Chapter 2
Load Balancing Algorithm

Training Team

HCSA-ADC Official Training

Integrative Cybersecurity

Visionary. AI-powered. Accessible.

Agenda

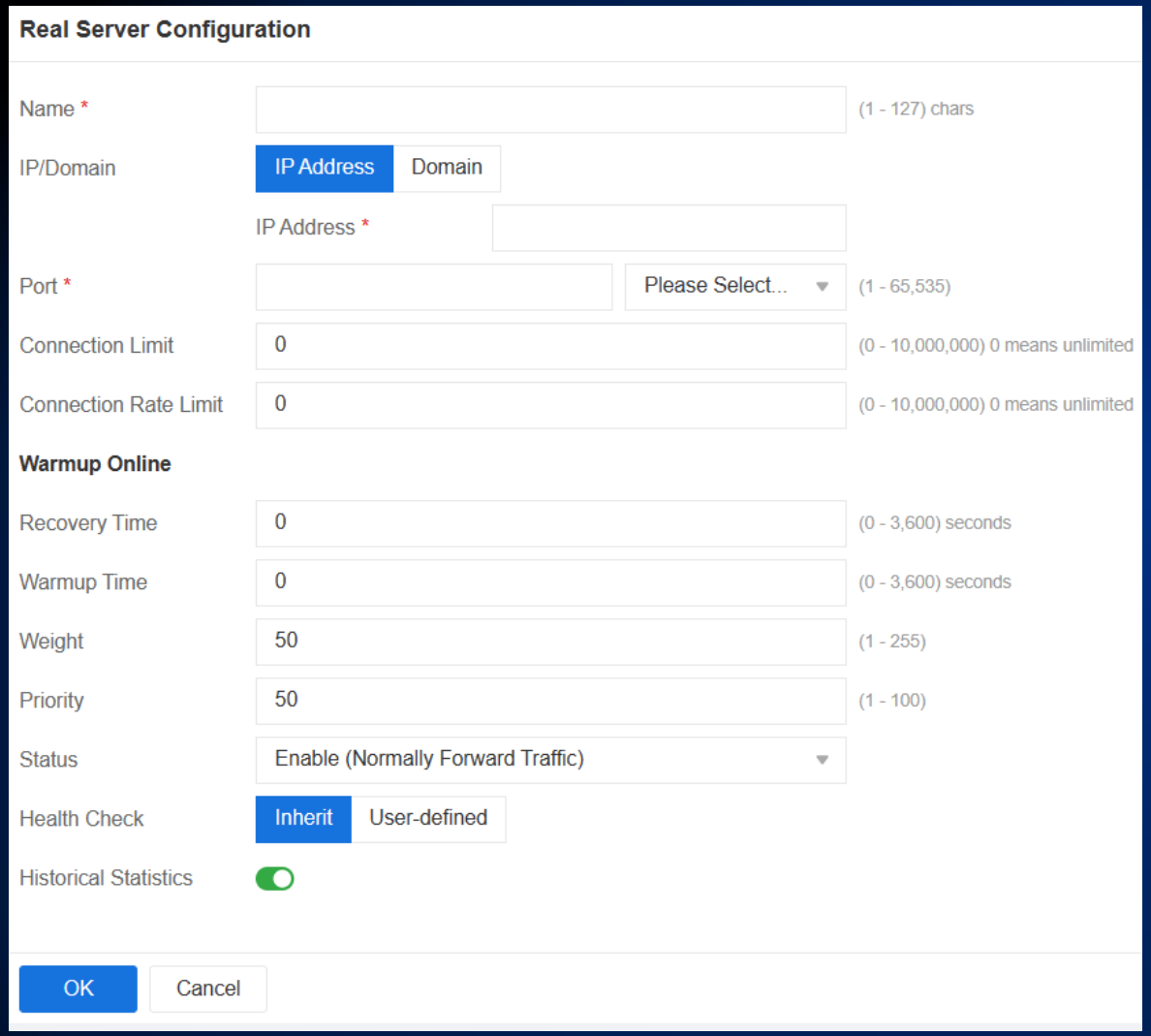
- Server Pool and Real Server
- Load Balancing Algorithm
- Session Persistence

1 Server Pool and Real Server

Real Server

- Real Server (RS) is used to describe a service on a virtual or physical server in a back-end server cluster. For example, a web service or database service on a VM or physical server.
- The most important configuration for a real service is usually be the IP and port; when the port is “Any”, it represents directly use request’s destination port (that is VS port) to be the RS port, normally with configure “Any” for port, “Any” for type on VS, or configured multiple IP port pairs on VS. Usually used in the situation when publish multiple services from one virtual server, for example, IPSec VPN is related to protocols like ESP, AH, UDP, IKE, etc. For another example, a group of mail server provides IMAP, POP3, SMTP services, but they only want to establish one virtual server, in this situation, the port can be configured as “Any”; In addition, if a virtual server wish to publish HTTP service through TCP 80, and HTTPS service through TCP 443. Then, you can configure two IP/Port pairs in a TCP type VS, however, the RS should configure the port to be “Any”, which represent that the RS will use VS’s destination port to be its port.

Real Server



Real Server

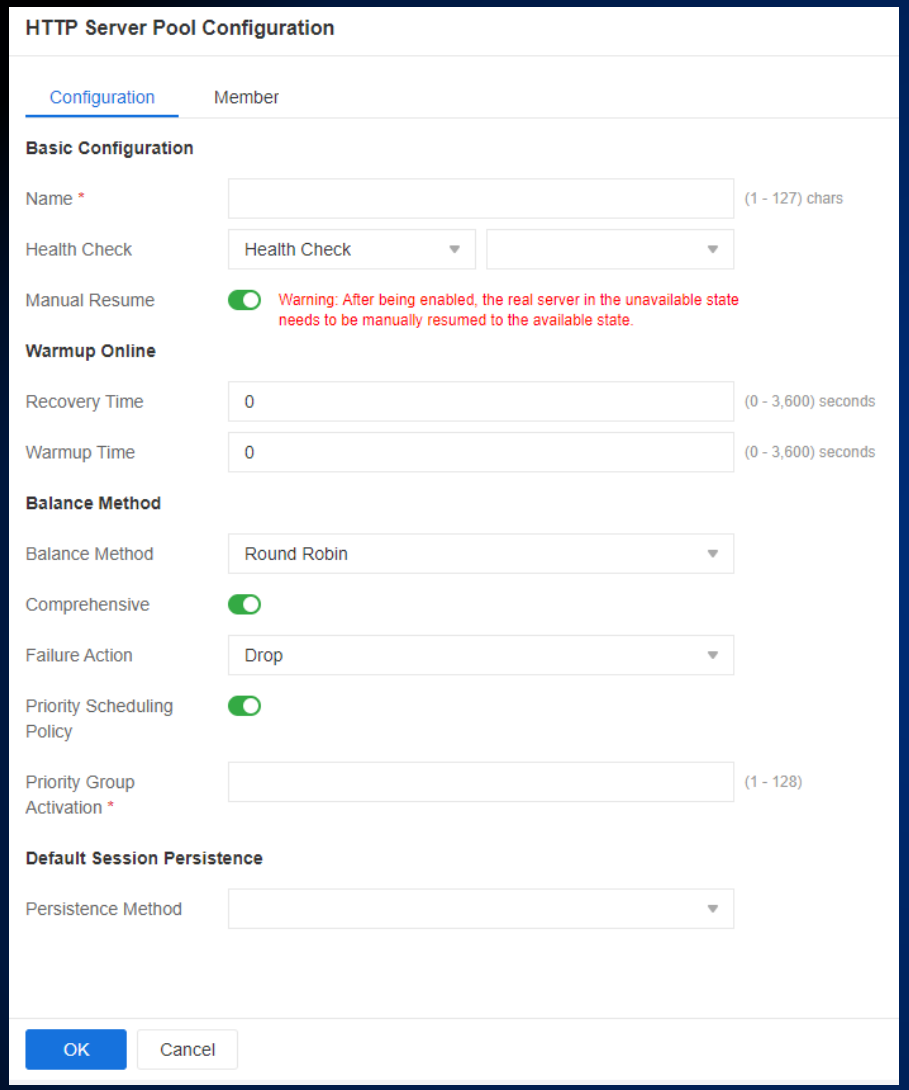
Config Item	Description
IP Address /Domain(*)	In most of scenarios, RS's IP address is fixed, so that you just need to configure the IP address; in some other scenarios, the RS is a domain name, so that it have to be resolved and dispatched by ADC. When you configure domain name for RS, it is possible that there are multiple IP address after resolve this domain name, you can configure to use the first IP it resolved, or use all the IP address; when you configure to use all IP address, it will generate multiple sub real servers, and be dispatched with other RS together.
Port(*)	When configure "Any" for RS port, it represents to use the destination port which customer request (which is some port of VS).
Connection Limit	When the connections exceed the connection limit, the new connection will not be dispatched to this RS.
Connection Rate Limit	When the connections exceed the connection rate limit, the new connection will not be dispatched to this RS.
Recovery Time	After RS is break down, and restore the status to up after health check, the system have to wait RS to restore the business and service, so that ADC will not dispatch any business to this RS during the recovery time.
Warmup Time	Warmup time represents for how long the RS have to take from start the service to provide the service normally. ADC will calculate how much business should dispatch to this RS according to the point in warmup time. The recovery time and warmup time mainly used for server online warmup.
Weight(*)	The weight of RS must be used in conjunction with the load balancing algorithm in server pool, like weighted round robin, weighted hash.
Priority	The priority of RS need to be used in conjunction with least active RS in server pool.
Status	Enable: forward data normally; Disable: drop all traffic; Waiting down: clear persistence, or wait for persistence to be aged out.
Health Check(*)	RS can inherit the health check from server pool, or it can also configure the health check itself.
Historical Statistics	Whether to enable statistics for this RS

