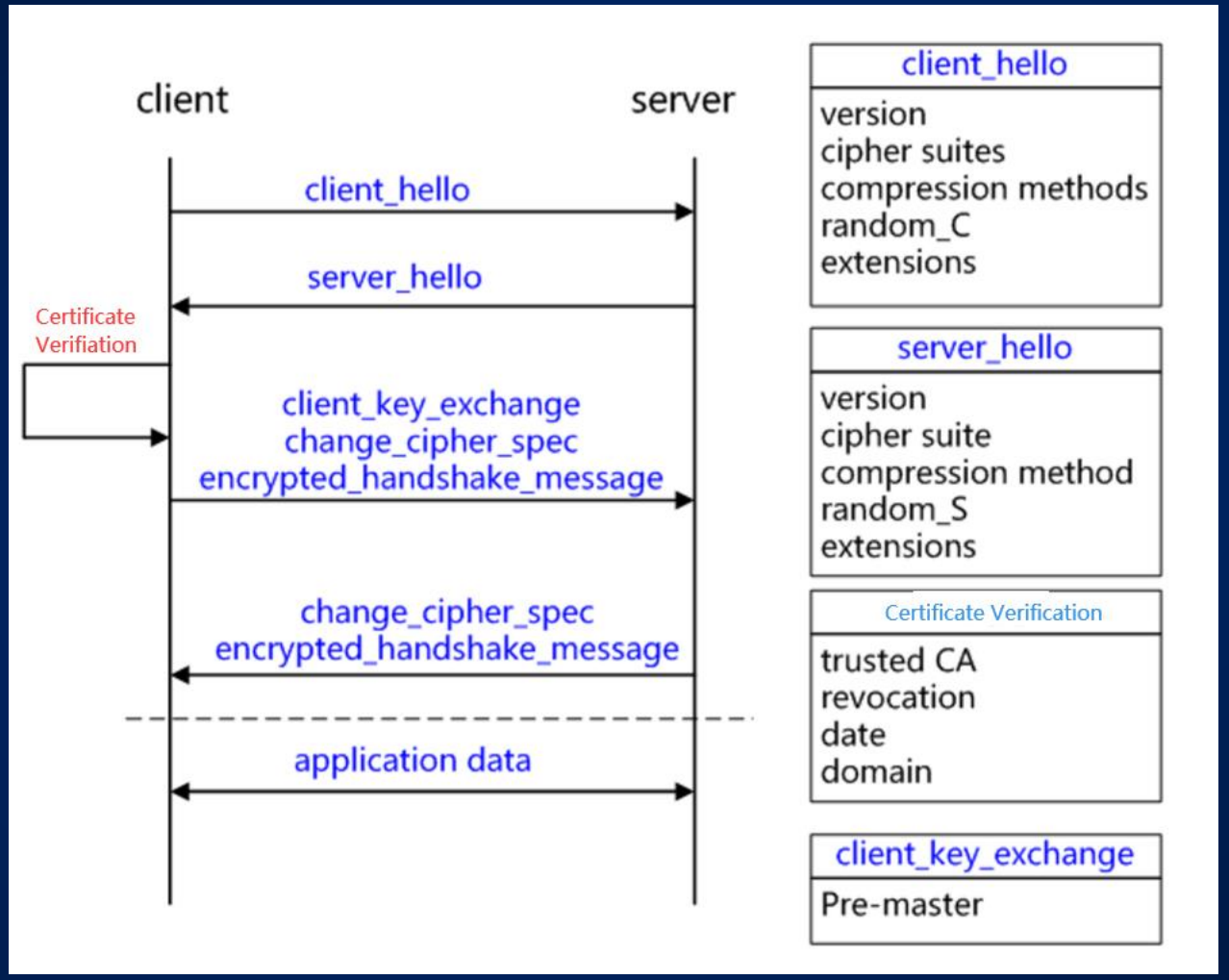
Config Item	Description
Verify Client	Disabled: ADC does not initiate a Certificate Request to the client, which means it does not verify the client's certificate. Optional: ADC may request the client to provide a certificate, but the client can choose not to provide it. If provided, the certificate will be verified. Force: ADC requires the client to provide a certificate. If the client does not provide it, the connection will not be allowed to continue.
CA Certificate	In order to verify client certificates, ADC needs to import the CA certificate chain that issued the client certificate. Only with these certificate chains, the client certificate can be verified. When there are multiple issuers for client certificates, multiple CA certificate chains need to be configured here.
CA Certificate Auto Advertisement/Cert-chain Advertisement	When this feature is enabled, along with the Certificate Request, ADC will announce all the CA certificates configured in the "CA certificate" to the client, and the client will provide their own certificate based on the CA certificates. Correspondingly, if this feature is disabled, it can be configured which certificates will be announced with the Certificate Request.
Min Key Length	Refers to the minimum length of the client's ECC/RSA certificate, and if it is less than this length, the verification will fail.
Max Verify depth	Refers to the maximum depth of the client certificate, which means the number of layers between the root certificate and the entity certificate. If it exceeds this limit, the verification will fail.
Verification rule	Different processing methods should be applied based on the failure result of the client certificate verification.

