F5 STUDY GUIDE

301b – BIG-IP LTM Technology Specialist: Maintain and Troubleshoot

Eric Mitchell
Channel SE, East US and Federal
F5 Networks

Contents

Overview

Welcome to the 301b - BIG-IP LTM Specialist compiled Study Guide. The purpose of this guide is to help you prepare for the F5 301b - LTM Specialist exam. The contents of this document are based on the 301b - BIG-IP LTM Specialist Blueprint Guide.

This study guide provides students with some of the basic foundational knowledge required to pass the exam.

This study guide is a collection of information and therefore not a completely original work. The majority of the information is compiled from F5 sources that are located on Internet. All of the information locations are referenced at the top of each topic instead of in an Appendix of this document. This was done to help the reader access the reference the linked information easier without having to search through a formal appendix. This guide may also reference the same books as the exam Resource Guide for each topic when applicable for consistency.

F5 Networks provides the 301b - BIG-IP LTM Specialist Study Guide as a study guide. The Study Guide is a list of reading material that will help any student build a broad base of general knowledge that can assist in not only their exam success but in becoming a well rounded systems engineer. The Study Guide will be available to the candidate once they are qualified for the 301b - BIG-IP LTM Specialist exam.

Taking certified F5 LTM training, such as Administering BIG-IP v11 and Configuring BIG-IP LTM v11, will surely help with the topics of this exam but does not teach directly to the exam content. Hands on administrative experience with the BIG-IP platform licensed with LTM will reinforce many of the topics contained in the 301b - BIG-IP LTM Specialist exam.

The F5 301a - Local Traffic Manager Specialist: Architect, Setup and Deploy, stands as a pre-requisite to this exam.

This guide was prepared by an F5 employee but is not an official F5 document and is not supported by F5 Networks.

Reading = Knowledge = Power

Printed References

These referenced books are and important and should be considered basic reading material for this exam. If you have a newer copy of the material that is fine, be aware that the exam is based on the 11.2 version and content could have changed.

(Ref:1) Configuring BIG-IP Local Traffic Manager v11.2. v11.2.0 Edition. F5 Networks Training Course Manual.

(Ref:2) Administering BIG-IP v11.2. v11.2.0 Edition. F5 Networks Training Course Manual.

(Ref:3) Troubleshooting BIG-IP v11.2. v11.2.0 Edition. F5 Networks Training Course Manual.

(Ref:4) Developing iApps for BIG-IP v11.2. v11.2.0 Edition. F5 Networks Training Course Manual.

Introduction

F5 – 301b BIG-IP Local Traffic Manager Specialist Exam

The F5 BIG-IP Local Traffic Manager (LTM) increases an application's operational efficiency and ensures peak network performance by providing a flexible, high-performance application delivery system. With its application-centric perspective, LTM optimizes your network infrastructure to deliver availability, security, and performance for critical business applications. Although the Exam Blueprint is not written in a structure that presents topics in an educational order, it does provide many of the necessary building blocks. The Certified LTM Training classes from F5 will help with many of the scenario-based topics on the test. An LTM Specialist must be proficient with all aspects of maintaining and troubleshooting the LTM within a network.

Overview of SNAT Features

Traffic Management Shell

Although it is not mentioned in the blueprint as a requirement, a candidate should not focus only on the GUI interface for management of the LTM platform. Some test questions will refer to the command line interface (CLI) TMSH commands. You should take time to understand where in the CLI that common commands are issued so you can not only correctly answer the questions presented on the exam but also have enough knowledge of the CLI structure to eliminate bad commands from your question's answer choices.

Try building your vLab environment from command line to gain CLI proficiency.

Many of the Objective Topics are the same between each of the Objectives and refer to the same material. The difference will lie in the perspective of the Objective on that material. The documentation from F5 will not distinguish on the differences of the Objective. Hand-on experience is the key to passing this exam.

SECTION 1 – TROUBLESHOOT BASIC VIRTUAL SERVER CONNECTIVITY ISSUES

Objective - 1.01 - Given a scenario, determine the appropriate profile setting modifications

Scenario Based Objective

To prepare for scenario based questions the candidate will need to complete hands-on configuration and testing of the configuration on the LTM. This will allow the candidate to better understand how different configurations can produce different results. All F5 exams use scenario-based questions that make the candidate apply what they know to a situation to determine the resulting outcome.

This topic is focused on how profiles can affect applications and how to achieve the functionality needed for the application by tuning the profiles on the BIG-IP virtual server.

1.01a – Determine the effect of changing profile settings on application traffic

Profile Concepts
Overview of the Stream Profile
Other Application-Layer Profiles
Session Persistence Profiles
Protocol Profiles

Profile Settings Behavior

There are over 25 different profile types and many different settings in each one. To know every single setting and the impact of changing the values of each setting and how it will impact the myriad of applications that we encounter in all of the environments running BIG-IP is beyond a daunting feat, it is likely impossible.

Understanding the more common profiles like TCP, HTTP and the others, that are commonly used with HTTP based traffic, is fairly doable. Read as much as you can to understand the common settings in these profiles to help with questions on this exam. With years of experience working with application delivery networks, most of this will become second nature to you.

Profiles that are assigned to a virtual server tell the virtual server how to process the traffic flowing through that virtual server. Some profile settings can make changes to the content of a packet as it flows through the

BIG-IP and some server as an application behavior regulator, while others tell the BIG-IP how to act based on what it is seeing with the application. And there are some profiles that can do a mixture of all of these actions.

An example of changes in a profile that will change the application traffic:

Stream Profile

The Stream profile performs a search and replace procedure for all occurrences of a string in a data stream.

How the Stream profile operates:

Standard TCP virtual server with no HTTP profile

When the Stream profile is applied to a standard TCP virtual server that is not configured with the HTTP profile, the search and replace procedure occurs on the entire data portion of each TCP segment. This applies to both data streams (the data transmitted by the client and the data transmitted by the server).

If such a virtual server processes L4 connections encapsulating HTTP requests and responses, the search and replace procedure is performed on both the HTTP headers and the HTTP payload (that is, the entire data portion of each TCP segment).

Standard TCP virtual server with HTTP profile

When the Stream profile is applied to a standard TCP virtual server that is also configured with the HTTP profile, the search and replace procedure occurs only on the HTTP payload. This applies to both client requests and server responses.

1.01b – Determine if changing profile settings will affect the LTM device or the application

Profile Concepts
Changing the Idle Timeout for a Protocol Profile
Other Application-Layer Profiles
Session Persistence Profiles
Protocol Profiles

Profile Settings Behavior

As I stated in section 1.01a, profiles that are assigned to a virtual server tell the virtual server how to process the traffic flowing through that virtual server. Some profile settings can make changes to the content of a packet as it flows through the BIG-IP and some server as an application behavior regulator, while others tell the BIG-IP how to act based on what it is seeing with the application. And there are some profiles that can do a mixture of all of these actions.

An example of changes in a profile affecting the LTM device:

Changing the idle timeout for a protocol profile

If idle connections are allowed to remain in the BIG-IP connection table for extended periods, they continue to consume system memory, which reduces the amount of memory available for new connections. To address this, you can adjust the idle timeout setting for the relevant protocol profile for a virtual server. The idle timeout setting specifies the length of time that a connection is idle before it is eligible for deletion by the BIG-IP LTM system.

Changing the idle timeout for an existing protocol profile

1. Log in to the Configuration utility.
2. Navigate to Local Traffic > Profiles.
3. Click the Protocol tab.
4. In the box, select the relevant protocol profile. For example, FastL4.
5. Select the profile to modify.
6. Type the new Idle Timeout value, in seconds.
7. Click Update.

   Note: Changing the idle timeout value for an existing protocol profile does not alter the timeout used for existing virtual server connections; existing connections continue to use the previous idle timeout value. To force an existing connection to use the new timeout value, you must manually remove the existing connection from the connection table using the TMSH delete sys connection command.

1.01c – Explain the effect of modifying buffer settings in the TCP profile

Investigating the LTM TCP Profile: Windows & Buffers

TCP Profile Buffer Settings

For equally fast clients and servers, there is no need to buffer content between them. However, if the client or server falls behind in acknowledging data, or there are lossy conditions, the proxy will begin buffering data. The proxy buffer high setting is the threshold at which the LTM stops advancing the receive window. The proxy buffer low setting is a falling trigger (from the proxy high setting) that will re-open the receive window once passed. Like the window, increasing the proxy buffer high setting will increase the potential for additional memory utilization per connection.

Typically the client side of a connection is slower than the server side, and without buffering the data the client forces the server to slow down the delivery. Buffering the data on the LTM allows the server to deliver its data so it can move on to service other connections while the LTM feeds the data to the client as quickly as possible. This is also true the other way in a fast client/slow server scenario.

With version 9.3, the LTM began shipping with pre-configured optimized tcp profiles for the WAN & LAN environments. The send buffer and the receive window maximums are both set to the max non-scaled window size at 64k (65535), and the proxy buffer high is set at 131072. For the tcp-lan-optimized profile, the proxy buffer low is set at 98304 and for the tcp-wan-optimized, the proxy buffer low is set the same as the high at 131072.

So for the LAN optimized profile, the receive window for the server is not opened until there is less than 98304 bytes to send to the client, whereas in the WAN optimized profile, the server receive window is opened as soon as any data is sent to the client. Again, this is good for WAN environments where the clients are typically slower.

1.01d – Explain the effect of modifying time out settings in the TCP/ UDP profile

Investigating the LTM TCP Profile: Windows & Buffers

Idle Timeout

The explanation of the idle timeout is fairly intuitive. This setting controls the number of seconds the connection remains idle before the LTM closes it. For most applications, the default 300 seconds is more than enough, but for applications with long-lived connections like remote desktop protocol, the user may want to leave the desk and get a cup of coffee without getting dumped but the administrators don't want to enable keep alives. The option can be configured with a numeric setting in seconds, or can be set to indefinite, in which case the abandoned connections will sit idle until a reaper reclaims them or services are restarted. I try to isolate applications onto their own virtual servers so I can maximize the profile settings, but in the case where a wildcard virtual is utilized, the idle timeout can be set in an iRule with the IP::idle_timeout command:

```
when CLIENT_ACCEPTED {
  switch [TCP::local_port] {
    “22” {
      IP::idle_timeout 600
    }
    “23” {
      IP::idle_timeout 600
    }
    “3389” {
      IP::idle_timeout 3600
```

```
  }
    default {
      IP::idle_timeout 120
    }
}
```

If you look at the connection table, the current and the maximum (in parentheses) idle values are shown.

```
b conn client 10.1.1.1 show all | grep –v pkts

VIRTUAL 10.1.1.100:3389 <-> NODE any6:any
  CLIENTSIDE 10.1.1.1:36023 <-> 10.1.1.100:3389
  SERVERSIDE 10.1.1.100:36023 <-> 10.1.1.50:3389
   PROTOCOL tcp  UNIT 1  IDLE 124 (3600)  LASTHOP 1 00:02:0a:13:ef:80
```

1.01e – Determine the effect of TCP/UDP profile interaction with other profiles (e.g. OneConnect, HTTP, FTP, etc. For example, how modifying settings in TCP profile can break FTP)

Overview of the OneConnect Profile
Choosing Appropriate Profiles for HTTP Traffic
The Deferred Accept TCP Profile Option May Not be Compatible with Some Applications

OneConnect

The BIG-IP system OneConnect feature can increase network throughput by efficiently managing connections created between the BIG-IP system and back-end pool members. The OneConnect feature works with HTTP Keep-Alives to allow the BIG-IP system to minimize the number of server-side TCP connections by making existing connections available for reuse by other clients.

For example, when a client makes a new connection to a BIG-IP virtual server configured with a OneConnect profile, the BIG-IP system parses the HTTP request, selects a server using the load-balancing method defined in the pool, and creates a connection to that server. When the client's initial HTTP request is complete, the BIG-IP system temporarily holds the connection open and makes the idle TCP connection to the pool member available for reuse.