Server Pool

- Server pool is a group of real servers which provide the same service. ADC requires that all the RS in the same server pool should provide the same service, for example, provide a web service for a website, or provide service for the same database; but ADC does not requires all the RS in the server pool have the same capacity, neither ask for all the RS use the same port, the difference in capacity can be described by the different weighted value.

- In simple, server pool is a back-end server group; However, server pool can not independently provide service to outside, it have to be associated by VS in order to provide service externally.

Server Pool



Server Pool

Config Item	Description
Health Check	Configure health check or health check group on server pool; this health check group is usually used by members in the pool (RS); of course, RS can also use their own specified health check.
Recovery Time	After RS is break down, and restore the status to up after health check, the system have to wait RS to restore the business and service, so that ADC will not dispatch any business to this RS during the recovery time. When you configure recovery time on both server pool and real server, the recovery time which configured on real server have the higher priority.
Warmup Time	Warmup time represents for how long the RS have to take from start the service to provide the service normally. ADC will calculate how much business should dispatch to this RS according to the point in warmup time. The recovery time and warmup time mainly used for server online warmup. When you configure warmup time on both server pool and real server, the warmup time which configured on real server have the higher priority.
Manual Resume	After the member's status in pool become unavailable, the real server in the unavailable state needs to be manually resumed to the available state by administrator.
Balance Method	Support for algorithms like Round Robin, Hash, least Connection, Fastest Response, Priority, Dynamic Ratio, and etc. Comprehensive: this function only take effects for round robin and weighted round robin; take round robin as an example, when new connection/request to match a content exchange rule or session persistence table, it is possible that it does not need to be dispatched by balancing algorithm, so that it neither need to join the load balancing calculation, as a result, this part of traffic is not dispatched according to round robin method, which will result in the actual traffic is unbalanced. Through "comprehensive" function, it will involve the result of content exchange and session persistence table into the round robin, and weighted round robin calculation, which will ensure the traffic balance in a better way.
Priority scheduling policy	Turn on the switch to enable priority scheduling policy and specify the minimum number of active real servers in the Priority Group Activation field. This number indicates the minimum number of real servers selected for server pool balancing scheduling in the current system. Valid values: 1 to 64. If there are 100 real servers in the system and the minimum number of active real servers is 20, the available real server with the highest priority is preferentially scheduled. If the number of real servers to be scheduled is less than 20, the available real server with the second highest priority is scheduled. The preceding process is repeated until the number of real servers to be scheduled is no less than 20 or all available real servers are scheduled. If a real server involved in scheduling is unavailable and thereby causes that the actual number of real servers involved in scheduling is less than 20, the device continues to select a real server with the highest priority from those not involved in scheduling. If a real server with a higher priority becomes available, another real server with a lower priority that is already scheduled may become unschedulable again.
Session Persistence	To maintain the continuity and consistency of a session, the device supports the session persistence function. By configuring a session persistence method, the device will always distribute client requests carrying the specified persistence feature to the same server as that of the client accessed for the first time instead of multiple servers, thereby ensuring session persistence.

2 Load Balancing Algorithm

Load Balancing Algorithm

- Balancing algorithm is the most basic and core module for the server load balancing scenario; the choice of different algorithms determine the different balancing effects.
- The load balancing algorithm should be configured on server pool, the server pool will dispatch traffic to its members based on the balancing algorithm.
- Different type of server pool support for different balancing algorithm group; for sever pool with the same type, it also support for multiple balancing algorithms, which can be applied to different scenarios.
- For some specific services, the choice of balancing algorithm should based on the service characteristics; there is no best or worst balancing algorithm exist, users can only judge a fittest algorithm for a particular scenarios.

Load Balancing Algorithm

Load Balancing Algorithm	Type supported	Description
Least Bandwidth(*)	IP	The device will record the sum of upstream and downstream bandwidth on each real server or port. When a new connection is requested, the device will distribute it to the real server with the least sum.
IP-Port Hash/Weighted IP-Port Hash	All types	First, the IP address and port number of the client will be hashed. Then, the client will be allocated to a real server according to the hash value. In the actual scenario, since we normally use session persistence to let the traffic from one client dispatch to the same RS, so that this balancing algorithm is rarely used.
Dynamic Ratio	All types	According to the different processing capabilities of real servers, the device calculates different dynamic weights. The device will distribute client requests in proportion to dynamic weights of real servers. The real server with higher dynamic weight will receive a higher proportion of requests. If the dynamic weight is 0, the real server will not be distributed with requests. In general, the dynamic weight value is calculated by the SNMP health check; in other cases, the value of 1 means that the real server can provide services, while 0 means it is unavailable.
Priority	All types	Requests will be distributed to a real server with the highest priority. The smaller the priority value is, the higher the priority of the real server will be. If there are multiple real servers with the same highest priority, resources will be allocated by round robin among these real servers.
Random	All types	Client request will be routed to available servers on a random basis.

Algorithmic Dispatching

- In the actual scenario, the dispatching result of load balance will be affected by session persistence, specially in the scenario when the number of source IP is small, and still use the source IP session or cookie session persistence, it have to persist the client to one specific RS, which will cause dispatching uneven from ADC's perspective. In the real environment, we will assume that the client requesting is discrete, and the balancing effect will be better.
- When using the round robin/weighted round robin, enable "comprehensive" will get a better balancing effect.

Option	Related Item	Description
Comprehensive	Round Robin/Weighted Round Robin	Turn on the switch to enable the Comprehensive function. With this function enabled, if the specified algorithm is Round Robin or Weighted Round Robin, system will perform load balancing based on the scheduling results of real servers by scripts, L7 Content Switching, L4 Content Switching, Session Persistence and other balance methods. System will determine the performance of each real server according to its current capacity. Then, after receiving new client requests, the device will first distribute the requests to the real servers with better performance in turn.
Priority Scheduling Policy	All types of Load Balancing Algorithm	Turn on the switch to enable priority scheduling policy and specify the minimum number of active real servers in the Priority Group Activation field. This number indicates the minimum number of real servers selected for server pool balancing scheduling in the current system. Valid values: 1 to 64. If there are 100 real servers in the system and the minimum number of active real servers is 20, the available real server with the highest priority is preferentially scheduled. If the number of real servers to be scheduled is less than 20, the available real server with the second highest priority is scheduled. The preceding process is repeated until the number of real servers to be scheduled is no less than 20 or all available real servers are scheduled. If a real server involved in scheduling is unavailable and thereby causes that the actual number of real servers involved in scheduling is less than 20, the device continues to select a real server with the highest priority from those not involved in scheduling. If a real server with a higher priority becomes available, another real server with a lower priority that is already scheduled may become unschedulable again.