Verify URL Rule

- Apart from applying a unified authentication strategy for all clients, it is also possible to determine whether to perform certificate verification based on the URL accessed by the client. For example, client certificate verification is only required when user want to accessing important resources, while it is not required for less critical resources.
- Each rule is matched in order of priority, and once a rule is matched, no further rules are applied.



SSL Interactive Process



client_hello

- The client initiates a request and transmits the request information in plaintext, including version information, a list of candidate encryption suites, a list of candidate compression algorithms, a random number, and extension fields. The relevant information is as follows:
- The highest supported TSL protocol version, version, from low to high: SSLv2, SSLv3, TLSv1, TLSv1.1, and TLSv1.2. Versions below TLSv1 are currently not used.
- The client's supported cipher suites list, where each cipher suite corresponds to a combination of the four functions in the TLS principle: authentication algorithm (Au), key exchange algorithm (KeyExchange), symmetric encryption algorithm Enc, and message digest Mac.
- The supported compression methods list, used for subsequent compressed information transmission.
- The random number, random_C, used for subsequent key generation.
- The extensions field supports protocol and algorithm-related parameters and other auxiliary information. Common extensions include SNI.

server_hello+server_certificate+sever_hello_done

- server_hello: The server returns the negotiated information, including the selected protocol version, cipher suite, compression method, random number (random_S), etc., which will be used for subsequent key negotiation.
- server_certificates: Configure corresponding certificate chain on server side, used for identity authentication and key exchange.
- server_hello_done: The server notifies the client that the server_hello message has been sent.

Certificate Verification

- The client verifies the legitimacy of the certificate. If the verification passes, subsequent communication will proceed. Otherwise, the client will provide different prompts and operations depending on the error situation. The legitimacy verification includes the following:
    - Trusted certificate path: Verify the credibility of the certificate chain, as described earlier.
    - Revocation: Check if the certificate has been revoked. There are two ways to do this: offline certificate revocation lists (CRL) and online OCSP. Different clients may behave differently.
    - Expiry date: Check if the certificate is within its validity period.
    - Domain: Verify if the certificate domain matches the current access domain. The matching rules will be analyzed later.

client_key_exchange+change_cipher_spec

- After the client_key_exchange has been validated for legality, the client will compute a random number called Pre-master, and encrypts it by using the certificate public key, and then sent to the server.
- At this point, the client has obtained all the necessary information to compute the negotiated key for encryption: two random numbers called random_C and random_S, and the Pre-master that the client has calculated by itself. Then, the negotiated key can be computed.
- enc_key = Fuc(random_C, random_S, Pre-Master)
- change_cipher_spec: The client notifies the server that all subsequent communication will be encrypted using the negotiated key and encryption algorithm agreed upon during the key exchange process.

encrypted_handshake_message

- encrypted_handshake_message: A piece of data that is generated by hashing all the communication parameters discussed before together with other relevant information. This data is then encrypted by using the negotiation key “session secret” and negotiation algorithm, and sent to the server for data and handshake verification.

change_cipher_spec+encrypted_handshake_message

- The server decrypts the encrypted Pre-master data by using its private key, and computes the negotiated key by using the two plaintext random numbers exchanged earlier which are random_C and random_S. The formula for computing the key is: enc_key = Fuc(random_C, random_S, Pre-Master).
- The server then computes the hash value of all the received information. Then, decrypte the encrypted_handshake_message sent by the client to verify the correctness of the data and key.
- change_cipher_spec, after verification passed, the server sends a change_cipher_spec message to the client to notify that all subsequent communication will be encrypted by using the negotiated key and negotiation algorithm.
- encrypted_handshake_message, the server also generates a piece of data by combining all current communication parameter information and uses the negotiated session secret and algorithm to encrypt and send it to the client.