When a new connection is initiated to the virtual server, if an existing server-side flow to the pool member is idle, the BIG-IP system applies the OneConnect source mask to the IP address in the request to determine whether it is eligible to reuse the existing idle connection. If it is eligible, the BIG-IP system marks the connection as non-idle and sends a client request over it. If the request is not eligible for reuse, or an idle server-side flow is not found, the BIG-IP system creates a new server-side TCP connection and sends client

requests over it. Because the OneConnect profile reuses idle connections, it may appear that the BIG-IP system is not load balancing traffic evenly across pool members.

> **Note:** Using a broad source mask may skew server-side statistical data. For example, when you use the mask `0.0.0.0`, requests from multiple clients may appear as though they are originating from a single IP address.

OneConnect and SNATs

When a client makes a new connection to a BIG-IP virtual server configured with a OneConnect profile and Secure Network Address Translation (SNAT), the BIG-IP system parses the HTTP request, selects a server using the load-balancing method defined in the pool, translates the source IP address in the request to the SNAT IP address, and creates a connection to the server. When the client's initial HTTP request is complete, the BIG-IP system temporarily holds the connection open and makes the idle TCP connection to the pool member available for reuse. When a new connection is initiated to the virtual server, the BIG-IP system performs SNAT address translation on the source IP address, and then applies the OneConnect source mask to the translated SNAT IP address to determine whether it is eligible to reuse an existing idle connection.

Content Switching

When an OneConnect profile is enabled for an HTTP virtual server, and an HTTP client sends multiple requests within a single connection, the BIG-IP system is able to process each HTTP request individually. The BIG-IP system sends the HTTP requests to different destination servers as determined by the load balancing method. Without an OneConnect profile enabled for the virtual server, the BIG-IP system performs load-balancing only once for each TCP connection.

> **Note:** If no OneConnect profile is configured for the virtual server, certain persistence methods can occasionally fail due to HTTP parsing issues. For more information, refer to SOL7964: Persistence may fail for subsequent requests on Keep-Alive connections.

When an OneConnect profile is enabled for a TCP virtual server that does not have an HTTP profile applied, and a client sends multiple requests within a single connection, the BIG-IP system is able to process each request individually. The BIG-IP system sends the requests to different destination servers as determined by the load balancing method. Without an OneConnect profile enabled for the virtual server, the BIG-IP system performs load-balancing only once for each TCP connection.

HTTP considerations

For HTTP traffic to be eligible for OneConnect connections, the web server must support HTTP Keep-Alive connections.

HTTP/1.1 requests

HTTP Keep-Alive connections are enabled by default in HTTP/1.1. With HTTP/1.1 requests, the server does not close the connection when the content transfer is complete, unless the client sends a Connection: close

header in the request. Instead, the connection remains active in anticipation of the client reusing the same connection to send additional requests.

HTTP/1.0 requests

HTTP Keep-Alive connections are not enabled by default in HTTP/1.0. With HTTP/1.0 requests, the client typically sends a Connection: close header to close the TCP connection after sending the request. Both the server and client-side connections that contain the Connection: close header are closed once the response is sent.

OneConnect Transformations

The OneConnect Transformations setting in the HTTP profile allows the BIG-IP system to perform HTTP header transformations for the purpose of allowing HTTP/1.0 connections to be transformed into HTTP/1.1 requests on the server side, thus allowing those connections to remain open for reuse when they would not otherwise be. The default setting is enabled.

When the OneConnect Transformations setting is enabled in the HTTP profile, the BIG-IP system transforms Connection: close headers in HTTP/1.0 client-side requests to X-Cnection: close headers on the server side. This allows the BIG-IP system to make client requests containing the Connection: close header such as HTTP/1.0 requests, eligible for connection reuse.

> **Note:** For more information, refer to SOL6997: Overview of HTTP headers used by BIG-IP to manage OneConnect connections.
>
> **Note:** The FastHTTP profile uses an implementation of OneConnect that transforms Connection: close headers to Xonnection: Close.
>
> **Note:** NTLM's HTTP 401 responses prevent the BIG-IP system from detaching the server-side connection. As a result, a late FIN from a previous client connection may be forwarded to a new client that reused the connection, causing the client-side connection to close before the NTLM handshake completes. If NTLM authentication support is desired when using the OneConnect feature, the NTLM profile introduced in BIG-IP 10.0.0 should be configured as well. For more information, refer to the Configuring an NTLM profile chapter in the Configuration Guide for your BIG-IP product.

OneConnect Profile Settings

The following OneConnect profile settings control OneConnect behavior and affect server-side connections:

- The Source Mask setting specifies the mask applied to the source IP address to determine the connection's eligibility to reuse a server-side connection. For more information, refer to SOL5911: Managing connection reuse using OneConnect source mask.
- The Maximum Size setting represents the maximum number of idle server-side connections kept in the connection pool.

- The Maximum Age setting represents the maximum number of seconds a server-side connection is allowed before the connection is deleted from the connection pool.
- The Maximum Reuse setting represents the maximum number of requests to be sent over a server-side connection. This number should be slightly lower than the maximum number of HTTP Keep-Alive requests accepted by back-end servers in order to prevent the back-end server from initiating connection close and entering the TIME_WAIT state.
- The Idle Timeout Override setting represents the maximum time that idle server-side connections are kept open. Lowering this value may result in a lower number of idle server-side connections, but may increase request latency and server-side connection rate.
- The Limit Type setting (available in BIG-IP 11.6.0 and later) controls how connection limits are enforced in conjunction with OneConnect. The following values are available:
    - None: (Default)
    - Idle: Specifies that idle connections will be dropped as the TCP connection limit is reached. For short intervals, during the overlap of the idle connection being dropped and the new connection being established, the TCP connection limit may be exceeded.
    - Strict: Specifies that the TCP connection limit is honored with no exceptions. This means that idle connections will prevent new TCP connections from being made until they expire, even if they could otherwise be reused. This is not a recommended configuration, except in very special cases with short expiration timeouts.

Recommendations

- When using the OneConnect feature, you should consider the following factors:
- When using OneConnect to optimize HTTP traffic, F5 recommends that you apply an HTTP profile to the virtual server. This allows the BIG-IP system to efficiently manage connection reuse without additional configuration. Failure to apply an HTTP profile may result in unexpected behavior such as backend server traffic being sent to the wrong client.
- Avoid using a OneConnect profile for non-HTTP virtual servers that process more complex transactions, such as FTP or RTSP. Doing so may result in traffic disruption and session failure. Even for simple non-HTTP protocols, an iRule may be required to manage connection reuse.
- The OneConnect profile may be used with any TCP protocol, but only when applied to virtual servers that process simple request/response protocols where transaction boundaries are explicitly obvious, such as those in which each request and each response is contained within a single packet.

- Avoid using an OneConnect profile for encrypted traffic that is passed through the virtual server to the destination resources in the encrypted state and is not terminated at the BIG-IP system.

HTTP

The BIG-IP system allows you to process HTTP traffic using various profiles, including TCP+HTTP, FastHTTP, and FastL4. Each profile, or combination of profiles, offers distinct advantages, limitations, and features.

F5 recommends that you assess the needs of each HTTP virtual server individually, using the following information, to determine which profile, or profile combination, best meets the requirements for each virtual server.

Important: The HTTP profile will work in all cases; however, the HTTP profile places BIG-IP in full Layer 7 inspection mode, which may be unnecessary when used on simple load balancing virtual servers. Thus, you should consider the other profile options provided in instances where the full Layer 7 engine is not necessary for a particular virtual server.

TCP+HTTP

Profiles: TCP+HTTP

Advantage: The HTTP profile can take full advantage of all of BIG-IP system's Layers 4 - 7 HTTP/HTTPS features.

When to use: The HTTP profile is used when any of the following features are required:

- IPv6 support
- TCP Express and content spooling features reduce server load
- Full OneConnect functionality (including HTTP 1.0 transformations)
- Layer 7 persistence (cookie, hash, universal, and iRule)
- Full HTTP iRules logic
- Cache and Web Acceleration features
- HTTP Compression
- HTTP pipelining
- Virtual Server Authentication
- Redirect Rewriting
- SPDY protocol support (11.3.0 and later)

Limitations

- More CPU-intensive
- Memory utilization:
  - Cache / Web Acceleration

    The caching / web acceleration features provision user-defined memory for cache content for each virtual server that uses the given HTTP and Cache profiles.
  - Compression

    Larger buffer sizes can increase memory utilization when compressing large objects.
  - TCP offloading/content spooling

    This can increase memory utilization in cases where either the client-side or the server-side of the connection is slower than the other. The BIG-IP system holds the data in the buffer until the slower side of the connection is able to retrieve it.

FTP

When enabled, the Deferred Accept TCP profile option specifies that the system does not dedicate resources to the connection until the system has received the data packet from the client. This setting is useful when negotiating 3-way handshake denial-of-service attacks.

The Deferred Accept option is disabled on the BIG-IP system by default.

The Deferred Accept option is not compatible with virtual servers (such as FTP, POP3, and SMTP) for applications that either require dialog or present a banner. For these applications you can define a custom TCP profile with the Deferred Accept option disabled and configure the non-compatible virtual servers to use this custom TCP profile.

1.01f – Describe how the source of traffic affects TCP/UDP profile settings that should be selected

Optimize WAN and LAN Application Performance with TCP Express

Tuning Layer 4 Profiles

The transmission control protocol (layer 4) rides on top of the Internet protocol (layer 3) and is responsible for establishing connections between clients and servers so data can be exchanged reliably between them. UDP does this same function without the reliability.

Normal TCP communication consists of a client and a server, a 3-way handshake, reliable data exchange, and a four-way close. With the LTM as an intermediary in the client/server architecture, the session setup/teardown is duplicated, with the LTM playing the role of server to the client and client to the server. These sessions are completely independent of each other. This places the LTM in a great position to optimize the TCP connections for both client and server separately from each other. If the client side traffic is generally slower or is coming in across a slow WAN then F5 can use a TCP profile that is tuned to work best for that WAN traffic speed. Alternately the client side communications are all LAN based, that conversation can be optimized for the LAN speeds.

For enterprises delivering Internet and extranet applications, TCP/IP inefficiencies, coupled the effects of WAN latency and packet loss, all conspire to adversely affect application performance. The result of these inefficiencies has inflated the response times for applications, and significantly reduced bandwidth utilization efficiency (ability to “fill the pipe”).

F5’s BIG-IP Local Traffic Manager provides a state-of-the-art TCP/IP stack that delivers dramatic WAN and LAN application performance improvements for real-world networks. These advantages cannot be seen in typical packet blasting test harnesses; rather they are designed to deal with real-world client and Internet conditions.

This highly optimized TCP/IP stack, called TCP Express, combines cutting-edge TCP/IP techniques and improvements in the latest RFCs with numerous improvements and extensions developed by F5 to minimize the effect of congestion and packet loss and recovery. Independent testing tools and customer experiences have shown TCP Express delivers up to a 2x performance gain for end users and a 4x improvement in bandwidth efficiency with no change to servers, applications, or the client desktops.

Customizable TCP Controls

While TCP Express is automatic and requires no modifications, the BIG-IP gives users advanced control of the TCP stack to tune TCP communications according to specific business needs. This includes the ability to select optimizations and settings at the virtual server level, per application being fronted on the device. Administrators can use a TCP profile to tune each of the following TCP variables:

- time wait recycle
- delayed acks
- proxy mss
- proxy options
- deferred accept
- selective acks

- ecn
- limited transmit
- rfc1323
- slow start
- bandwidth delay
- nagle
- proxy buffer

Administrators can also use these controls to tune TCP communication for specialized network conditions or application requirements. Customers in the mobile and service provider industries find that this flexibility gives them a way to further enhance their performance, reliability, and bandwidth utilization by tailoring communication for known devices (like mobile handsets) and network conditions.

Tuning Stack Settings for Applications

TCP Express provides flexible stack settings to optimize custom services. For example, you can adjust these settings to optimize an ASP application delivered to mobile users.