Session Persistence

Round Robin

	RS1	RS2	RS3
IP1	1		
IP2		1	
IP3			1
IP1	1		
IP1		1	
IP1			1
IP4	1		
IP5		1	
IP6			1

Round Robin after enable source IP session persistence

	RS1	RS2	RS3
IP1	1		
IP2		1	
IP3			1
IP1	1		
IP1	1		
IP1	1		
IP4		1	
IP5			1
IP6	1		

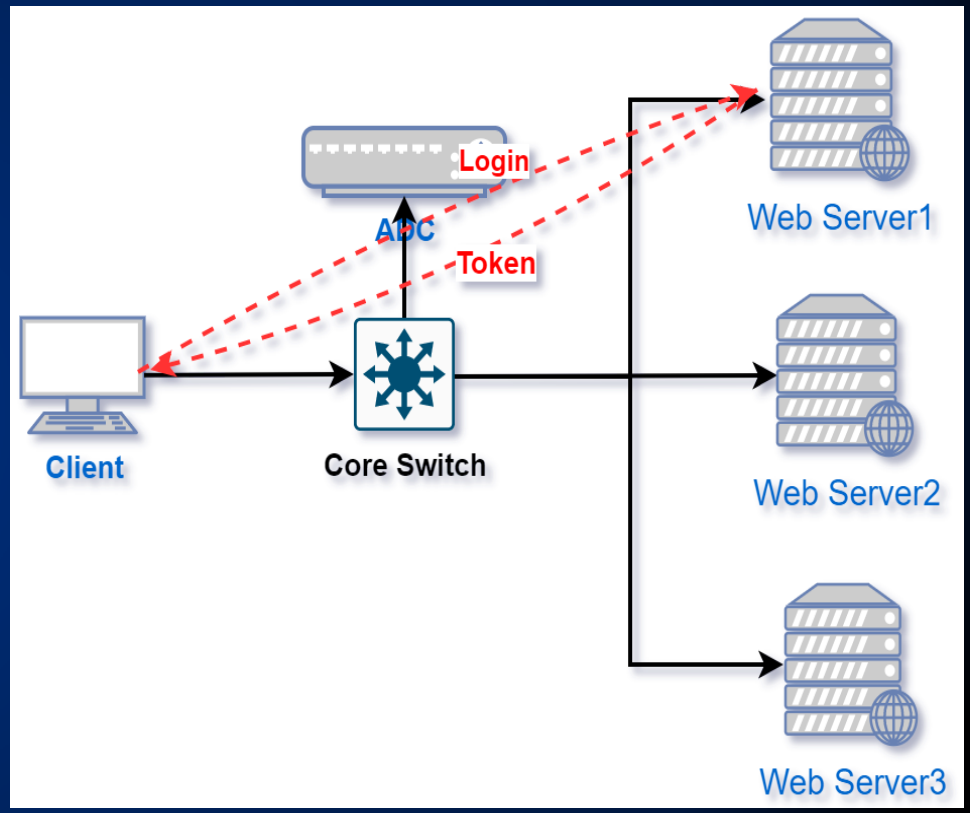
Round Robin after enable source IP session persistence and comprehensive

	RS1	RS2	RS3
IP1	1		
IP2		1	
IP3			1
IP1	1		
IP1	1		
IP1	1		
IP4		1	
IP5			1
IP6		1	

3 Session Persistence

Session Persistence

- Common Scenario: the graph shows the most common scenario for session persistence. After the client access to the service which ADC published, ADC will dispatch the first request to a server, and it will do authentication for the client, after the authentication passed, server will confer a token for the client, if this authorization is not saved on a dedicated authorization server, it just saved on the Web Server1, if the request from client not continually distributed to Web Server1 before the Token expired (for instance, dispatch to Web Server2), it might trigger the authentication again. So that we need to use session persistence to let the device always distribute client requests to the same server as that of the client accessed for the first time instead of multiple servers.
- Other: For example, if a mobile client switch 4G/5G to WIFI, the IP address will be changed, then we can also use session persistence to ensure the authentication information of that user retained, so that they do not have to log in again.

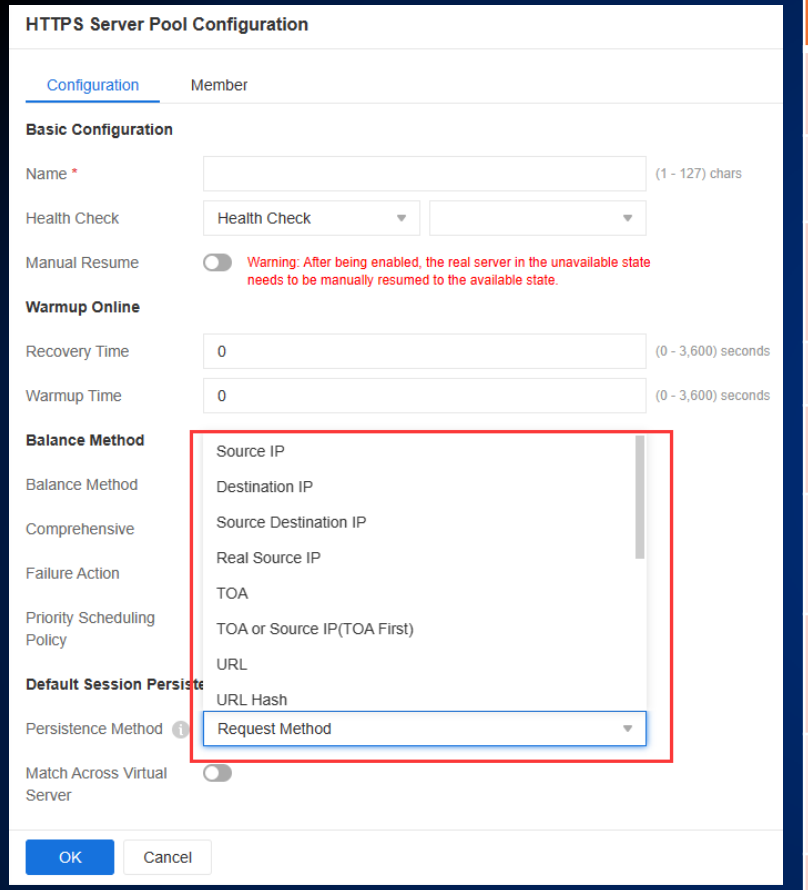


Relationship Between Load Balancing Algorithm and Session Persistence

- Session persistence have the higher priority than balancing algorithm, which means if a request match the session persistence, then the balanced algorithm is no longer carried out. Let's take source IP session persistence as an example, if a client initiate a request and been distribute to RS1, ADC will create a entry which use this source IP to be the Key, and RS1 to be the Value. Then, if the client access ADC again and hit this entry, ADC will distribute the request to RS1.
- Normally, session persistence is based on a certain VS's certain pool; which means it use VS+Pool as part of the default composite key that holds entries. When a pool is called by multiple VS, you can enable the "match across virtual server function". With this function enabled, the session persistence table will be shared with other virtual servers.
- Same as Load Balancing Algorithm, session persistence can be also configured at server pool.

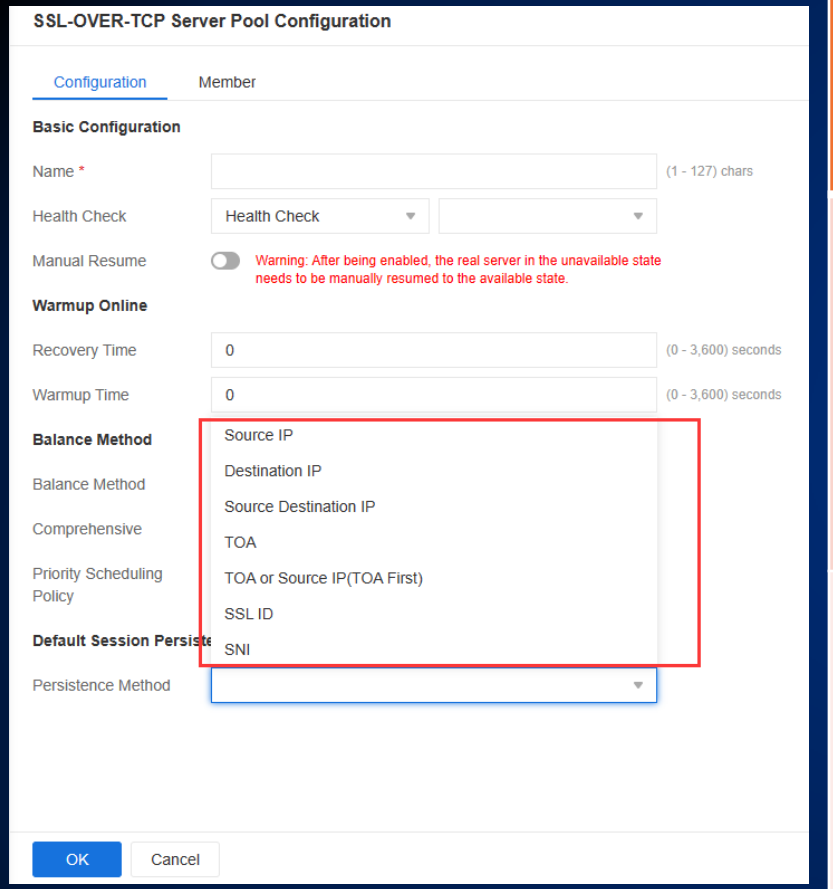
```
SG-6000(CR)# show slb persistence
SLB total persist item number: 76
================================================================================================================
===============
Virtual-server      Server-pool         Real-server         Method         Life      Content

----------------------------------------------------------------------------------------------------------------
---------------
https-fr-offload-p~ https-fr-proxy      10.100.7.20-443     src-ip         259       10.230.1.84
https-fr-offload-p~ https-fr-proxy      10.100.7.20-443     src-ip         294       10.230.1.25
https-fr-offload-p~ https-fr-proxy      10.100.7.20-443     src-ip         280       10.230.0.212
https-fr-offload-p~ https-fr-proxy      10.100.7.20-443     src-ip         287       10.230.0.189
https-fr-offload-p~ https-fr-proxy      10.100.7.20-443     src-ip         230       10.131.0.96
https-fr-offload-p~ https-fr-proxy      10.100.7.20-443     src-ip         172       10.230.1.80
https-fr-offload-p~ https-fr-proxy      10.100.7.20-443     src-ip         232       10.130.2.125
https-fr-offload-p~ https-fr-proxy      10.100.7.20-443     src-ip         83        10.230.0.132
https-fr-offload-p~ https-fr-proxy      10.100.7.20-443     src-ip         268       10.230.0.227
https-fr-offload-p~ https-fr-proxy      10.100.7.20-443     src-ip         153       10.131.0.97
```