Handshake completed + encrypted communication

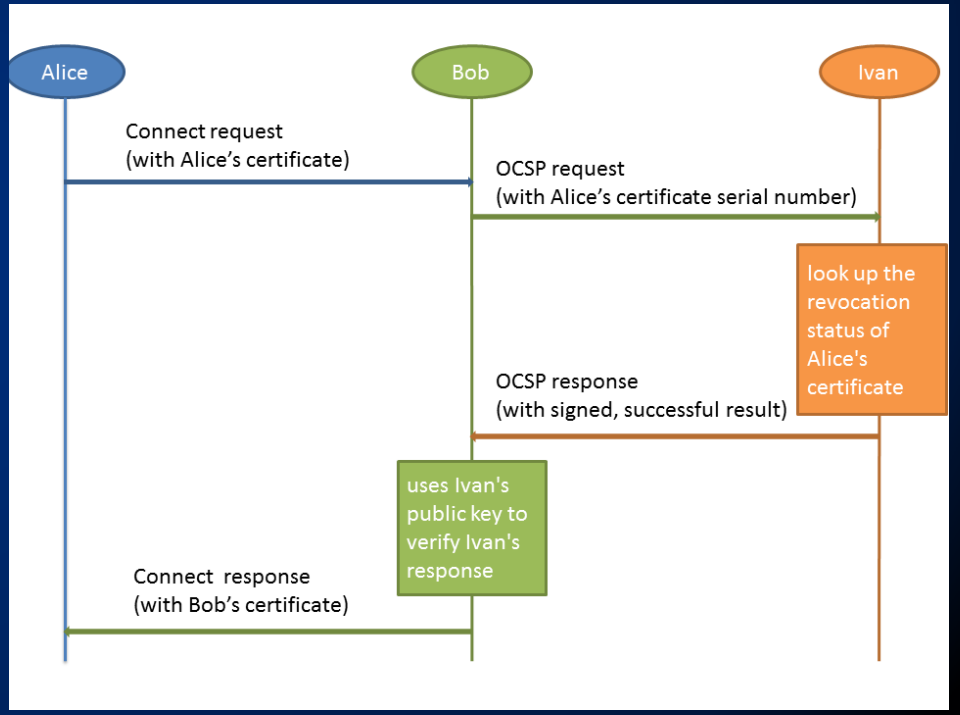
- Handshake completed
  - The client calculates the hash value of all received information, decrypts the encrypted_handshake_message by using the negotiation key, and verifies the data and key which received from the server. If the verification passes, the handshake is completed.
- Encrypted communication
  - Start to use the negotiation key and algorithm for encrypted communication.

Notes for Interaction

- The server can also request to authenticate the client, that is, two-way authentication. The client_certificate_request message can be sent in process 2. In process 4, the client firstly sends the client_certificate and certificate_verify_message information. The certificate verification method is basically the same. The certificate_verify_message is a piece of data which obtained based on the communication information which have already been negotiated , and encrypted with the client's private key. The server can decrypt and verify it by using the corresponding public key.
- Depending on the different key exchange algorithms that have been used, such as ECC, there may be slight differences in the negotiation details, but the overall process is similar.
- The purpose of server key exchange is when the server certificate does not carry enough information, send it to the client to calculate the pre-master, such as DH based certificate, where the public key is not included in the certificate and needs to be sent separately.
- The alert message is used to indicate the state changes or error information during the handshake or communication process. The trigger conditions for generate general alarm message are when the connection is closed, when illegal information is received, when information decryption fails, or when the user cancels the operation, etc. After receiving the alert message, the communication will be disconnected or let the receiving side decides whether to disconnect the connection.

OCSP Principle

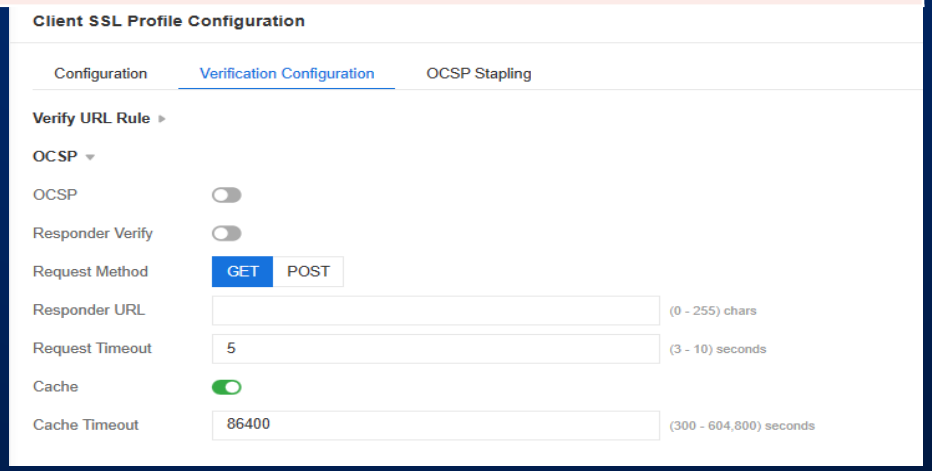
- OCSP stands for Online Certificate Status Protocol. It is a way of validating whether a certificate is legitimate or not.
- Another way to do this is through CRL, which stands for Certificate Revocation List. This involves the verifier having a local list of revoked certificates that they use this list to validate the client's certificate.
- In the scenario shown in the diagram, when Alice connects to Bob and sends her certificate (which was issued by Ivan), Bob is able to verify Alice's certificate but cannot determine whether it has been revoked. He sends an OCSP request to Ivan (who is the OCSP responder). Ivan checks for any records of the certificate being revoked, and if none are found, he sends a response to Bob indicating that the certificate is valid. SSL negotiation can then proceed normally.