Optimizing Highly Interactive Traffic over the LAN

If the traffic on a LAN is highly interactive, F5 recommends a different set of TCP settings for peak performance. F5 has found that Nagle's algorithm works very well for packet reduction and general compression/RAM caching over a WAN. In addition, tweaks to various buffer sizes can positively impact highly interactive communications on low-latency LANs with the only possible cost of increased memory utilization on the BIG-IP.

1.01g – Determine the appropriate settings to use based on a given application behavior

Profiles for Managing HTTP Traffic
Session Persistence Profiles
Overview of the HTTP Class Profile

Determining configuration settings

You will be required to evaluate a scenario-based question that will give you required needs of an application or simply explain what is or is not working currently. From that information you will have to evaluate a configuration to recognize what is not configured or is configured that is causing the issue. You will need to be

familiar with all of the possible settings in the HTTP profile and some of the other common profiles that can affect web based traffic.

Know concepts like when is it appropriate to set a fallback server in the HTTP profile and what problem it solves. Understand why persistence is needed and what types work best with different applications. For instance you can't use cookie persistence with FTP based applications and sourced IP address types tend to not work well with clients who's IP addresses are proxied to the internet.

Understand the functions of an HTTP Class profile and when it is appropriate to use them. An HTTP Class profile allows you to sort selected HTTP traffic and perform an action based on the profile configuration. For example, you can send the traffic to a destination, such as a load balancing pool or rewrite the URI. F5 recommends using HTTP Class profiles when it is possible to classify HTTP traffic using simple strings or regex patterns. For more complex operations, you may need to use an iRule.

1.01h – Determine the appropriate netmask settings when using a OneConnect profile

Managing Connection Reuse Using OneConnect Source Mask

Understanding the OneConnect Netmask Setting

The BIG-IP OneConnect feature can increase network throughput by efficiently managing connections created between the BIG-IP system and back end nodes. OneConnect allows the BIG-IP system to minimize the number of server-side TCP connections by making existing idle connections available for reuse by other clients. The OneConnect source mask setting manages connection reuse, and is applied to the server-side source IP address of a request to determine its eligibility for connection reuse.

Overview of the OneConnect Mask

OneConnect applies a mask (much like applying an independent subnet mask) to client source IP addresses on server-side connections. This mask determines the availability of an existing idle TCP connection for the request on the selected destination server. The following list displays the idle TCP port reuse behavior. To simplify things, regarding these explanations, F5 assumes that a previous TCP request has established a connection on the destination server, the connection is idle, and the same destination server has been selected for a subsequent client request.

> **Note:** If SNAT is configured, the BIG-IP system performs SNAT address translation on the source IP address, and then applies the OneConnect source mask to the translated SNAT IP address to determine whether it is eligible to reuse an existing idle connection. For more information about the OneConnect profile and SNATs, refer to SOL7208: Overview of the OneConnect profile.

OneConnect Mask	OneConnect masking behavior
`255.255.255.255`	The entire client IP address is evaluated. A request from the same address reuses an established idle TCP connection.
`255.255.255.0`	Octets 1-3 are evaluated client addresses matching this subset and reuse an existing idle TCP connection.
`255.255.0.0`	Client addresses are evaluated based on the first and second octets when selecting an idle TCP connection for reuse.
`255.0.0.0`	Only the first octet of a client IP address is evaluated when selecting an idle TCP connection for reuse.
`0.0.0.0`	The all zeros mask (the default setting) looks for any open idle TCP connection on the destination server. The main difference is OneConnect does not attempt to group the request based on octets matched, but uses open idle TCP connections in a highly efficient manner.

Note: Using a broad source mask may skew server-side statistical data. For example, when using the mask 0.0.0.0, requests from multiple clients may appear as though they are originating from a single IP address.

Effects of modifying the source mask

The following three scenarios describe the effects of using different source masks:

Using a OneConnect profile with a 0.0.0.0 source mask

An OneConnect profile with a source mask of `0.0.0.0` shares idle connections across all client requests in the following manner:

- Client A with source IP address `10.10.10.10` connects to a virtual server.
- The BIG-IP system load-balances the connection and applies the source mask to the request on the server-side flow, finds no suitable connection for reuse, and creates a TCP connection to server A in the pool.
- Client B with source IP address `10.20.20.20` connects to the same virtual server.
- The BIG-IP system load-balances the connection, applies the source mask to the server-side flow, and finds an eligible idle TCP connection.
- The BIG-IP system aggregates the request from client B over the existing TCP connection created for client A.

Using a OneConnect profile with a 255.255.255.0 source mask

An OneConnect profile with a source mask of `255.255.255.0` aggregates connections from client IP addresses sharing the same last octet in the following manner:

- Client A with a source IP address of `10.10.10.10` connects to a virtual server.
- The BIG-IP system load-balances the connection and applies the source mask to the request on the server-side flow, finds no suitable connection for reuse, and creates a TCP connection to server A in the pool.
- Client B with a source IP address of `10.10.10.100` connects to the same virtual server.
- The BIG-IP system load-balances the connection, applies the source mask to the server-side flow, and finds an eligible idle TCP connection.
- The BIG-IP system aggregates the request from client B over the existing TCP connection created for client A.
- Client C with source IP address `10.10.20.10` connects to the same virtual server.
- The BIG-IP system load-balances the connection, applies the source mask to the server-side flow, and finds no suitable connection for reuse.
- The BIG-IP system creates a new TCP connection to the selected pool member.

Using an OneConnect profile with a 255.255.255.255 source mask

An OneConnect profile with a source mask of `255.255.255.255` will only aggregate connections originating from the same server-side client IP address.

Objective - 1.02 - Given a sub-set of an LTM configuration, determine which objects to remove or consolidate to simplify the LTM configuration

1.02a - Interpret configuration

Overview of UNIX Configuration Files (11.x)

Interpret Configuration

This exam will test your ability to read configuration and comprehend what is configured. You will need to be familiar with the running config as well as TMSH configuration output. There are many more exhibits that show configuration as seen in the console rather than the GUI. You should build some advanced configurations by turning on settings you are likely used to seeing and then look at what that configuration looks like in a config file.

Example of a config file:

```
[root@bigipc1:Active:Disconnected] config # more bigip.conf

apm resource remote-desktop citrix-client-bundle /Common/default-citrix-client-
bundle { }
ltm default-node-monitor {
  rule none
}
ltm node /Common/10.128.20.11 {
  address 10.128.20.11
}
ltm node /Common/10.128.20.12 {
  address 10.128.20.12
}
ltm pool /Common/pool1 {
  members {
    /Common/10.128.20.11:80 {
      address 10.128.20.11
    }
    /Common/10.128.20.12:80 {
      address 10.128.20.12
    }
  }
```

```
  monitor /Common/http
}
ltm pool /Common/pool2 {
  members {
    /Common/10.128.20.11:80 {
      address 10.128.20.11
    }
    /Common/10.128.20.12:80 {
      address 10.128.20.12
    }
  }
  monitor /Common/http
}
ltm virtual /Common/test {
  destination /Common/10.128.10.100:80
  disabled
  ip-protocol tcp
  mask 255.255.255.255
  pool /Common/pool1
  profiles {
    /Common/tcp { }
  }
  translate-address enabled
  translate-port enabled
  vlans-disabled
}
ltm virtual /Common/test2 {
  destination /Common/10.128.10.100:0
  ip-protocol tcp
  mask 255.255.255.255
  pool /Common/pool2
  profiles {
    /Common/tcp { }
  }
  translate-address enabled
  translate-port disabled
  vlans-disabled
}
```

```
ltm virtual-address /Common/10.128.10.100 {
  address 10.128.10.100
  mask 255.255.255.255
  traffic-group /Common/traffic-group-1
}
ltm classification signature-version {
  version-number 0
}
net route /Common/external_default_gateway {
  gw 10.128.10.1
  network default
}
net ipsec ike-daemon /Common/ikedaemon { }
wom endpoint-discovery { }
```

Example of a list command in TMSH:

```
root@(bigipc1)(cfg-sync Disconnected)(Active)(/Common)(tmos)# list ltm virtual
ltm virtual test {
  destination 10.128.10.100:http
  disabled
  ip-protocol tcp
  mask 255.255.255.255
  pool pool1
  profiles {
    tcp { }
  }
  vlans-disabled
}
ltm virtual test2 {
  destination 10.128.10.100:any
  ip-protocol tcp
  mask 255.255.255.255
  pool pool2
  profiles {
    tcp { }
  }
  translate-port disabled
  vlans-disabled
}
```

1.02b - Identify redundant application functions

Redundant Configuration

Because there're so many different ways of configuring functionality on the BIG-IP platform, many times a configuration will be added that is redundant. You will need to have the experience of reviewing configurations both in the command line as well as the GUI Interface for redundant configurations. Many times when redundant configurations are made it is due to doing the function in an iRule as well as in the configuration of a virtual server. Other times is may be as simple as building listeners that overlap other listeners that are not necessary to do the function that is needed for an application. The list of possibilities is finite but far too great for F5 to have made a document of what to do and not do here. You will need to be very aware of configuration settings and what they do to be able to identify.

Perhaps you have a virtual server configured with a pool and also an iRule that is making a choice of multiple pools to route traffic on some criteria, and that iRule has a default pool at the end which is catching all traffic not matching the other pools criteria. You notice that the default pool on the virtual server is not getting any connections in the Statistics of the pool and realize it is not necessary since the iRule is dealing with default flows in the logic. Or you may realize that the default pool statement in the iRule is not needed since you want the pool on the virtual server to process the traffic not handled but the iRule logic.

1.02c - Describe how to consolidate redundant application functions

Consolidate Redundant Configuration Objects

On the BIG-IP platform there is usually more than one way to configure a function or method of solving a problem through settings. There are also multiple ways of implementing the system into the network. Many times the way we deploy the BIG-IP will define how we have configure setting to pass traffic correctly. The need to SNAT traffic as it flows through every virtual server can changed by deploying the system in a routed architecture. The key to this topic for the exam is being able to identify redundant configuration objects in the configuration.

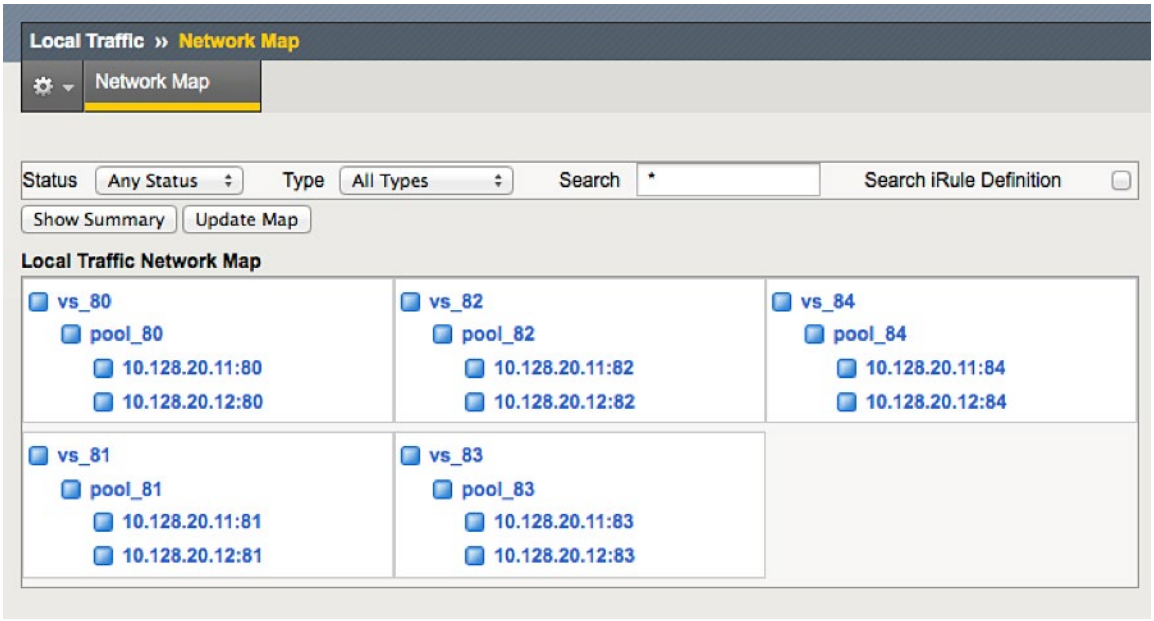
Many times the redundant functions will be between a setting in the virtual server configuration and a function within an iRule that is applied to the virtual server. The configuration may have a pool for the virtual server however it is not receiving any traffic in the statistics because in the iRule configuration is processes the traffic and if the logic falls through it is send that traffic to a different default pool.