Session Persistence Method-HTTP



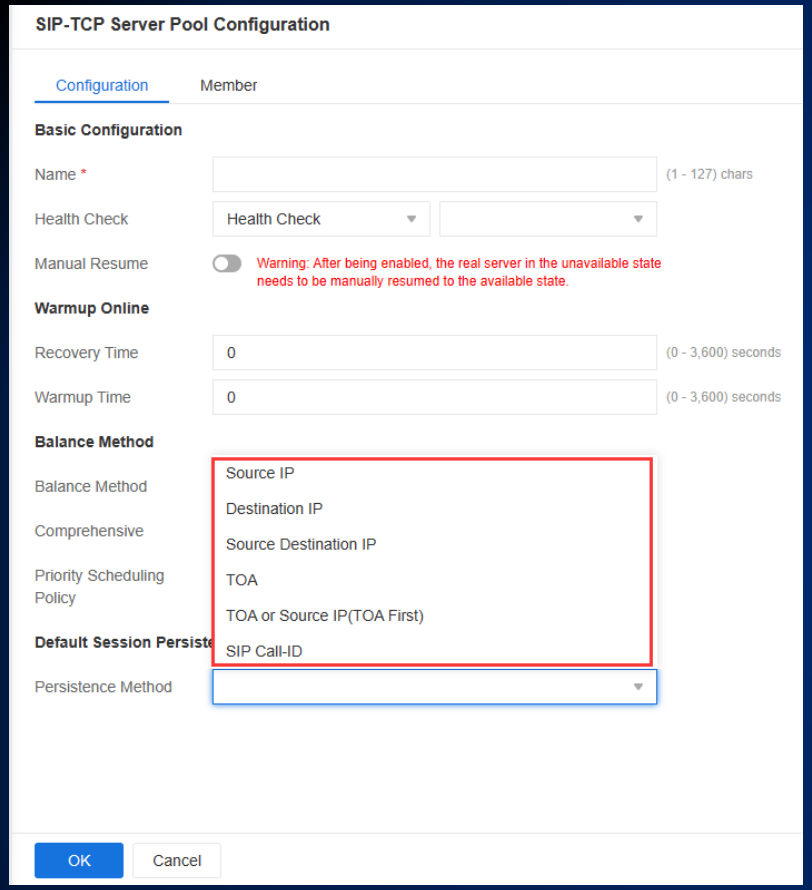
Session Persistence	Type Supported	Description
Source IP	All types	Sends client requests (including all subsequent connection requests) from the same source IP address to the same real server.
Destination IP	All types	Sends client requests (including all subsequent connection requests) for accessing the same destination IP address (VS address)to the same real server.
TOA	All types	TOA (TCP Option Address), Obtains the TOA from a TCP request of a client. The request carrying the TOA value will be distributed to the same server as that of the client accessed for the first time.
TOA or IP	All types	If there is TOA exist, use the IP address in TOA; if there is no TOA exist, use the source IP.
Cookie Hash	HTTP/HTTPS	Obtains a string from the HTTP cookie of a client request for hashing. The request carrying the string will be distributed to the same server as that of the client accessed for the first time.
Insert Cookie	HTTP/HTTPS	For an HTTP request passing through the device, the device will insert a cookie into the response. The request carrying the cookie will be distributed to the same server as that of the client accessed for the first time.
URL/URL Hash	HTTP/HTTPS	Obtains a string from the HTTP URL path or URL parameter value in a client request for hashing. The request carrying the string will be distributed to the same server as that of the client accessed for the first time.
Header/Header Hash	HTTP/HTTPS	Obtains a string from the HTTP header of a client request for hashing. The request carrying the string will be distributed to the same server as that of the client accessed for the first time.
Request Method	HTTP/HTTPS	Obtains the request method from a client request. The request carrying the method will be distributed to the same server as that of the client accessed for the first time.

Session Persistence Method-SSL-OVER-TCP Service



Session Persistence	Type Supported	Description
SSL ID	HTTPS/SSL-OVER-TCP	Obtains an SSL ID from an HTTPS client request. After the first SSL negotiation, it will generated SSL Session ID; When the current server allow to reuse SSL Session ID, the request carrying the ID will be distributed to the same server as that of the client accessed for the first time.
SNI	HTTPS/SSL-OVER-TCP	SNI (Server Name Identification) is an extension option of SSL protocol, it is used to It is used to identify different domain names on the same IP address/port

Session Persistence Method-SIP-TCP Service



Session Persistence	Type Supported	Description
SIP Call-ID	SIP	In SIP protocol, Call-ID uniquely identifies the session that is being established globally; since multiple channels are involved in a SIP call, we can obtains the Call-ID from a SIP request of a client, requests carrying the Call-ID will be distributed to the same server as that of the client accessed for the first time.

Real Source IP

- “X-Forwarded-For” is the most common used method for real source IP transmission, as the graph shown, IP from left to right are client IP, proxy 1, proxy 2…
- “X-Real-IP” is IP of the last proxy or client.
- “X-Forwarded-For”, “X-Real-IP” usually used for proxy; TOA usually used for layer four device.
- The priority of obtaining a real source IP is as follows: X-Forwarded-For > X-Real-IP > the source IP carried in the message.

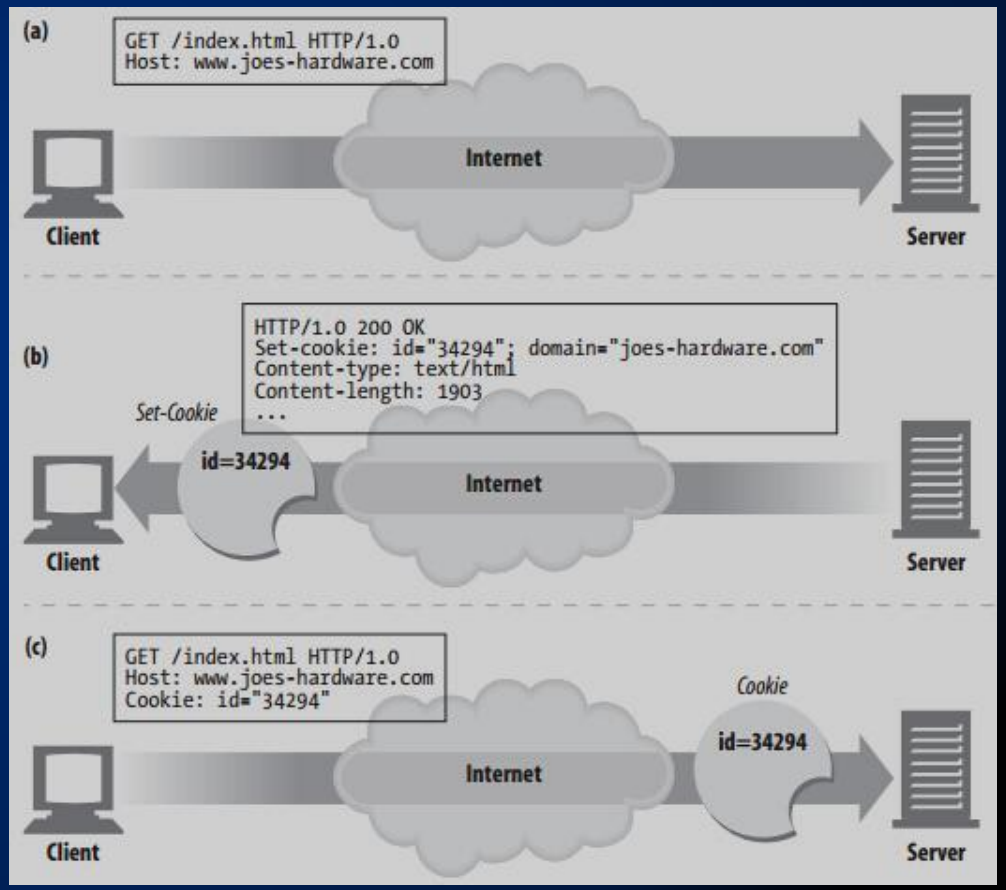
```
GET /foobar.jpg HTTP/1.1
Host: your_origin_host
X-Forwarded-Host: <zonename>-<id>.kxcdn.com
X-Forwarded-For: 178.82.72.134, 173.2.73.234
X-Forwarded-Scheme: http
X-Pull: KeyCDN
Connection: close
Accept: */*
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5)
Accept-Language: en-US,en;q=0.8,de;q=0.6,ja;q=0.4
Cookie: foobar
```

```
GET /foobar.jpg HTTP/1.1
Host: your_origin_host
X-Real-IP: 178.82.72.134
X-Pull: KeyCDN
Connection: close
Accept: */*
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5)
Accept-Language: en-US,en;q=0.8,de;q=0.6,ja;q=0.4
Cookie: foobar
```