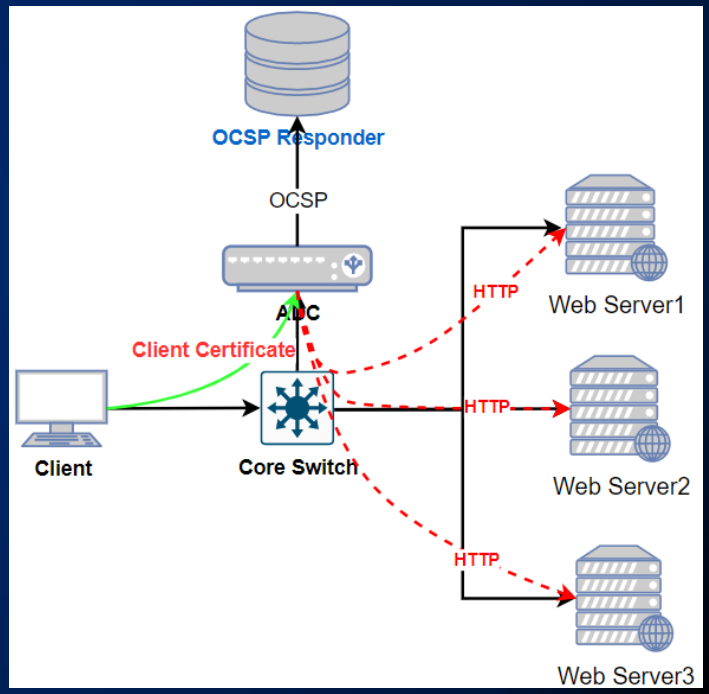


OCSP Client

At this moment, the ADC acts as an OCSP client, working with the OCSP server to verify client certificates.

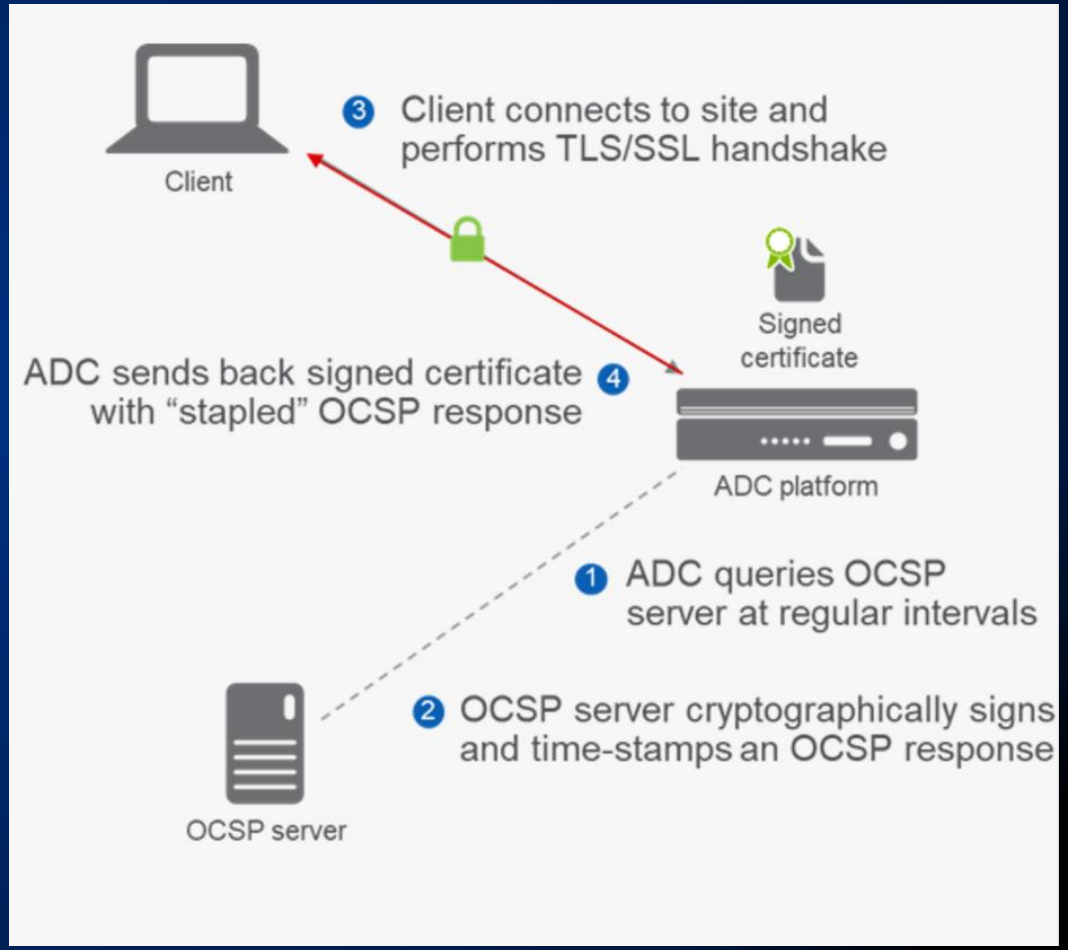
Config Item	Description
Responder Verify	Once enabled, the ADC will verify the signature of the OCSP response
Request Method	OCSP is based on the HTTP protocol and can use either the GET or POST request method. When the OCSP request is small, the GET method can be used, but for larger requests, the POST method should be used.
Responder URL	This refers to the URL requested by OCSP service.
Request Timeout	The timeout set for ADC establishes a connection with the OCSP server, and if the connection is not successfully established within this time, it will be disconnected.
Cache	Since the interaction between the ADC and the OCSP server also incurs additional costs, enabling "caching" allows the ADC to cache the last validation result from the OCSP server locally. When the same client requests again, the ADC can use the cached validation result from the local cache.
Cache Timeout	The ADC saves the timeout for the OCSP; once it exceeds that time, the system will clear the status of the certificate in the cache.