Another Example:

If you had to load balance an application that needs to listen on five different ports, you could build five different virtual servers that are all listening on the same IP address with each using one of the five ports.

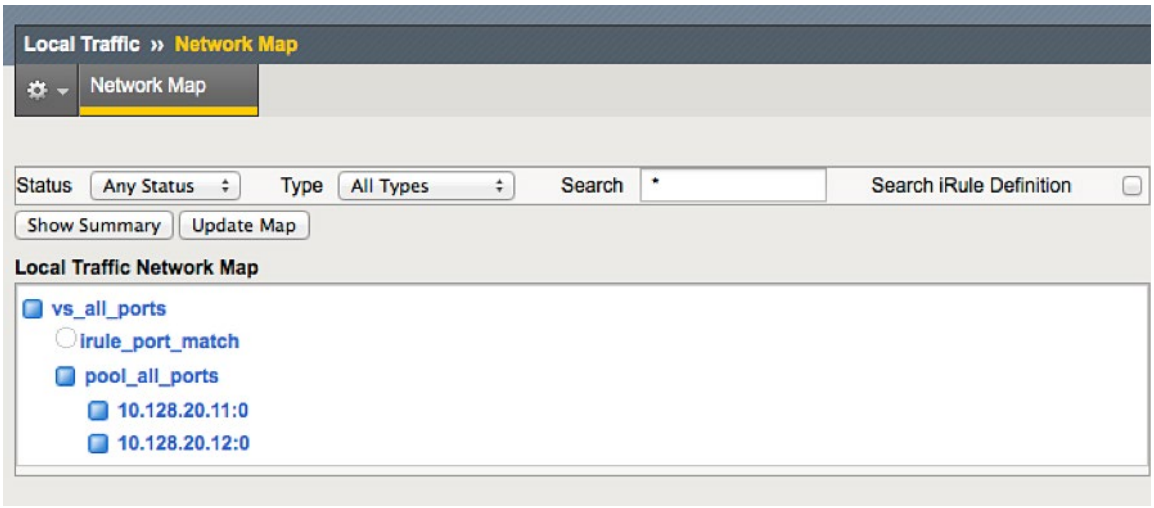
And each virtual server will use pools that each have the same two servers in them but each defined on one of the five port numbers.

The configuration would look like this:



You could accomplish the same solution of taking in the client traffic to the application on the five separate ports without building nearly as many objects. You would simply build single virtual server listening on the same IP address but on all ports with one pool that has each pool member listening on all ports. This would work the same however we will also be listening on all of the remaining ports one that virtual server's IP address. This is not as secure so we could write a quick iRule to restrict the inbound traffic to the five known ports and we are secure again.

The configuration would look like this:



This is far less configuration object and great for configuration consolidation but neither is a right or wrong way of deploying the application.

1.02d - Determine the appropriate redundant function to remove/keep

Redundant functions to remove

Once again there is no documentation on what to remove or keep in a configuration with redundant settings.

The best bit of advice I can give here is to keep the function that poses the least overhead for processing on the system. Example, if a function is being done in a virtual server's profile settings and is also being handled in an iRule, you will want to get rid of the function in the iRule and do it in the profile settings. The iRule is running as a script on top of the operating system and the profile function is a part of the OS.

1.02e - Explain the effect of removing functions from the LTM device configuration

Effect of removing redundant configuration

Many functions that can be set in the LTM configuration affect how the LTM processes the traffic for an application and will more than likely break the application for the current connections. However if we are talking about redundant functions then in most cases removing the redundant object could be harmless to

the client session to the application. The best way to get comfortable with this is to build out a working configuration and connect the virtual server and then make changes to see what will happen.

If the application is long-lived like FTP then it may be impacted by some changes like modifications to the TCP profile settings, but not others like adding a connection limit to the virtual server.

Following the previous description of an iRule that has a default pool to handle the traffic that doesn't match the decision logic of the iRule and the virtual server has a pool assigned to it that is not receiving traffic the removal of the pool that is assigned to the virtual server may not impact traffic however modifying the iRule may.

Objective - 1.03 - Given a set of LTM device statistics, determine which objects to remove or consolidate to simplify the LTM configuration

1.03a - Interpret performance statistics

Ref: 2, Chapter 8: Troubleshooting.

Performance Statistics

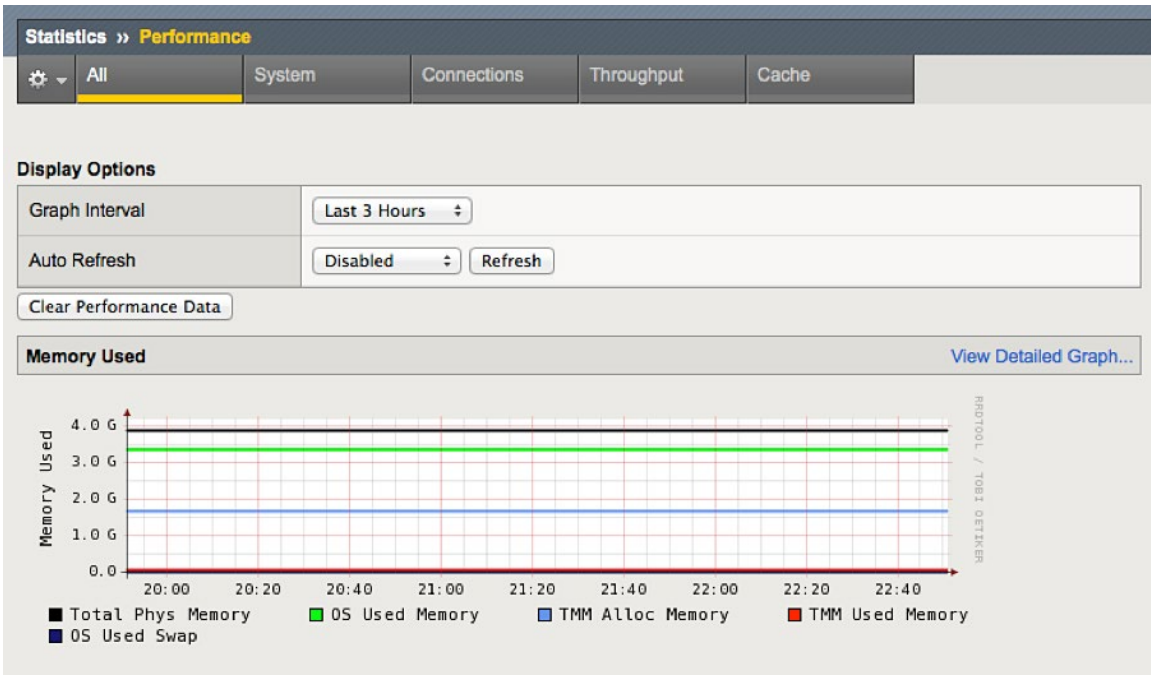
The BIG-IP platform provides statistics on platform performance so that you can easily track how the platform is running. These statistics can also provide insight into configuration that may be impacting the system. You should be comfortable with reviewing the statistics in the GUI and via the Dashboard as well as in the console.

You can view CPU usage, memory usage, connection types, throughput, HTTP Requests and SSL Transaction statistics just to name a few for the BIG-IP system.

To view Performance Statistics in the GUI:

On the Main tab of the navigation pane, expand Overview and click Performance.

The Performance screen opens.



1.03b - Identify redundant application functions

See section 1.02b

This topic content is the same as topic 1.02b – simply applied to the 1.03 objective.

1.03c - Describe how to consolidate redundant application functions

See section 1.02c

This topic content is the same as topic 1.02c – simply applied to the 1.03 objective.

1.03d - Determine the appropriate redundant function to remove/keep

See section 1.02d

This topic content is the same as topic 1.02d – simply applied to the 1.03 objective.

1.03e - Explain the effect of removing functions from the LTM device configuration

See section 1.02e

This topic content is the same as topic 1.02e – simply applied to the 1.03 objective.

Objective - 1.04 - Given a scenario, determine the appropriate upgrade and recovery steps required to restore functionality to LTM devices

Scenario Based Objective

To prepare for scenario based questions the candidate will need to complete hands-on configuration and testing of the configuration on the LTM. This will allow the candidate to better understand how different configurations can produce different results. All F5 exams use scenario-based questions that make the candidate apply what they know to a situation to determine the resulting outcome.

This topic is focused on upgrading the BIG-IP platforms for single unit, HA pairs and vCMP environments.

1.04a - Explain how to upgrade a vCMP environment

vCMP Host and Supported Guest Version Matrix
Maintaining Host Resources when Upgrading vCMP Guests
Understanding vCMP Guests

Upgrade a vCMP Environment

In a vCMP system there are two operating systems always running at a minimum, the host OS and the Guest OS. There may be multiple guests running on the host, so there could be multiple versions of guest OS running.

As an administrator you may need to upgrade the host OS to gain new features or to support a newer version for a guest OS. You do not have to run the same version of host OS as your guest's OS however there is a matrix of which host OS versions will support which guest OS version for you to follow. See link below:

SOL14088: vCMP Host and Supported Guest Version Matrix

When you initially create a vCMP guest, you choose the ISO image to install for that guest. Then, when you move the guest to the Provisioned state, the vCMP host installs that ISO image onto each of the newly-created virtual disk images pertaining to that guest.

> **Important:** The initial software image is used only when the system first creates the virtual disk images. Subsequent software upgrades are done within the guest using the live installation process.

About vCMP guest states

A vCMP guest is always in one of these states:

Configured

This is the initial (and default) state for newly-created guests. In this state, the guest is not running, and no resources are allocated to the guest. The BIG-IP system does not create virtual disks for a guest until you set that guest to the Provisioned state. If you move a guest from another state to the Configured state, the BIG-IP system does not delete the virtual disks previously attached to that guest. The guest's virtual disks persist on the system. Other resources, however, such as CPU cores, are automatically de-allocated. When the guest is in the Configured state, you cannot configure the BIG-IP modules that are licensed to run within the guest; instead, you must first provision and deploy the guest, then you can provision the BIG-IP modules within the guest.

Provisioned

When you move a vCMP guest to the Provisioned state, the system allocates resources (CPU, memory, network interfaces, and disk space) to that guest. The system also creates virtual disks for the guest and installs the selected ISO image on them. A guest does not run while in the Provisioned state.

Deployed

After provisioning a guest, you deploy it. When deploying a guest for the first time, the system installs an instance of the guest host on the guest's virtual disk. For guests in this state, the BIG-IP system attempts to start and maintain a VM on each slot for which the guest has resources allocated. If you reconfigure the properties of a guest after its initial deployment, the system immediately propagates some of those changes to all of that guest's VMs. The changes that the system immediately propagates are: Host name, cluster IP Address (including network mask and management route), and the list of allowed VLANs.

When you set up and deploy multiple guests at once, there is good reason to move each guest first to the Provisioned state. This allows you to verify that the guest allocations are satisfactory before you commit the guests to full deployment. This allows you to confirm that the virtual disk installations are successful before deploying the guests. If there is a problem with one guest's allocation or virtual disk installation, you might need to rearrange the resource allocations for your guests. Keeping the guests in the Provisioned state until you are confident in your allocations prevents you from having to shut down deployed guests to make these changes.