Cookie

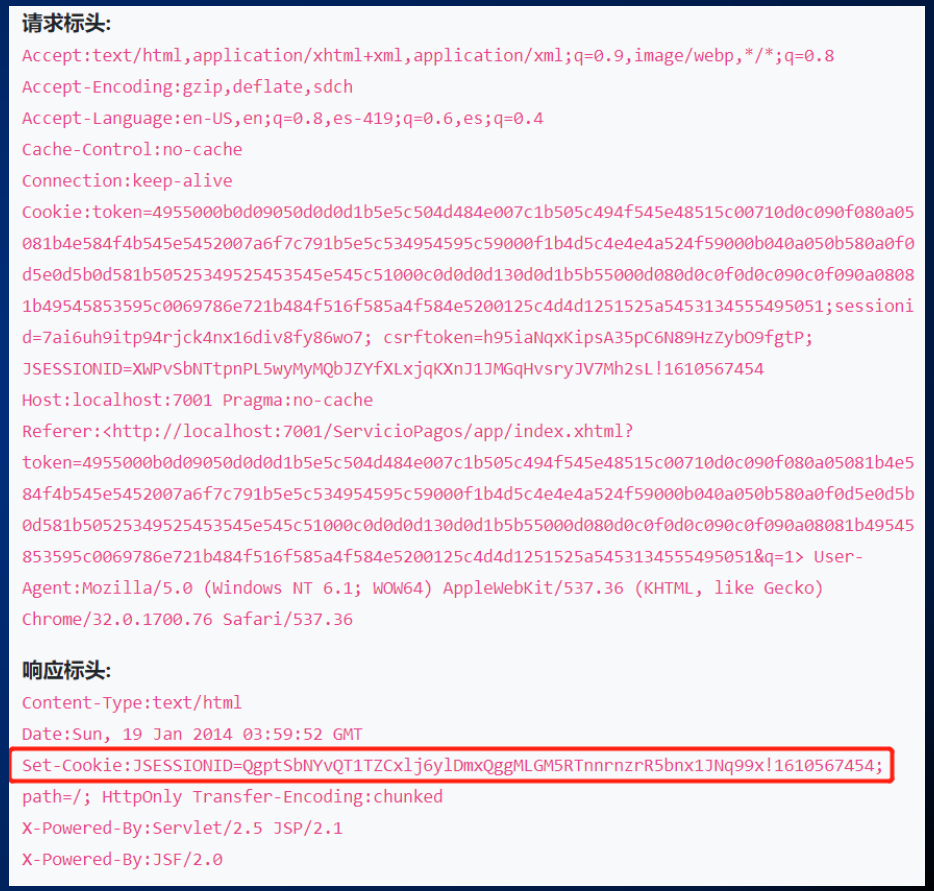
Cookie Session Persistence can be categorized as: cookie (hash), insert cookie, and rewrite cookie.

- Why we need Cookie: HTTP is a stateless protocol, cookies are a mechanism for the server to record the state of the client.
- Cookie Type: Session cookie will only be stored into the memory of client (browser), if the client shutdown, cookie will be cleared; correspondently, Cookie can be set for a timeout period, and be stored on the hard disk, if the client be restarted after it shutdown, and the cookie is still in the validity period, it can still been used.
- For example, as the graph shown on the right:
  - When the client visiting server at the first time, server doesn't have any record about this client.
  - When the server return to the client, it set up a cookie, the name is "id", and the value is "34294", the server side can record some information according to "34294".
  - When client access to server at the second time, it will attach the Cookie with "id" for name, and "34294" for value, server can search for the previous record by information in the Cookie.

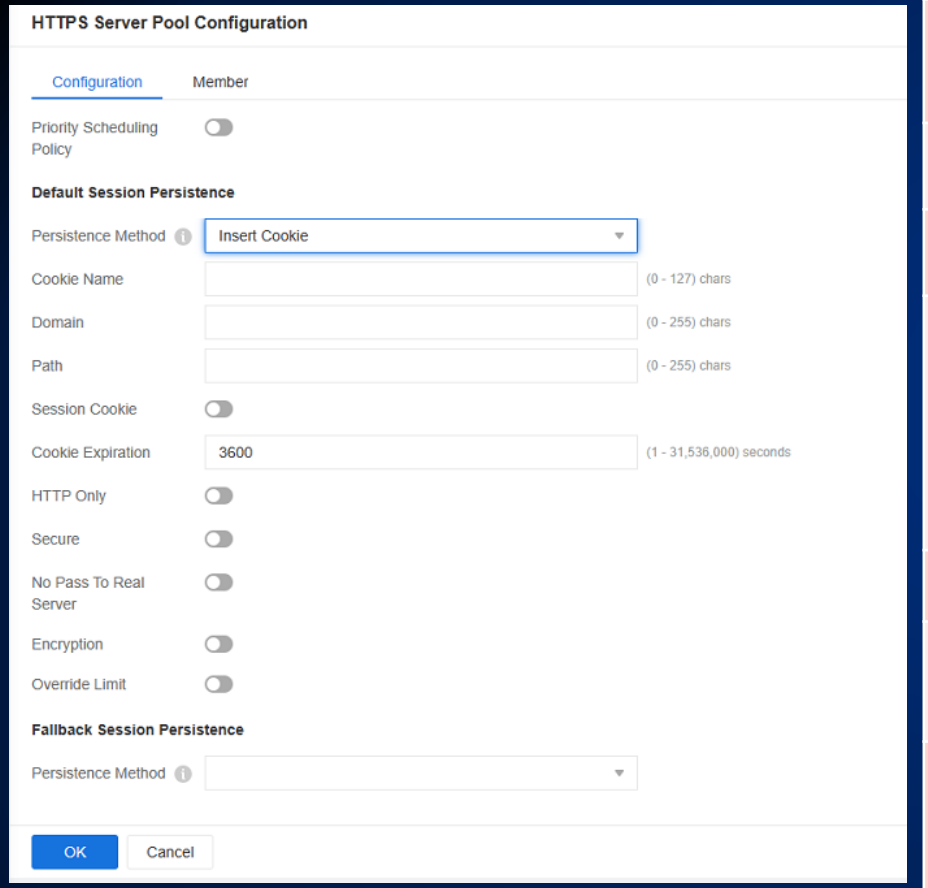


Cookie

- As shown in the picture, it is a single sign-on request and response.
- Typically, the server uses “JSESSIONID” to identify a user‘s login information for a certain period of time. When the client makes another request, the server can retrieve user (client)’s corresponding information based on the “JSESSIONID”.
- Generally, the purpose of session persistence is to keep a user connected to a backend server for a certain period of time. Therefore, it is necessary to have an understanding of the client's business, and know what kind of Cookie is appropriate.
- Typically, you can do a basic analysis of the client's business by using debug mode on the browser side or by capturing packets, or by debugging/capturing packets on the ADC device. This can help to select an appropriate type of Cookie for session persistence.