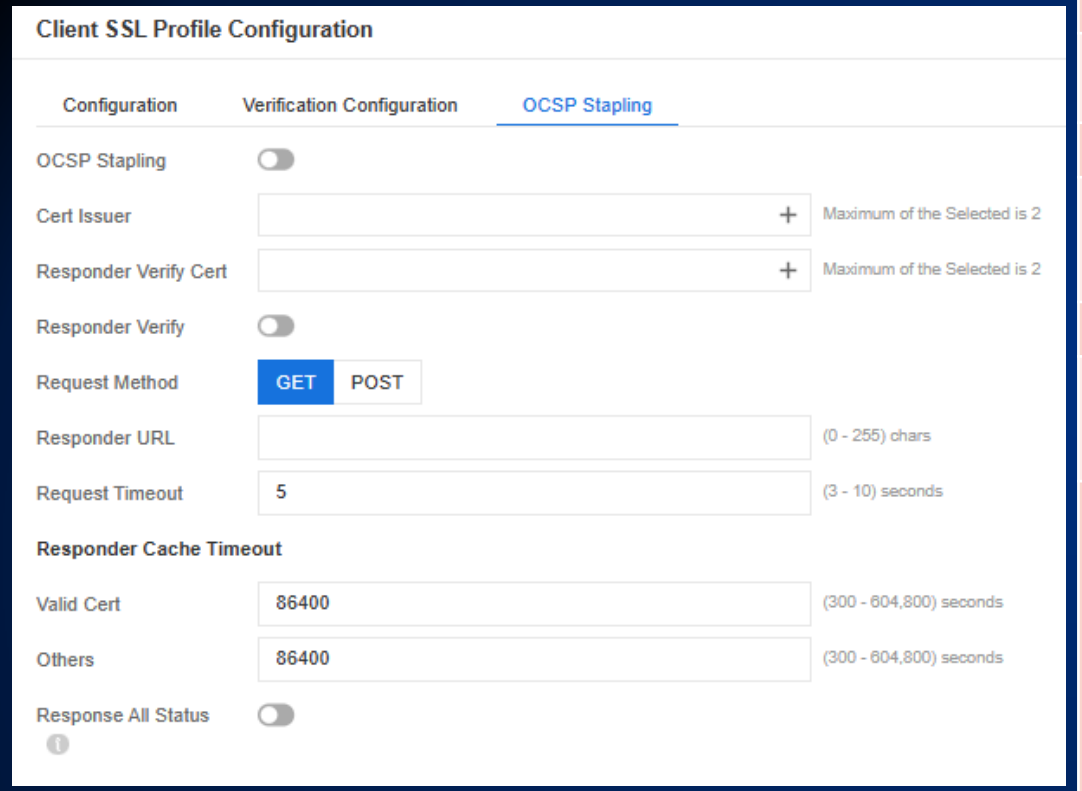


OCSP Stapling

- OCSP Stapling, from its name, appears to be similar to OCSP, but it is used in a different context. The purpose of OCSP Stapling is to speed up the process of client verification of server certificates.
- Without OCSP Stapling, the client needs to initiate a request to the OCSP responder to verify whether the server's certificate has been revoked. This verification process slows down the SSL connection establishment between the client and the server.
- With OCSP Stapling, the server (such as the ADC) queries the OCSP responder for the status of its certificate and attaches the result to the certificate information. When the client receives this information, it can locally verify whether the certificate has been revoked.
- In practice, OCSP Stapling can significantly improve the efficiency of establishing HTTPS connections on Apple devices (since Apple devices enable OCSP by default). Using OCSP Stapling eliminates the need for interaction between the terminal and the OCSP responder.