Specifying the ISO image for a guest

The file that the vCMP host reads to re-create the virtual disk for a guest must reside in the host's /shared/images folder, to be available as an Initial Image selection on the Guest List screen. If the file is not present, it may well be on the vCMP guest's /shared/images folder. If this is the case, you must copy that file to the proper folder on the host and wait for the host to validate it.

Before you start this task, you must have already disabled the vCMP guest so that you can edit its settings, and the necessary ISO file must be in place on the vCMP host.

When performing a BIG-IP software upgrade, you have the option of changing the ISO image version to match. Once you decide to keep that software version for processing traffic, you should change the ISO image version so that the vCMP host can recreate the virtual disk for the vCMP guest if one of the blades requires a hot swap.

1. On the Main tab, click vCMP > Guest List.
2. In the Name column, click the name of the vCMP guest that you want to modify.
3. From the Properties list, select Advanced.
4. From the Initial Image list, select the ISO image file for creating the guest's virtual disk that matches the other guests in the cluster.
5. Click Update.

Installing and upgrading the guest operating system is essentially the same as preforming the tasks on a regular BIG-IP system, which is covered in the next topic.

1.04b - Explain how to upgrade an LTM device from the GUI

BIG-IP System: Upgrading Active-Standby Systems Version 11.2

Upgrading an LTM in version 11.2

In addition to updates to the BIG-IP operating software, F5 provides additional updates for features like the IP Geolocation database, OPSWAT, and BIG-IP ASM security updates. The updates for these features are managed separately from the core BIG-IP updates. F5 recommends that administrators keep these features updated to ensure your BIG-IP system has current information and protection against the latest security threats.

> **Note:** You can only perform major software upgrades if the BIGIP system is entitled under a current technical support contract.

Installation overview

This document covers very basic steps for installing the software. You can find complete, step-by-step installation and upgrade instructions in this section's link BIG-IP Systems: Upgrading Active-Standby Systems, and it is strongly recommend that you reference these documents to ensure successful completion of the installation process.

Find the latest software

Release notes for the version of software you want to install will contain instructions for the specific installation.

To find the latest software version for your F5 product

1. Go to the F5 Downloads page (downloads.f5.com).
2. Click Find a Download.
3. Find the product you want to download and click the link for the appropriate version.
4. Find and click the link for the update you want to download.
5. Read and accept the End User Software license agreement.
6. Click the file name, choose a download location, and then save the file to your computer.

Installing the software

You can install the software at the command line using the Traffic Management shell, TMSH, or in the browser-based Configuration utility using the Software Management screens, available in the System menu. Choose the installation method that best suits your environment.

Installation Method	Command
Install to existing volume, migrate source configuration to destination	tmsh install sys software image [image name] volume [volume name]
Install from the browser-based Configuration utility	Use the Software Management screens in a web browser.

Sample installation from the browser-based Configuration utility

To find the latest software version for your F5 product

1. Expand System, and click Software Management.
2. On the Image List Tab, Click Import

3. Click Choose File and navigate to your downloaded image and select Choose.
4. Click Import

Your Uploaded ISO should not be in the Available Images List

5. Select the image and click Install
6. In the Install Software Image box choose a volume and click Next.

   Note: You cannot install to the volume you are currently running in and that volume will not be available in the list.

7. Install will complete, visible in the Install Status field.

Boot to the newly installed version

To boot the BIG-IP into the new software follow these steps:

1. Expand System, and click Software Management > Boot Locations.
2. Click on the inactive Boot Location that you installed the new OS onto.
3. Click Activate

The system will reboot into that drive volume.

1.04c - Describe the effect of performing an upgrade in an environment with device groups and traffic groups

About Traffic Groups

Upgrades

BIG-IP redundant system configuration is based on a few key components.

Devices

A device is a physical or virtual BIG-IP system, as well as a member of a local trust domain and a device group. Each device member has a set of unique identification properties that the BIG-IP system generates.

Device groups

A device group is a collection of BIG-IP devices that trust each other and can synchronize, and sometimes fail over, their BIG-IP configuration data.

Important: To configure redundancy on a device, you do not need to explicitly specify that you want the BIG-IP device to be part of a redundant configuration. Instead, this occurs automatically when you add the device to an existing device group.

You can create two types of devices groups:

Sync-Failover

A Sync-Failover device group contains devices that synchronize configuration data and support traffic groups for failover purposes when a device becomes unavailable.

Sync-Only

A Sync-Only device group contains devices that synchronize configuration data, such as policy data, but do not synchronize failover objects.

A BIG-IP device can be a member of only one Sync-Failover group. However, a device can be a member of both a Sync-Failover device group and a Sync-Only device group.

To minimize issues with config sync, failover, or mirroring, F5 Networks recommends as a best practice that devices in a device group match as closely as possible with respect to hardware platform, product licensing, and module provisioning. At a minimum, mirroring requires that the hardware platforms of the mirrored devices match, and config sync between devices requires that the devices are running the same version of BIG-IP system software.

Traffic groups

A traffic group is a collection of related configuration objects (such as a virtual IP address and a self IP address) that run on a BIG-IP device and process a particular type of application traffic. When a BIG-IP device becomes unavailable, a traffic group can float to another device in a device group to ensure that application traffic continues to be processed with little to no interruption in service.

Device trust and trust domains

Underlying successful operation of device groups and traffic groups is a feature known as device trust. Device trust establishes trust relationships between BIG-IP devices on the network, through mutual certificate-based authentication. A trust domain is a collection of BIG-IP devices that trust one another and can therefore synchronize and fail over their BIG-IP configuration data, as well as exchange status and failover messages on a regular basis. A local trust domain is a trust domain that includes the local device, that is, the device you are currently logged in to.

Folders and sub folders

Folders and sub-folders are containers for the configuration objects on a BIG-IP device. For every administrative partition on the BIG-IP system, there is a high-level folder. At the highest level of the folder hierarchy is a folder named root. The BIG-IP system uses folders to affect the level of granularity to which it

synchronizes configuration data to other devices in the device group. You can create sub-folders within a high-level folder, using TMSH.

> **Note**: In most cases, you can manage redundancy for all device group members remotely from one specific member. However, there are cases when you must log in locally to a device group member to perform a task. An example is when resetting device trust on a device.

About traffic groups

A traffic group is a collection of related configuration objects that run on a BIG-IP device. Together, these objects process a particular type of traffic on that device. When a BIG-IP device becomes unavailable, a traffic group floats (that is, fails over) to another device in a device group to ensure that application traffic continues to be processed with little to no interruption in service. In general, a traffic group ensures that when a device becomes unavailable, all of the failover objects in the traffic group fail over to any one of the devices in the device group, based on the number of active traffic groups on each device.

A traffic group is initially active on the device on which you create it, until the traffic group fails over to another device. For example, if you initially create three traffic groups on Device A, these traffic groups remain active on Device A until one or more traffic groups fail over to another device. If you want to balance the traffic group load among all devices in the device group, you can intentionally cause a traffic group to fail over to another device. You do this using the Force to Standby option of the Configuration utility.

> **Important:** Although a specific traffic group can be active on only one device in a device group, the traffic group actually resides and is in a standby state on all other device group members, due to configuration synchronization.

Only certain types of configuration objects can belong to a traffic group. Examples of traffic group objects are self IP addresses and virtual IP addresses.

An example of a set of objects in a traffic group is an iApps application service. If a device with this traffic group is a member of a device group, and the device becomes unavailable, the traffic group floats to another member of the device group, and that member becomes the device that processes the application traffic.

When a traffic group fails over to another device in the device group, the device that the system selects is normally the device with the least number of active traffic groups. When you initially create the traffic group on a device, however, you specify the device in the group that you prefer that traffic group to run on in the event that the available devices have an equal number of active traffic groups (that is, no device has fewer active traffic groups than another). Note that, in general, the system considers the most available device in a device group to be the device that contains the fewest active traffic groups at any given time.

> **Note:** A Sync-Failover device group can support a maximum of 15 traffic groups.

1.04d - Explain how to perform an upgrade in a high availability group

Preparing BIG-IP Active-Standby Systems for an Upgrade

Upgrading in an HA Environment

Preparing BIG-IP active-standby systems for an upgrade

The following prerequisites apply when you upgrade BIG-IP active and standby devices from version 10.x to 11.x.

- The BIG-IP systems (Device A and Device B) are configured as an active-standby pair.
- Each BIG-IP device is running the same version of 10.x software.
- The BIG-IP active-standby devices are the same model of hardware.

When you upgrade a BIG-IP active-standby pair from version 10.x to 11.x, you begin by preparing the devices.

> **Note:** If you prefer to closely observe the upgrade of each device, you can optionally connect to the serial console port of the device that you are upgrading.

1. For each device, complete the following steps to prepare the configuration and settings.
   a. Examine the Release Notes for specific configuration requirements, and reconfigure the systems, as necessary. For example, you must reconfigure version 10.x symmetric WebAccelerator modules as asymmetric systems before upgrading to version 11.x.
   b. Examine the Release Notes for specific changes to settings that occur when upgrading from version 10.x to 11.x, and complete any in-process settings. For example, you must publish any unpublished BIG-IP WebAccelerator module policies in order for them to migrate to version 11.x.
2. From the device that is running the latest configuration, synchronize the configuration to the peer unit.
   a. On the Main menu, click System > High Availability > ConfigSync. A message appears for the Status Message.
   b. Click Synchronize TO Peer.
3. For each device, reactivate the license.
   a. On the Main menu, click System > License.
   b. Click Re-activate.

c. In the Activation Method area, select the Automatic (requires outbound connectivity) option.

d. Click Next. The BIG-IP software license renews automatically.

4. For each device, click System > High Availability > Redundancy, and, from the Redundancy State Preference list, select None.

5. For each device, create a backup file.

   a. Access the TMSH command line utility.

   b. At the prompt, type save /sys ucs /shared/filename.ucs.

   c. Copy the backup file to a safe location on your network.

6. Download the BIG-IP version 11.x .iso file from the AskF5 downloads web site (https://www.downloads.f5.com) to a preferred location.

7. Using a tool or utility that computes an md5 checksum, verify the integrity of the BIG-IP version 11.x .iso file.

8. Import the version 11.x software image file to each device.

   a. On the Main menu, click System > Software Management > Image List > Import.

   b. Click Choose File, locate and click the image file, click Open, and click Import.

   c. When the software image file completes uploading to the BIG-IP device, click OK. A link to the image file, but not to the .md5 file, appears in the Software Image list.

The BIG-IP devices are prepared to install the version 11.x software onto Device B (the standby BIG-IP 2 device).

Upgrading the standby BIG-IP 2 system

The following prerequisites apply for this task.

- Device A (the active BIG-IP 1 system) and Device B (the standby BIG-IP 2 system) must be prepared to upgrade Device B with version 11.x software.
- The version 11.x software image and MD5 files are downloaded and available.

After you prepare Device A (the active BIG-IP 1 system) and Device B (the standby BIG-IP 2 system) for upgrading the software, you can perform these steps to install the version 11.x software onto Device B.

1. On the Main menu, click System > Software Management > Image List.
2. In the Available Images area, select the check box for the version 11.x software image.
3. Select a location to install the image, and click Install.

   Important: In the Install Status list for the specified location, a progress bar indicates the status of the installation. Ensure that installation successfully completes, as indicated by the progress bar, before proceeding.
4. Reboot the device to the location of the installed version 11.x software image.

   a. On the Main menu, click System > Software Management > Boot Locations.

   b. In the Boot Location list, click the boot location of the installed version 11.x software image.

   c. Click Activate. The BIG-IP device reboots to the version 11.x boot location with traffic-group-1 in standby state.

   Note: If the device appears to be taking a long time to reboot, do not cycle the power off and on. Instead, verify the status of the device by connecting to its serial console port. The device might be performing firmware upgrades.

Version 11.x software is installed on Device B, with traffic-group-1 in standby state.