Insert Cookie



Config Item	Description
Cookie Name	Specify the name of the HTTP cookie. If the name is not specified, system will use the ID (such as SLBServerPool3) of the current address pool as the default cookie and insert it into the request
Domain	Specify a domain name to be matched. The client request that matches with it will be inserted with the cookie, and the session will be persistent.
Path	Specify a URL path to be matched. The client request that matches with it will be inserted with the cookie, and the session will be persistent.
Session Cookie/Cookie expiration	session cookie informs the client that the cookie will be deleted when the current client (browser) exits. On the other hand, if there is no session cookie, the cookie's expiration time needs to be configured. When the application wants to persist for a longer time (for example, wanting the login information to be valid for a month without visiting), the Cookie expiration needs to be configured to be more than one month.
HTTP Only	this cookie is only used in the HTTP protocol and is not effective when using HTTPS.
No Pass To Real Server	After the client request is delivered to the ADC, the cookie will be deleted to reduce interference with the server before the ADC sends the request to the RS . By default, the ADC will forward the inserted cookie to the RS.
Encryption	As previously stated, the ADC will schedule its business based on the value of the inserted cookie in the client's request, which records the pool and RS. This leaves a hidden danger, where an attacker can construct the cookie value to make the ADC route its business to a specific RS. This problem can be solved by "encryption." This feature will encrypt the value of the cookie and only the ADC can decrypt it, to preventing attackers from forging content in the cookie.

Insert Cookie

- Compared to Cookie hashing, inserting cookies is more straightforward and requires fewer knowledge of the client's business and what cookies are appropriate to use.
- Insert Cookies is done by the ADC device inserting a "Set-Cookie" header in the response, and subsequent client requests will carry this Cookie to access the ADC. The ADC can then determine which RS to dispatch the request to by parsing the Cookie.
- When the cookie name is not specified, the ADC device will automatically generate a cookie name in the format of "SLBServerPool<poolid>". The format of the cookie value is "0000.<vsid>.<poolid>.<rsid>.<timestamp>;...". When the ADC receives a request, it can parse the RS's ID based on the value of the cookie, and decide which RS the request should be dispatched to. Therefore, "Insert Cookies" does not require to generating a session persistence entry.
- To prevent the RS from seeing unnecessary new Cookies, it also able to configure the inserted Cookie not to be forwarded to the RS. In this case, ADC will delete the Cookies in request from client which inserted by ADC.

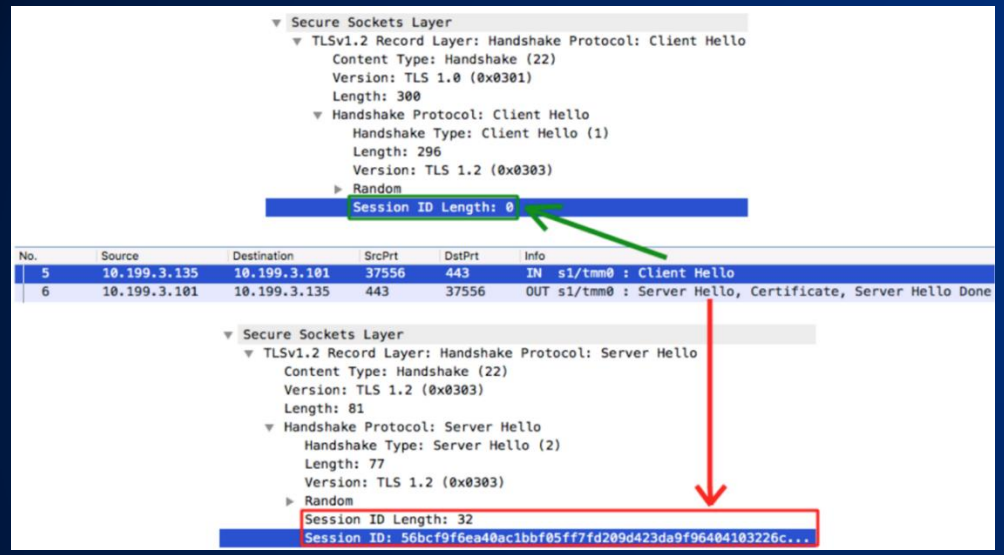
Session Persistence - SSL ID

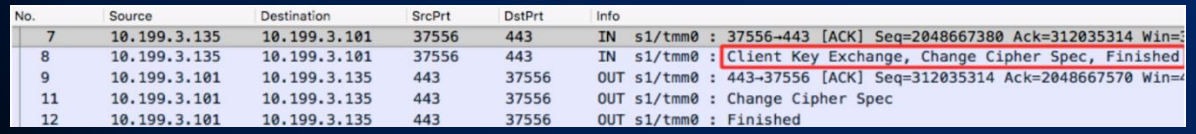
The SSL protocol provides two session recovery mechanisms: Session ID and Session Ticket. Since session ID is stored on the server, it has a multiple server caching synchronization problem.

- Translate Session ID: the correspondent secret key information to the Session ID is saved on the server side.
  - If a connection have been established between the client and server previously, the server returns the session ID after the handshake is successful established and saves the corresponding communication parameters on the server.
  - If the client needs to establish a connection with the same server again, it will carry the recorded information in the session ID in the client_hello and sends it to the server.
  - The server will retrieve the cached record based on the received session ID. If there is no retrieval or the cache has expired, it will process the normal handshake.
  - If the corresponding cache record is retrieved, the previously encrypted parameters will be used on this SSL connection.
- Session Ticket: the secret key information is saved on the client.
  - If the client and server have established a connection before, the server will carry the encrypted session_ticket information in the new_session_ticket data, and the client will save it.
  - If the client needs to connect to the server again, it carries the encrypted information in the session_ticket extension field in the client_hello, along with a session ID, and sends it to the server.
  - The server decrypted the session_ticket data, If decryption fails, it will respond empty session_ticket. Then server will send a new session_ticket message.
  - If decryption is successful, the server responds the same session ID in the server_hello, informing the client that the ticket has been accepted.
- Session persistence for SSL ID can be also known as do session persistence for session ID, keeping SSL connections with the same Session ID on the same RS.

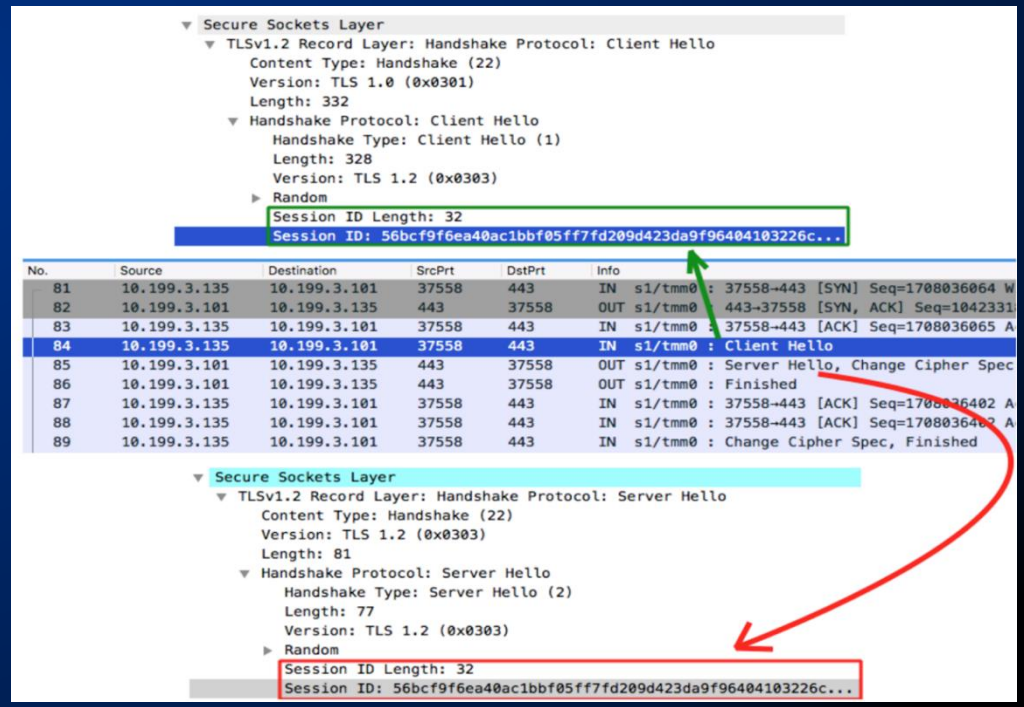
Session ID Recovery Process

- When starting a new session, the client sends an empty Session ID in the Client Hello, and the server will return a Session ID in the Server Hello.
- Then, a full certificate validation process and key exchange process will be proceed.