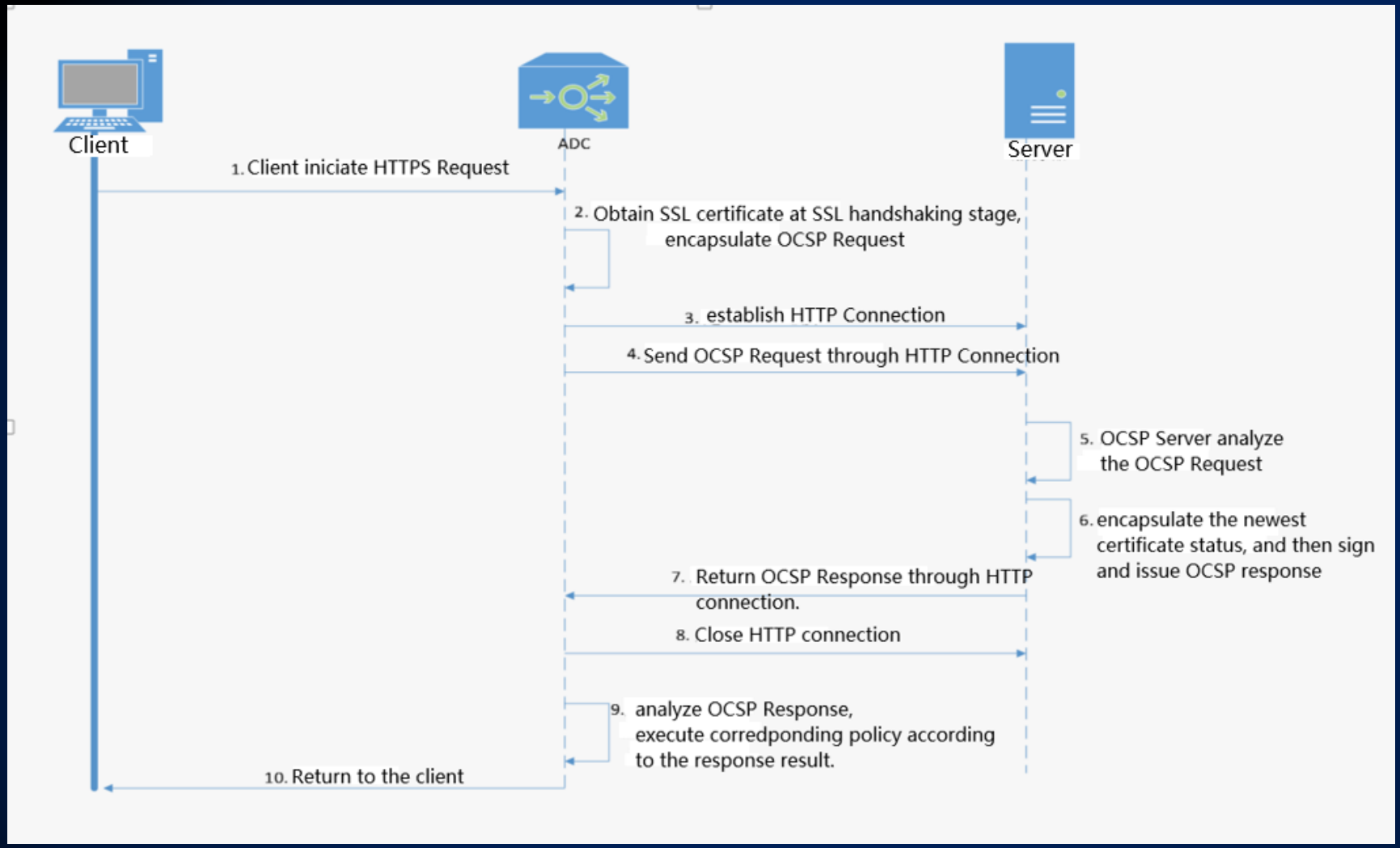


OCSP Stapling Configuration



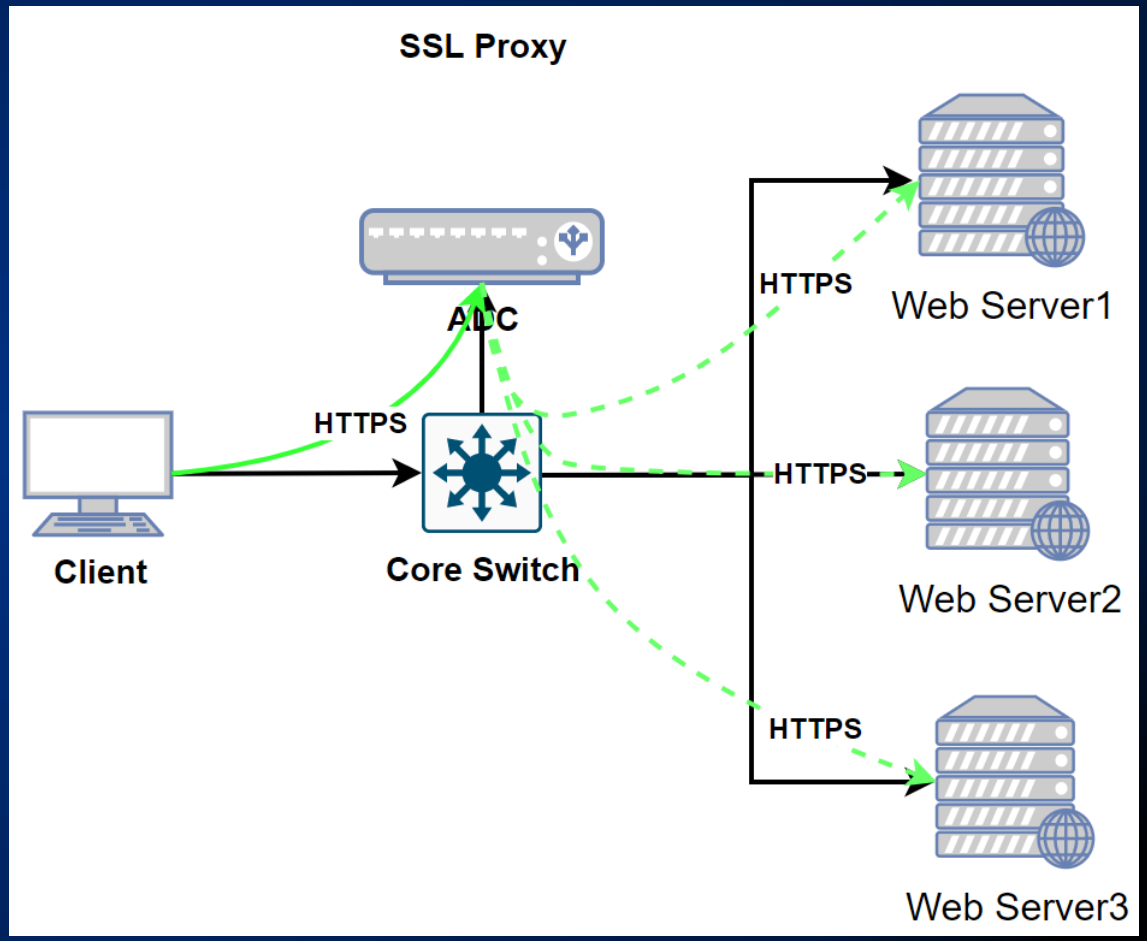
Config Item	Description
Cert Issuer	Referring to the issuer certificate of a virtual server; if the certificate chain of the client SSL profile includes the issuer, this item does not need to be configured
Responder Verify Cert	Used to verify the certificate of the OCSP responder's response.
Responder Verify	If enabled, it will verify the response of the OCSP responder.
Request Method	OCSP is based on the HTTP protocol and can use either the GET or POST request method. When the OCSP request is small, the GET method can be used, but for larger requests, the POST method should be used.
Responder URL	This refers to the URL requested by OCSP service.
Request Timeout	The timeout set for ADC establishes a connection with the OCSP server, and if the connection is not successfully established within this time, it will be disconnected.
Cache	Since the interaction between the ADC and the OCSP server also incurs additional costs, enabling "caching" allows the ADC to cache the last validation result from the OCSP server locally. When the same client requests again, the ADC can use the cached validation result from the local cache. • The "validity period of the certificate" refers to the time retained for when the OCSP response is "good". • In other cases, it refers to the retention time of the result when it is **not** "good"
Responder Cache Timeout	When enabled, regardless of the state of the OCSP verification result, it is attached to the certificate information and provided to the client; otherwise, only the "good" result is provided to the client.