Upgrading the active BIG-IP 1 system

The following prerequisites apply in upgrading Device A (the BIG-IP 1 system).

- Device A (the version 10.x BIG-IP 1 system) must be prepared to upgrade the software to version 11.x.
- Device A is in active mode.
- Device B (the version 11.x BIG-IP device with traffic-group-1) is in standby state.

After you prepare Device A (the standby BIG-IP 1 system) for upgrading the software, you can perform these steps to upgrade the software to version 11.x.

1. On the Main menu, click System > Software Management > Image List.
2. In the Available Images area, select the check box for the version 11.x software image.
3. Select a location to install the image, and click Install.

Important: In the Install Status list for the specified location, a progress bar indicates the status of the installation. Ensure that installation successfully completes, as indicated by the progress bar, before proceeding.

4. Force the BIG-IP device (Device A) to standby mode.

   a. On the Main menu, click System > High Availability > Redundancy

   b. Click Force to Standby The BIG-IP device (Device A) changes to standby mode and the peer BIG-IP device (Device B) changes to active state.

   Important: Once the peer BIG-IP device (Device B) changes to active state, ensure that it passes traffic normally.

5. Reboot the BIG-IP device (Device A) to the location of the installed version 11.x software image.

   a. On the Main menu, click System > Software Management > Boot Locations.

   b. In the Boot Location list, click the boot location of the installed version 11.x software image.

   c. Click Activate. The BIG-IP device (Device A) reboots to the version 11.x boot location with traffic-group-1 in standby state.

   Note: If the device appears to be taking a long time to reboot, do not cycle the power off and on. Instead, verify the status of the device by connecting to its serial console port. The device might be performing firmware upgrades.

6. Synchronize the configuration.

   a. On the Main tab, click Device Management > Device Groups

   b. Click the name of a device group.

   c. On the menu bar, click Config Sync. The Config Sync screen appears, displaying the status for each member.

   d. As indicated by the Status Message, click one of the following buttons.

      - Synchronize TO Group
      - Synchronize FROM Group

Version 11.x software is installed on Device A (the BIG-IP system with traffic-group-1 in standby state).

Verifying a BIG-IP active-standby upgrade

When you have completed upgrading the BIG-IP active-standby pair from version 10.x to version 11.x, you should verify that the upgraded configuration is working properly. Perform the following steps to verify the version 11.x upgrade.

1. Verify the Platform configuration for each device.
    a. On the Main menu, click System > Platform.
    b. From the Root Folder Device Group setting, verify that the device group is identical on the pair of devices.
    c. From the Root Folder Traffic Group list, verify that the correct traffic group (traffic-group-1) is selected.
2. Verify the configuration for each device.
    a. On the Main menu, click Device Management > Devices.
    b. Verify the information for the device and the peer device.
    c. On the Main menu, click Device Management > Device Trust.
    d. In the Peer Authority Devices area, verify that the peer device is specified.

    Note: Ensure that all information for the peer device appears correctly and complete.
3. Verify the traffic groups for each device.
    a. On the Main menu, click Network > Traffic Groups.
    b. Click traffic-group-1.
    c. If you configured MAC Masquerade addresses for VLANs on the version 10.x devices, verify that the traffic-group-1 includes an address in the MAC Masquerade Address field.
    d. Verify that the floating traffic group is correct.
    e. Verify that the failover objects are correct.
4. Verify the Current ConfigSync State for each device.
    a. On the Main menu, click Device Management > Devices.
    b. In the Device List, in the Status column, verify that each device shows a sync status of green.

1.04e - Describe the process for using SCF and UCS files

Overview of Single Configuration Files (11.x)

Working with Single Configuration Files

A single configuration file (SCF) is a flat, text file that contains a series of Traffic Management Shell (TMSH) commands and the attributes and values of those commands that reflect the configuration of the BIG-IP system.

Using the SCF feature, a user can define all BIG-IP configurations in a single, flat file, using the TMSH utility. The SCF feature allows you to use an SCF file from one BIG-IP system to configure additional BIG-IP systems, and it is useful when you want to configure a new BIG-IP system. An SCF may be useful in one or more of the following circumstances:

- You want to replicate a configuration across multiple BIG-IP systems.
- You want to migrate a configuration from one platform type to another (for example, VE to a hardware platform).

General guidelines for using SCF files are as follows:

- Build a BIG-IP LTM template configuration by configuring a system using the Configuration utility or TMSH utility. Most custom Linux configurations must be completed using the TMSH utility.
- Save an SCF file from the fully-configured system using the TMSH save /sys command, and store the SCF file in a safe place for future use. The SCF file can be used as a template to configure future BIG-IP systems.
- When you are ready to use the SCF file to configure a new BIG-IP system, copy the SCF file to the new BIG-IP system, and edit the SCF file prior to importing it. For example, change the IP addresses, routing information, interface settings and other common settings, as needed.
- Install the SCF file into a new system using the TMSH load /sys command.

TMSH commands for single configuration files

You use the TMSH utility to perform the basic management of a single configuration file (SCF). This table contains an overview of the commands to accomplish this.

TMSH Command	Usage
`save sys config file [filename]`	Use this command to save a copy of the currently running configuration to an SCF. It is important to note that saving a configuration to an SCF does not affect the running or stored configuration of the BIG-IP system on which you run the command.
`load sys config file [filename]`	Use this command to replace or restore an SCF with a saved configuration. When you use this command, the system saves any previously running configuration to the directory /var/local/scf/, by default.
`load sys config default`	Use this command to restore the factory default settings of the configuration file, while retaining the management IP address and the administrator user name and password.

You can perform three main tasks with respect to single configuration files.

Creating and saving an SCF

Use this procedure to create and save a single configuration file.

1. Access the TMSH utility.
2. Use the following syntax: `save sys config file [filename]`. If you include the .scf extension in the file name, the system does not add an additional file extension. If you create an SCF file twice (on two different occasions), you can compare the contents of the two files.

This procedure causes the TMSH utility to gather all of the commands (and their attributes and values) that compose the running configuration. Once gathered, the system saves the configuration to a flat file with the name you specify and the extension of .scf. By default, the system stores this file in the /var/local/scf directory, but you can specify a different path if you prefer.

Loading an SCF onto a target BIG-IP system

The primary benefit of the SCF feature is that it gives you the ability to create a configuration on one BIG-IP system that you can load onto other BIG-IP systems (hereafter referred to as the target BIG-IP system), rather than having to recreate the configuration multiple times.

After you have created and saved the SCF using the save sys config file [filename] command, you can modify any data unique to the specific target BIG-IP system, then load the configuration on that system.

> **Note:** To successfully load a configuration you have replicated, ensure that no line of the configuration is longer than 4096 characters. If there are more than 4096 characters in a single line, the system reverts to the previous running configuration.

1. On the target BIG-IP system, load the saved SCF file by typing the following command: tmsh load sys config file [filename] The tmsh utility first saves the system's stored configuration in a backup file (named /var/local/scf/backup.scf), and then uses the configuration stored in the SCF that you are loading.
2. Use a text editor to open the SCF and edit any data that is unique to the target BIG-IP system, such as the management IP address.
3. Save the SCF to the target BIG-IP system by typing the following command: sys save config file [filename] If a backup SCF already exists, the TMSH utility appends a number to the name of the existing backup file, and then creates a new backup file. Thus:
   - The first time the system backs up the running configuration during a load operation, the system names the backup file /var/local/scf/backup.scf.
   - The next time the system backs up the running configuration, the system renames the file from /var/local/scf/backup.scf to /var/local/scf/backup-1.scf and creates a new file named /var/local/scf/backup.scf.
   - If you run the load command a third time, the system renames the file from /var/local/scf/backup-1.scf to /var/local/scf/backup-2.scf, renames the file /var/local/scf/backup.scf to /var/local/scf/backup-1.scf, and once again creates a new file named /var/local/scf/backup.scf.

Using an SCF to restore a BIG-IP system configuration

You can use an SCF to restore a BIG-IP system configuration. The BIG-IP system ships with a default SCF. Depending on whether you want to restore the factory default configuration or load a specific configuration, perform this step.

1. From a console window, access the tmsh prompt.
2. Choose one of these options.

Option	Description
Restore a system to the factory default configuration	Type the command tmsh load sys config default. This command retains the management IP and the assigned root and administrator passwords. When you use this command, the system first saves the running configuration in the backup.scf file and then resets the local traffic management and the operating system configuration to the factory default settings by loading the SCF, /defaults/defaults.scf.
Restore a system with values defined in the specified SCF	Type the command tmsh load sys config file [filename]. When you use this command, the system first saves the running configuration in the backup.scf file, and then resets the running configuration to the values contained in the specified SCF. You must run the save sys config partitions all command to save the running configuration in the stored configuration files.

Backing Up Your BIG-IP System Configuration

Working with User Configuration Sets

A user configuration set (UCS) is an archive of the configuration files that the BIG-IP system requires to restore the system. The UCS archive, by default, contains all of the files that are required to restore your current configuration to a new system, including configuration files, the product license, local user accounts, and Secure Socket Layer (SSL) certificate/key pairs.

Prior to BIG-IP 11.0.0, the BIG-IP system restored the configuration either in full or only partially, depending on the host name of the unit on which you installed the UCS configuration archive.

If the host name of the unit matches the host name stored in the UCS archive, the BIG-IP system restores the full configuration, including self-IP addresses and VLANs. If the host name of the unit does not match the host name stored in the UCS archive, the BIG-IP system restores only the shared configuration, such as virtual servers, pools, and profiles.

Beginning in BIG-IP 11.0.0, when installing a UCS configuration archive, the BIG-IP system restores the full configuration.

Due to an architectural change in BIG-IP 11.0.0, the UCS configuration archive is intended for complete configuration backup and restore usage only.

File names and location

By default, the BIG-IP system saves the UCS archive file with an .ucs extension if you do not include it in the file name. You can also specify a full path to the archive file, and then the archive file is saved to the specified location. If you do not include a path, the file is saved to the default archive directory, /var/local/ucs. Archives located in a directory other than the default do not appear in the list of available archives when using the Configuration utility to create or restore a UCS archive, or when using the list /sys ucs command in the TMSH utility. To easily identify the file, F5 recommends that you include the BIG-IP host name and current time stamp as part of the file name.

Secure Storage

Ensure that you have access to a secure location for storage of your UCS archive files. A typical UCS archive contains user accounts, passwords, critical system files, and SSL private keys. However, you can explicitly exclude SSL private keys from a UCS archive during the backup process. It is important to store the backup UCS archives containing sensitive information in a secure location.

Considerations for restoring configuration data

BIG-IP software version and platform

- F5 recommends that you run the same version of the BIG-IP software on the BIG-IP system from which it was backed up. However, you can restore a BIG-IP 10.x UCS archive on a system running BIG-IP 11.x software.
- Due to an issue in BIG-IP 11.0.0, you must perform a configuration restoration using a configuration archive that is taken from the same hardware platform. For more information, refer to SOL13136: The UCS configuration archive cannot be restored on a platform other than the one on which the archive was created.
- The UCS archive is intended to back up and restore the configuration of a specific platform. When installing a UCS archive on a dissimilar platform, the configuration may fail to load due to the differing hardware components. These failures require that you intervene manually, and identify and resolve each error that the system presents when you attempt to load the configuration.

Licensing

The BIG-IP license is associated with a specific hardware serial number. The UCS archive contains the license file of the system where the configuration was saved. To successfully install a UCS archive file on a BIG-IP system, you must perform one of the following actions:

- Restore the UCS archive to the same system from which it was saved.
- Have the license associated with the serial number of a new system. To do so, contact F5 Technical Support.