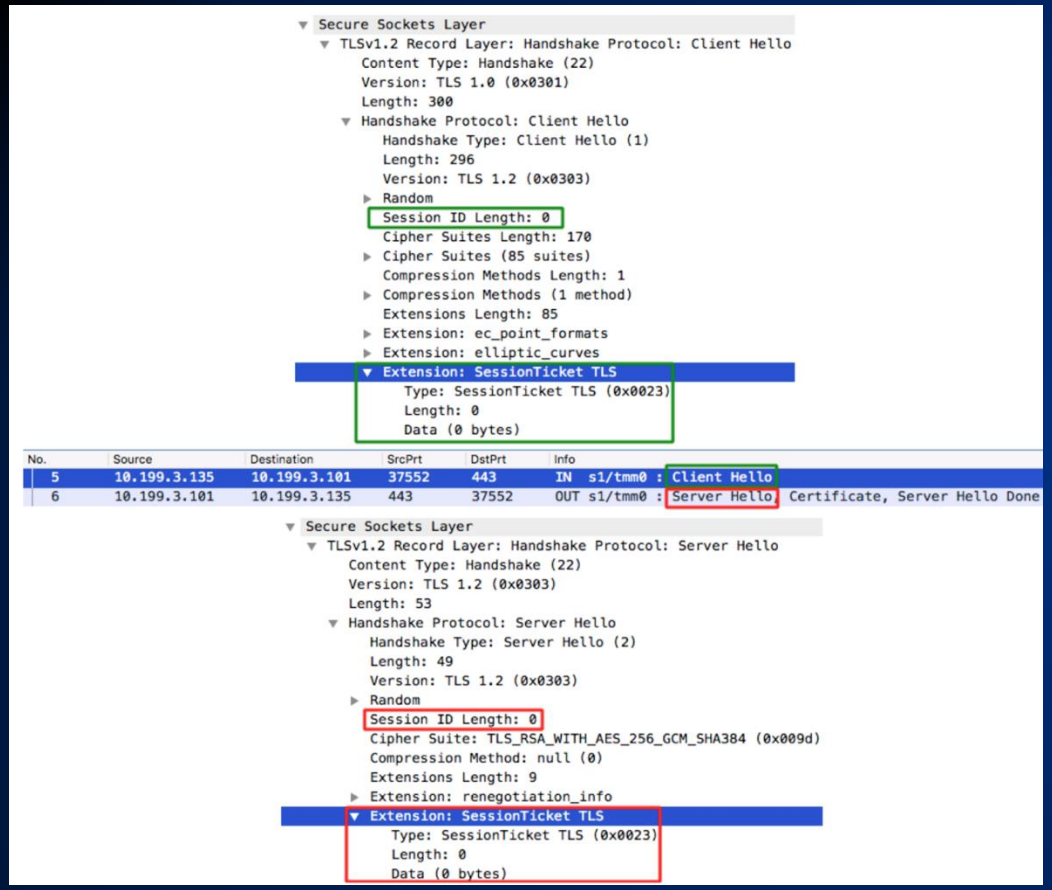


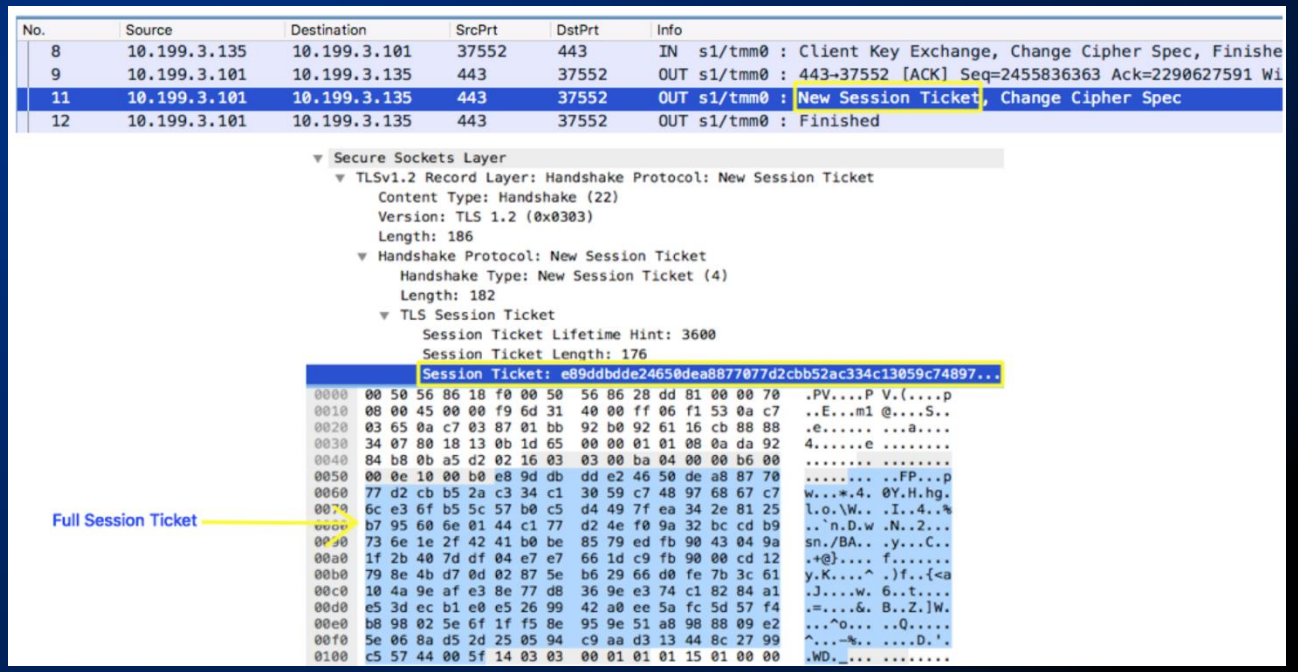
- When the client initiates a connection again, it will send the Session ID in the Client Hello.
- If the server confirms the information of the Session ID in its local cache, it will return the same Session ID in the Server Hello back to the client. If the server found it is invalid, a new Session ID will be returned.



Session Ticket - New Session

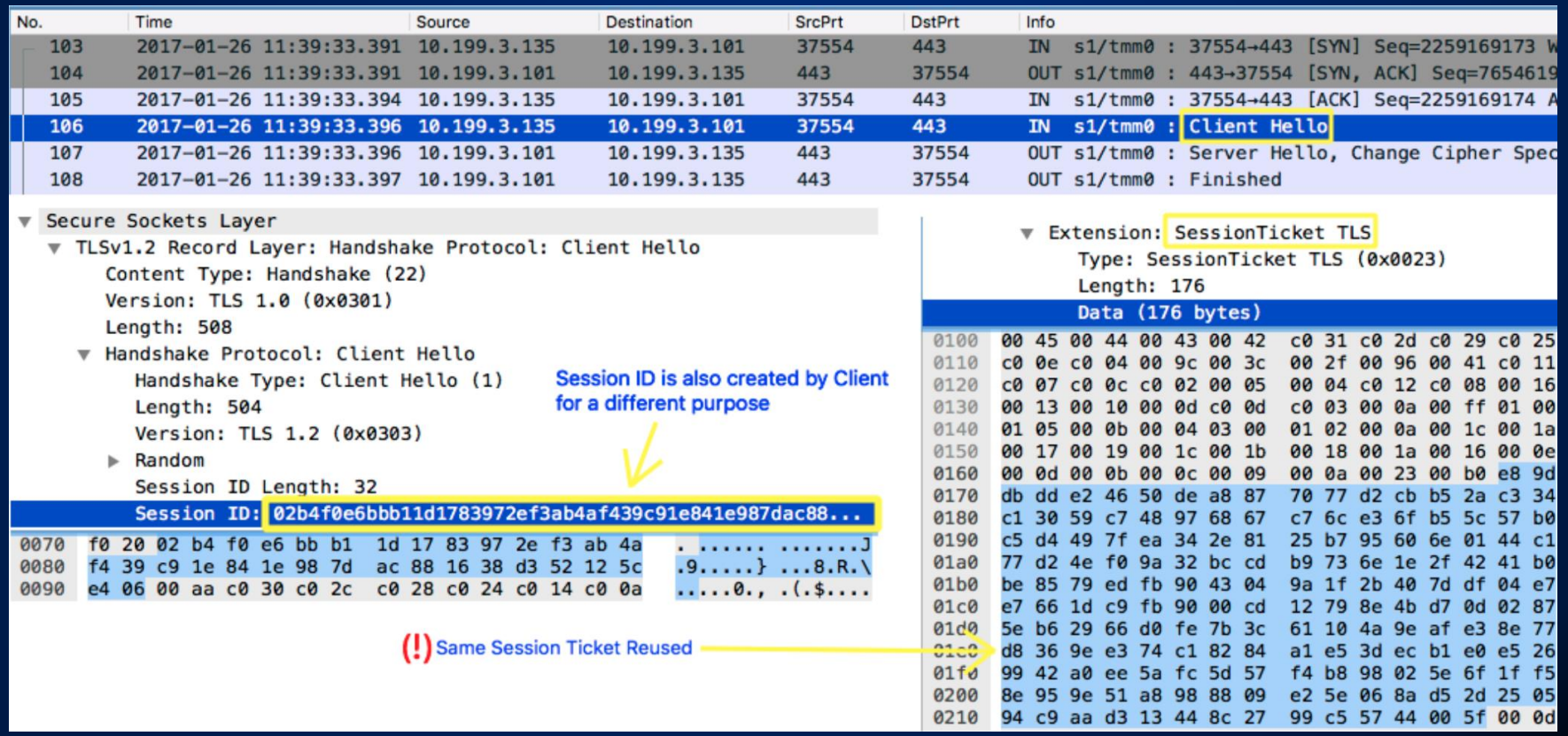
- When starting a new session, the client sends an empty Session Ticket in the Client Hello.
- After the server receive it, if the server also supports Session Ticket, it will also reply with an empty Session Ticket in the Server Hello.
- Then, the server initiates a new Session Ticket, and during this process, the server does not use the client's Session ID.