OCSP Interactive Process



Server SSL Profile

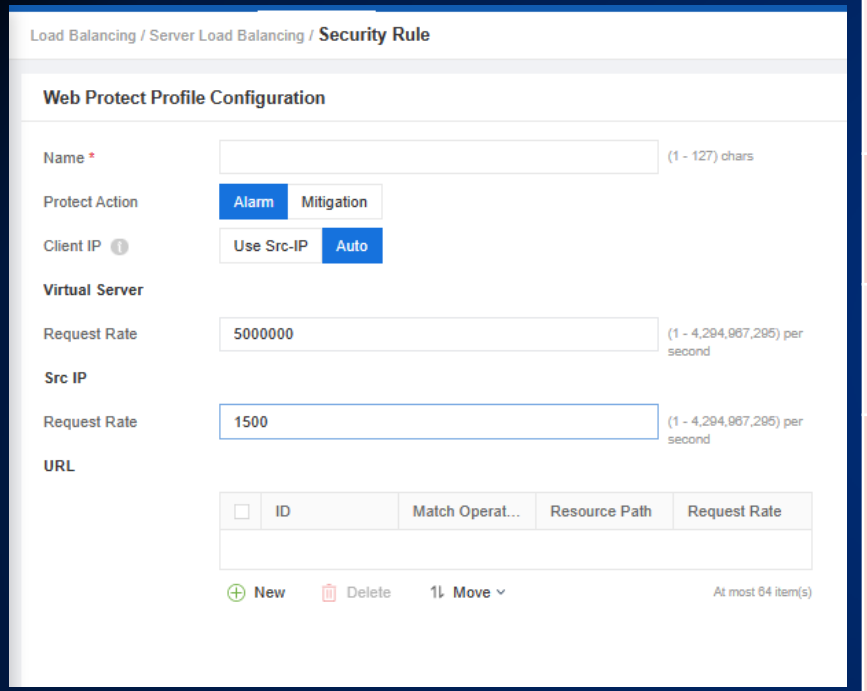
- The server SSL profile is commonly used in situations where ADC is used as an SSL proxy, and both the client-to-ADC and ADC-to-backend server connections are HTTPS. In the connection between the client and ADC, the ADC acts as an SSL server, while in the connection between the ADC and RS, the ADC acts as an SSL client.
- When the ADC acts as a client to establish an SSL connection with RS, a separate key suite is required, and the server SSL profile is used to configure content in this part.



Web Protect Profile

- The system provides a Web protection profile to defend against HTTP flood attacks. For example, an attacker launches a large number of HTTP messages to consume server resources, making the server unable to respond to normal client requests.
- After configuring the Web protection profile, when users create HTTP or HTTPS virtual servers, they can directly apply this profile to protect the internal network from malicious attacks and ensure the normal operation of the internal network and system.
- The device currently employs three dimensions for protection: based on source IP, URL resource path, and virtual server.
- The device determines whether to take protective actions by judging whether the client request rate exceeds the threshold. If the request rate exceeds the threshold rate, the device provides three protective actions:
- Alert: The system allows the request to pass and generates an alert log.
- Redirection: The device returns a redirection message to the client to verify if the client is a legitimate client.
  - A legitimate client can complete the redirection process and send the request to the device again, at which point the device adds the validated client to the trusted list.
  - An attacking client, such as an automated script, cannot complete the redirection process. If an attacking client makes another request while the request rate exceeds the threshold, the system returns a redirection message and records the request log exceeding the threshold. The device also does not forward the attacking request message to the server, thereby providing protection.
- Block: The device directly blocks client requests that exceed the threshold request rate and adds the client to the blacklist.

Web Protect Profile



Config Item	Description
Protect Action	If the client-side attack requests exceed the configured rate threshold, three corresponding actions will be taken: alarm, redirection, and blocking. The mitigation methods include redirection and blocking.
Client IP	Specify the source of the client IP checked when based on the source IP detection: • Use source IP: The device will use the source IP from the request message as the detection target. • Auto: The device obtains the client IP from the request message in the following order of priority: X-Forwarded-For -> X-Real-IP -> source IP carried in the message.
Virtual Server	When the access rate to a virtual server reaches a specified threshold, the system regards such requests as attacking behaviors and takes specified protective actions against them. The request rate refers to the access rate threshold for the specified virtual server.
Src IP	When the request rate initiated by a certain source IP and reaches the specified threshold, the system regards such requests as attacking behaviors and takes specified protective actions against them. The request rate refers to the access rate threshold initiated by the specified source IP.
URL	When the target URL requested by the client matches the specified resource path and the request rate reaches the specified threshold, the system regards such requests as attacking behaviors and takes specified protective actions against them. The configuration details are as follows: • ID (optional): The ID is based on the URL detection rule. If not configured, the system will automatically assign an ID for it within the range of 1 to 64. • Match operation: Specifies the match operation for the URL resource path, which can be "contain", "equal", or "prefix match". • Resource path: Specifies the URL path to be detected, and the system will perform web protection on the requests that match this path. • Request rate: Specifies the threshold for the access rate of the URL. • Position: The URL rule can be placed at the front or the end of the list, before or after a certain ID, as required. The device matches traffic based on the display order of the URL rules, not based on the ID.

Hillstone NETWORKS

Integrative Cybersecurity

Visionary. AI-powered. Accessible.

+1 408 508 6750
inquiry@hillstonenet.com
5201 Great America Pkwy, #420
Santa Clara, CA 95054
www.hillstonenet.com