  Note: F5 Technical Support will associate a license file with a new serial number only on an as-needed basis, in the event of a Return Materials Authorization (RMA).
- Relicense the BIG-IP system after restoring the UCS archive.
- Save the license file prior to restoring the configuration from another system, and then copy the license file back.
- Install the UCS archive by using the tmsh no-license option. For the command syntax, refer to the following example:

  ```
  tmsh load sys ucs [ucs file name] no-license
  ```

  Important: If you use a different license than the one contained in a restored UCS archive, the replacement license must include authorization for the same options and add-on modules, such as BIG-IP WebAccelerator or BIG-IP ASM. If you attempt to restore a UCS configuration referencing an

unlicensed module, the BIG-IP system does not properly restore the UCS archive. Additionally, the BIG-IP system reports a Provisioning Warning message in the Configuration utility, as well as the status of ModuleNotLicensed in its command-line prompt.

UCS files

If necessary, copy the UCS archive file you want to restore to the BIG-IP file system.

Host name and base configuration

The UCS restore operation restores the full configuration to the target system, including the host name and the base configuration.

Note: This behavior has changed from previous versions of the BIG-IP system as mentioned in the beginning of this section.

BIG-IP configuration objects with passphrases

If you are restoring on a new system, a UCS archive that includes encrypted passphrases cannot be decrypted by the new system. This format is an intentional security measure.

When replacing one system of a failover pair, F5 recommends that you configure basic networking on the replacement unit and synchronize the configuration from its peer instead of restoring the configuration by installing the UCS archive. Because the master key is shared between units of a redundant pair, the configuration synchronization process synchronizes the original master key to the newly installed device. If you cannot synchronize the original master key to the new system from its peer, but you know the original unencrypted passphrases, you can install the UCS file to restore the configuration, modify the affected configuration objects to replace the encrypted passphrases with unencrypted versions, and save the resulting configuration.

If you are restoring a backup that contains encrypted passphrases after reinstalling the operating system, replacing a failed system with a new system, or otherwise moving an existing configuration to a new system, the encrypted passphrases for objects used in the configuration cannot be decrypted. An error message similar to the following example appears:

```
0107102b:3: Master Key decrypt failure - decrypt failure - final
```

Backing up your BIG-IP system configuration

Impact of procedure: Performing the following procedure should not have a negative impact on your system.

1. Log in to the Configuration utility.
2. Navigate to System > Archives.
3. To initiate the process of creating a new UCS archive, click Create.
4. In the File Name box, type a name for the file.

   Important: You must use a unique file name. If a file by the same name already exists, the UCS archive file is not created, and the system displays a warning message that appears similar to the following example:

   The file already exists on the system
5. Optional: If you want to encrypt the UCS archive file, from the Encryption menu, select Enabled and enter a passphrase. You must supply the passphrase to restore the encrypted UCS archive file.
6. Optional: If you want to exclude SSL private keys from the UCS archive, from the Private Keys menu, select Exclude.
7. To create the UCS archive file, click Finished.
8. When the backup process is done, examine the status page for any reported errors before proceeding to the next step.
9. To return to the Archive List page, click OK.
10. Copy the .ucs file to another system.

Backing up configuration data by using the TMSH utility

Impact of procedure: Performing the following procedure should not have a negative impact on your system.

1. Log in to the TMSH utility by typing the following command:
   ```
   tmsh
   ```
2. Create the UCS archive file by using the following command syntax. Replace <path/to/UCS> with the full path to the UCS archive file:
   ```
   save /sys ucs <path/to/UCS>
   ```
   For example:
   ```
   save /sys ucs /var/tmp/MyUCS.ucs
   ```

3. Optional: If you want to encrypt the UCS archive with a passphrase, use the following command syntax. Replace <path/to/UCS> with the full path to the UCS archive file, and replace <password> with the passphrase you want to use to encrypt the UCS archive:

   ```
   save /sys ucs <path/to/UCS> passphrase <password>
   ```

   For example:

   ```
   save /sys ucs /var/tmp/MyUCS.ucs passphrase password
   ```

4. Optional: If you want to exclude SSL private keys from the UCS archive, use the following command syntax. Replace <path/to/UCS> with the full path to the UCS archive file.

   ```
   save /sys ucs <path/to/UCS> no-private-key
   ```

   For example:

   ```
   save /sys ucs /var/tmp/MyUCS.ucs no-private-key
   ```

5. Copy the .ucs file to another system

Restoring configuration data by using the Configuration utility

Impact of procedure: The BIG-IP system replaces any existing configuration with the UCS archive file configuration.

To restore a configuration in a UCS archive by using the Configuration utility, review the considerations described in the Considerations for restoring configuration data section of this article before performing the following procedure:

1. Log in to the Configuration utility.
2. Navigate to System > Archives.
3. Click the name of the UCS archive you want to restore.
4. If the UCS archive is encrypted, type the passphrase for the encrypted UCS archive file in the Restore Passphrase field. If the UCS archive is not encrypted, you can skip this step.
5. To initiate the UCS archive restore process, click Restore.
6. When the restore process is completed, examine the status page for any reported errors before proceeding to the next step.
7. To return to the Archive List page, click OK.

8. If you restored the UCS archive on a different device and received the errors noted in the Considerations for restoring configuration data section of this article, you must reactivate the BIG-IP system license.

9. After relicensing the system, restart the system to ensure that the configuration is fully loaded. To restart the system, navigate to System > Configuration, and then click Reboot.

10. If the system you restored contains the FIPS 140 HSM, you must configure the FIPS 140 HSM Security World after completing steps 1 through 9.

Restoring configuration data from the command line by using the TMSH utility

Impact of procedure: The BIG-IP system replaces any existing configuration with the UCS archive file configuration.

1. Log in to the TMSH utility by typing the following command:
   ```
   tmsh
   ```

2. Restore the UCS archive file by using the following command syntax. Replace <path/to/UCS> with the full path of the UCS archive file you want to restore:
   ```
   load /sys ucs <path/to/UCS>
   ```

   If you do not specify the path, the BIG-IP system performs as if the UCS archive file is located in the default /var/local/ucs directory.

3. If the UCS archive file was encrypted with a passphrase during the backup, you are prompted to enter the passphrase for the archive file.

4. If you are running BIG-IP on a 6400, 6800, 8400, or 8800 hardware platform, type the following command to switch to the bash shell:
   ```
   run /util bash
   ```

5. Type the following command to verify that the new or replaced secure shell (SSH) keys from the UCS file are synchronized between the BIG-IP system and the Switch Card Control Processor (SCCP):
   ```
   keyswap.sh sccp
   ```

6. Type the following command to switch back to tmsh:
   ```
   exit
   ```

7. Restart the system by typing the following command:
   ```
   reboot
   ```

If you installed the UCS archive on the same device on which the backup was created, the system loads the restored configuration after the restarting. If you restored the backup on a different device and received the first error noted in the Considerations for restoring configuration data section of this article, you must reactivate the BIG-IP system license. Alternatively, you can replace the /config/bigip.license file with the original bigip.license file that you backed up from the target system.

8. If the system you restored contains the FIPS 140 HSM, you must configure the FIPS 140 HSM Security World after completing steps 1 through 5.

Objective - 1.05 - Given a scenario, determine the appropriate upgrade steps required to minimize application outages

Scenario Based Objective

To prepare for scenario based questions the candidate will need to complete hands-on configuration and testing of the configuration on the LTM. This will allow the candidate to better understand how different configurations can produce different results. All F5 exams use scenario-based questions that make the candidate apply what they know to a situation to determine the resulting outcome.

This topic is focused on upgrading the BIG-IP platforms to minimize application outages.

1.05a - Explain how to upgrade a vCMP environment

See section 1.04a

This topic content is the same as topic 1.04a – simply applied to the 1.05 objective.

1.05b - Explain how to upgrade an LTM device from the GUI

See section 1.04b

This topic content is the same as topic 1.04b – simply applied to the 1.05 objective.

1.05c - Describe the effect of performing an upgrade in an environment with device groups and traffic groups

See section 1.04c

This topic content is the same as topic 1.04c – simply applied to the 1.05 objective.

1.05d - Explain how to perform an upgrade in a high availability group

See section 1.04d

This topic content is the same as topic 1.04d – simply applied to the 1.05 objective.

1.05e - Describe the process for using SCF and UCS files

See section 1.04e

This topic content is the same as topic 1.04e – simply applied to the 1.05 objective.

Objective - 1.06 - Describe the benefits of custom alerting within an LTM environment

1.06a - Describe Enterprise Manager, its uses, and alerting capabilities

Overview of Monitoring and Alerting Rates

Enterprise Manager

Enterprise Manager is an appliance that helps you streamline the administrative tasks associated with managing multiple network devices. These administrative tasks include: performance monitoring, software installation and upgrades, configuration archival and restoration, certificate monitoring, security policy management, software image storage, and user account management. Enterprise Manager works in many types of network topologies, including those in multi-tiered configurations containing multiple firewalls.

You can use Enterprise Manager to manage networks with devices running the following software.

- BIG-IP system version 9.3 and later
- BIG-IP Local Traffic Manager Virtual Edition (VE) version 10.2 and later
- BIG-IP Secure Access Manager version 8.0 and later
- WANJet version 5.0 and later
- Enterprise Manager version 1.0 and later

Enterprise Manager Alerting

The Enterprise Manager system generates two general categories of alerts:

- Enterprise Manager alerts include all alerts concerning local conditions on the Enterprise Manager system
- Device alerts include all alerts concerning conditions on remote BIG-IP systems that are monitored by the Enterprise Manager system

Each alert is based on a specific alert condition, which is monitored by the Enterprise Manager. Some monitoring tasks are automatically scheduled, and the system administrator schedules some tasks. Tasks may be scheduled to run once, or on a recurring basis. Once an alert condition is detected, the Enterprise Manager may generate a single alert, or an alert at repeated intervals until the alert condition has cleared. Alerts are logged to the /var/log/em file, and optionally to an email address or a syslog server, depending on configuration.

For example, enabling the default Statistics Data Storage Utilization Enterprise Manager alert causes the Enterprise Manager to check the storage utilization every 10 minutes, and the system will fire an alert when the alert condition is first detected. The system will continue to check the utilization every 10 minutes, as scheduled. If the alert condition persists, the system will generate subsequent alerts every 60 minutes thereafter. Once the alert condition has cleared, the system continues to monitor the alert condition every 10 minutes, and no further alerts are generated unless the alert condition occurs again.

The following tables include the default monitor and alert intervals for Enterprise Manager alerts and Device alerts:

Enterprise Manager alerts (local events)

Alert	Monitoring interval	Alerting interval
RAID Drive Failure *	30 seconds	Once
RAID Drive Rebuild Complete *	30 seconds	Once
Attack Signature Update(s) Available	Scheduled task	Once
Average CPU Usage	10 minutes	10 minutes
Calculated Statistics Data Storage Capacity	10 minutes	60 minutes
Disk Usage	10 minutes	10 minutes
EM Config Sync Failure	Scheduled/User-init task	Once
Memory Usage	10 minutes	10 minutes
Statistics Data Backup Failure	Scheduled/User-init task	Once
Statistics Data Collection Rate Capacity	5 minutes	60 minutes
Statistics Data Storage Utilization	10 minutes	60 minutes

* EM-3000 platform only

Device alerts (remote events)

Alert	Monitoring interval	Alerting interval
Attack Signature Install Completion	User-initiated task	Once
Attack Signature Install Failure	User-initiated task	Once
Hotfix Install Completion	User-initiated task	Once
Hotfix Install Failure	User-initiated task	Once
Software Upgrade Completion	User-initiated task	Once
Software Upgrade Failure	User-initiated task	Once
Rotating Archive Create Failure	User-initiated task	Once
Certificate Expiration	60 minutes	60 minutes
Device Status Change	45 seconds	Once
Device Status Change (Impaired) *	45 seconds	60 minutes
Clock Skew from EM	45 seconds	60 minutes
Config Sync **	45 seconds	60 minutes
Statistics Data Thresholds	5 minutes	60 minutes

* Enterprise Manager 2.0.0 and later only

** Due to a known issue, the Enterprise Manager 2.3.0 and later may fail to generate a ConfigSync alert for a managed device.