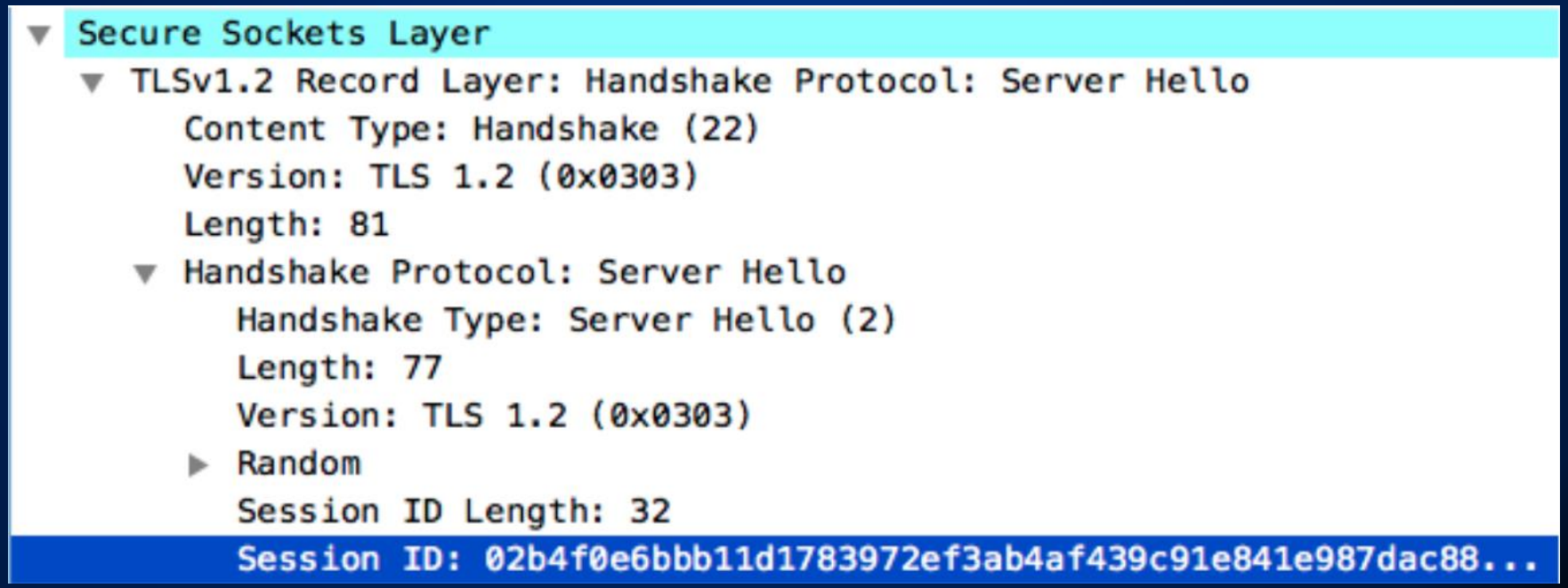
Session Ticket – Session Match

- When the client initiates a connection again, it will carry the Session Ticket in the Client Hello (which contains the previously negotiated key information).
- At the same time, the client creates a Session ID, and carry it in the Client Hello.



Session Ticket – Session Match

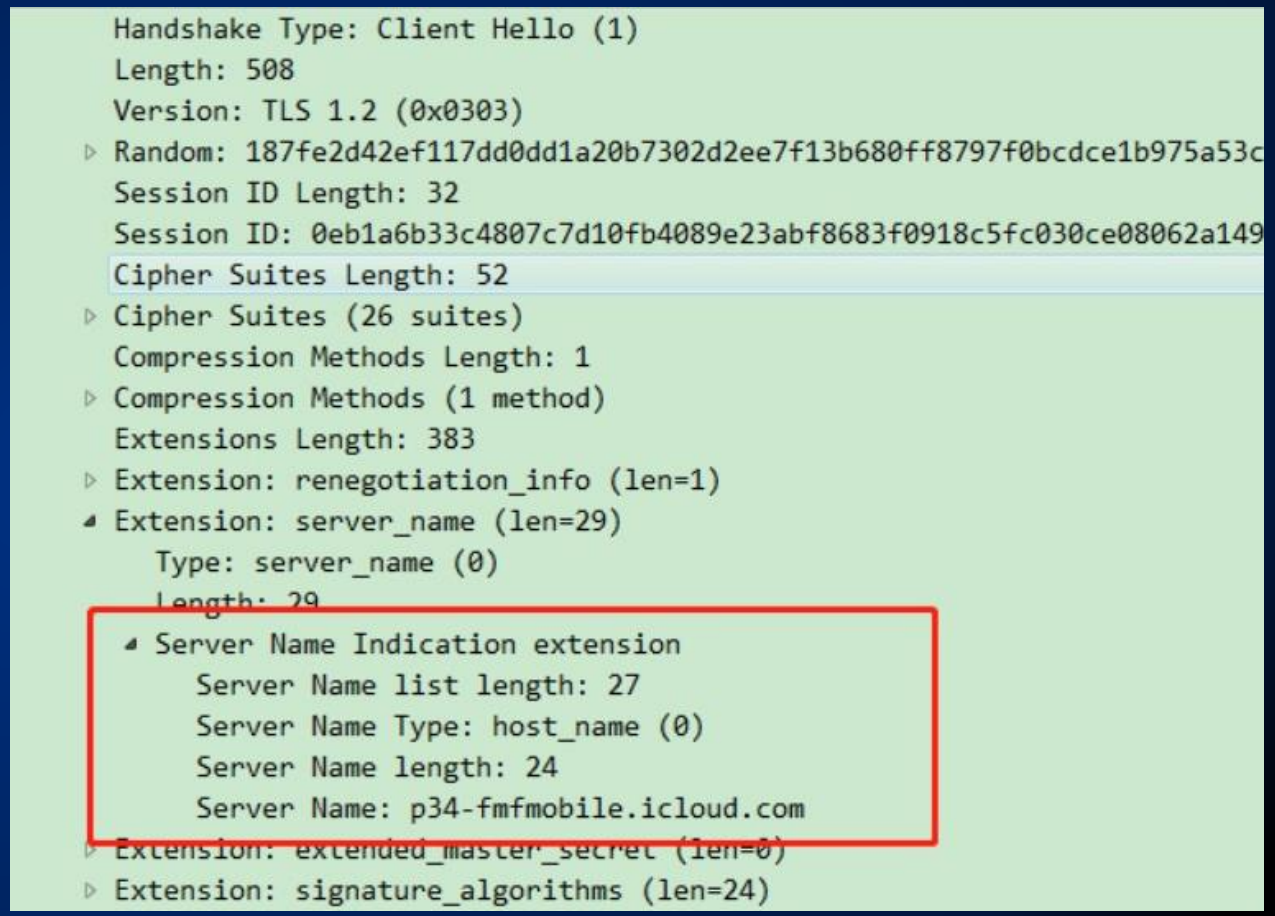
- If the server is able to correctly decrypt the Session Ticket, it will respond with the same Session ID in the Server Hello,
- in this process, the ADC does not need to record the information of the Session Ticket, as it is all recorded by the client, thus reducing the overhead on the server.



SNI

SNI (Server Name Identification) is an extension of the SSL protocol, designed to address the need to provide SSL services for multiple domains on the same port.

- Currently, most mainstream browsers and apps support the SNI extension when initiates requests.
- The server selects the correct certificate based on the SNI extension.



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