1.06b - Describe AVR alerting capabilities

Overview of Monitoring and Alerting Rates

AVR Alerting

Before you can configure the system to send alerts concerning statistics, you need to have created an Analytics profile to collect application statistics locally (Statistics Logging Type must have Internal selected).

You can configure the BIG-IP system to send alerts concerning local application statistics based on threshold values that you set. The system sends notifications when threshold values are breached, and when they return to normal. Therefore, it is a good idea to get familiar with the typical statistics for the web application before attempting to set up alerts and notifications. When you understand the typical values, you can configure the system to alert you of limiting system situations, such as system overload.

There are 4 means of logging alerts for AVR. You can log to the local BIG-IP syslog, a remote syslog, SNMP and email alerts. If you choose to use syslog the alerts are logged in the /var/log/LTM file or System > Logs > Local Traffic in the GUI. To use remote syslog you must first configure the remote syslog server on the BIG-IP system and then choose to log to syslog just like you did with local syslog. To send AVR alerts to an external SNMP receiver you need to configure SNMP and select SNMP. To send email alerts, you need to configure the BIG-IP system to communicate with a mail server and then select E-Mail.

> **Note:** End user response times and latencies can vary significantly based on geography and connection types, which makes it difficult to set an accurate alerting threshold for page load times.

Objective - 1.07 - Describe how to set up custom alerting for an LTM device

1.07a - List and describe custom alerts: SNMP, email and Remote Syslog Identify the location of custom alert configuration files

SNMP Agent Configuration
Configuring the BIG-IP System to Deliver Locally-Generated Email Messages (11.x)
Logging

Custom Alerts

Configuring custom alerts are important to making the daily administration and support of a BIG-IP platform a manageable task. Setting custom alerts allows the administrator to only see the alerts that they have determined important to the management of the environment. If these custom filtered alert events are only

recorded on the local system and no one is actively checking the system, they cannot bring any value to the administrator of the environment. Most datacenter environments will have a centralized alert tracking system. This could be a SNMP server where all SNMP traps are sent giving a network operations center (NOC) a single pane of glass to see all events within the environment. It may be a remote syslog server for logging all events or in some environments they have the systems send email alerts directly to support team accounts so the supporting team can respond directly.

SNMP

You can use the industry-standard SNMP protocol to manage BIG-IP devices on a network. To do this, you must configure the SNMP agent on the BIG-IP system. The primary tasks in configuring the SNMP agent are configuring client access to the SNMP agent, and controlling access to SNMP data.

To better control access to SNMP data, you can assign an access level to an SNMP v1 or v2c community, or to an SNMP v3 user. There is a default access level for communities, and this access level is read-only. This means that you cannot write to an individual data object that has a read/write access type until you change the default read-only access level of the community or user.

The way to modify this default access level is by using the Configuration utility to grant read/write access to either a community (for SNMP v1 and v2c) or a user (SNMP v3), for a given OID. When you set the access level of a community or user to read/write, and an individual data object has a read-only access type, access to the object remains read-only. In short, the access level or type that is the most secure takes precedence when there is a conflict.

To configure SNMP on the BIG-IP system, you must perform a series of small tasks.

Specifying BIG-IP system information

You can use the Configuration utility to specify some basic system information.

1. On the Main tab, click System > SNMP.
2. In the Global Setup area, in the Contact Information field, type contact information such as a name and email address. The contact information is a MIB-II simple string variable defined by almost all SNMP machines. The contact name usually contains a user name, as well as an email address.
3. In the Machine Location field, type the location of the system, such as Network Closet 1. The machine location is a MIB-II variable that almost all machines support. It is a simple string that defines the location of the machine.

Configuring client access

An SNMP client refers to any system running the SNMP manager software for the purpose of remotely managing the BIG-IP system. To set up client access to the BIG-IP system, you specify the IP or network

addresses (with netmask as required) from which the SNMP agent can accept requests. (By default, SNMP is enabled only for the BIG-IP system loopback interface, 127.0.0.1.)

Allowing access to the SNMP agent

Use this procedure to allow client access to an SNMP agent.

1. On the Main tab, click System > SNMP.
2. For the Client Allow List setting, Type option, select Host or Network, depending on whether the IP address you specify is a host system or a subnet.
3. Specify the following information:
    a. In the Address field, type an IP address or network address from which the SNMP agent can accept requests.
    b. If you selected Network in step 3, type the netmask in the Mask field.
4. Click the Add button to add the host or network address to the list of allowed clients.
5. Click Update.

After you perform this task, the BIG-IP system has a list of IP addresses from which the system can accept SNMP requests.

Allowing monitoring of the SNMP agent

Use this procedure to configure the BIG-IP system to create a self IP address. This makes it possible for a client to monitor the SNMP agent.

1. On the Main tab, click Network > Self IPs.
2. If you have not configured the self IP address that you will use for monitoring the SNMP agent, click Create. Otherwise, in the IP Address column, click a self IP address.
3. From the Port Lockdown list, select Allow Custom.
4. Select UDP.
5. Select Port, and in the field, type 161 (the well-known port number for SNMP).
6. Click Add.
7. Click Finished or Update.

After you perform this task, a client system can monitor an SNMP agent.

Controlling access to SNMP data

To better control access to SNMP data, you can assign an access level to an SNMP v1 or v2c community, or to an SNMP v3 user.

Granting community access to v1 or v2c SNMP data

Note: SNMPv1 does not support Counter64 OIDs, which are used for accessing most statistics. Therefore, for SNMPv1 clients, an SNMP walk command skips any OIDs of type Counter64. F5 Networks recommends that you use only clients that support SNMPv2 or higher.

1. On the Main tab, click System > SNMP.
2. From the Agent menu, choose Access (v1, v2c).
3. In the upper-right corner of the screen, click Create.
4. Select the type of address to which the access record applies, either IPv4 or IPv6.
5. In the Community field, type the name of the SNMP community for which you are assigning an access level.
6. In the Source field, type the source IP address.
7. In the OID field, type the OID for the top-most node of the SNMP tree to which the access applies.
8. For the Access setting, select an access level, either Read Only or Read/Write. (This access level applies to the community name you specified previously.)
9. Click Finished.

When you use the Configuration utility to assign an access level to a community, the utility updates the snmpd.conf file, assigning only a single access setting to the community.

Granting user access to v3 SNMP data

To better control access to SNMP data, you can assign an access level to an SNMP v3 user.

1. On the Main tab, click System > SNMP.
2. From the Agent menu, choose Access (v3).
3. In the upper-right corner of the screen, click Create.
4. In the User Name field, type a user name for which you are assigning an access level.
5. For the Authentication setting, select a type of authentication to use, and then type and confirm the user's password.

6. For the Privacy setting, select a privacy protocol, and either type and confirm the user's password, or select the Use Authentication Password check box.
7. In the OID field, type the object identifier (OID) for the top-most node of the SNMP tree to which the access applies.
8. For the Access setting, select an access level, either Read Only or Read/Write. This access level applies to the user name that you specified previously.
9. Click Finished.

When you use the Configuration utility to assign an access level to a user, the utility updates the snmpd.conf file, assigning only a single access setting to the user.

Implementation result

When you use the Configuration utility to assign an access level to a community, the utility updates the snmpd.conf file, assigning only a single access setting to the community. This figure shows a sample snmpd.conf file when you use the Configuration utility to grant read/write access to a community:

```
rocommunity public default
rwcommunity public1 127.0.0.1 .1.3.6.1.4.1.3375.2.2.10.1
```

In this example, the string rocommunity identifies a community named public as having the default read-only access level (indicated by the strings ro and default). This read-only access level prevents any allowed SNMP manager in community public from modifying a data object, even if the object has an access type of read/write.

The string rwcommunity identifies a community named public1 as having a read/write access level (indicated by the string rw). This read/write access level allows any allowed SNMP manager in community public1 to modify a data object under the tree node `.1.3.6.1.4.1.3375.2.2.10.1` (ltmVirtualServ) on the local host `127.0.0.1`, if that data object has an access type of read/write.

Email

Prior to version 11 the BIG-IP used postfix to send email alerts.

If you want the BIG-IP system to send email or alerts, you need to create an SMTP server configuration.

1. On the Main tab, click System > Configuration > Device > SMTP.
2. Click Create. The New SMTP Configuration screen opens.
3. In the Name field, type a name for the SMTP configuration.

4. In the SMTP Server Host Name field, type the IP address or the fully qualified domain name (FQDN) of the SMTP server. If using the FQDN, make sure the DNS server is on the DNS lookup server list and configure the DNS server on the BIG-IP system (System > Configuration > Device > DNS).
5. In the SMTP Server Port Number field, type the port number used for the SMTP server. Typically, the default SMTP port numbers are 25 (unencrypted or TLS), 465 (SSL encrypted), or 587 (TLS encrypted).
6. In the Local Host Name field, type the host name used in SMTP headers in the format of an FQDN; for example, bigip.net. This setting does not refer to the host name of the BIG-IP system.
7. In the From Address field, type the email address that the email is being sent from. This is the address that the recipient sees. Because the BIG-IP system does not accept reply email, use either a valid email address or a from address such as do-not-reply@bigip.net.
8. To encrypt traffic between the BIG-IP system and the SMTP server, for Encrypted Connection, select the type of encryption, SSL (Secure Sockets Layer) or TLS (Transport Layer Security).
9. To require authentication on the SMTP server, for Use Authentication, select the Enabled check box, and type the user name and password.
10. Click Finish.

The SMTP configuration you created is now available for use. For example, you can use it when emailing statistics.

After you create the SMTP configuration, you need to specify it in the appropriate profile. You can create more than one SMTP configuration, if needed.

Remote Syslog

When configuring the BIG-IP system as a data center firewall, you might want to implement high-speed logging and define a group of remote Syslog servers. You can do this by creating a pool of servers, creating a custom request logging profile that determines log content and references the log server pool, and then assigning the profile to each virtual server that you create to process application traffic.

Specifying Syslog servers

Use this task to log messages to one or more remote Syslog servers.

1. From the Main tab, click System > Logs.
2. From the Configuration menu, choose Remote Logging.
3. In the Remote IP field, type the IP address of the remote server to which the BIG-IP system will send the log messages.

4. In the Remote Port field, retain the default port number or type a different port number.
5. Optionally, in the Local IP field, type the IP address of the local BIG-IP system that is sending the log messages.
6. Click Add.
7. Repeat steps 3 through 6 for each remote logging server to which you want the BIG-IP system to send log messages.
8. Click Update.

The remote Syslog servers are defined on the BIG-IP system.

Creating a pool of servers for high-speed logging

For the LTM firewall configuration, you can create a pool of remote servers for high-speed logging.

1. On the Main tab, click Local Traffic > Pools. The Pool List screen opens.
2. Click Create. The New Pool screen opens.
3. In the Name field, type a unique name for the pool.
4. For the Health Monitors setting, in the Available list, select a monitor type, and click << to move the monitor to the Active list.

   Tip: Hold the Shift or Ctrl key to select more than one monitor at a time.
5. From the Load Balancing Method list, select how the system distributes traffic to members of this pool. The default is Round Robin.
6. For the Priority Group Activation setting, specify how to handle priority groups:
   a. Select Disabled to disable priority groups. This is the default option.
   b. Select Less than, and in theAvailable Members field, type the minimum number of members that must remain available in each priority group in order for traffic to remain confined to that group.
7. Using the New Members setting, add the IP address for each logging server that you want to include in the pool:
   a. Type an IP address in the Address field, or select a node address from the Node List.
   b. Type a service number in the Service Port field, or select a service name from the list.
   c. You may type a priority number in the Priority field.

d. Click Add.

8. Click Finished.

The new pool containing the remote Syslog servers appears in the Pools list.

After creating the pool, you must create a request logging profile and specify this pool name within the profile. This eliminates the need for you to assign this pool to a virtual server.

Configuring a profile for high-speed logging

You must have already created a pool that includes logging servers as pool members.

Many sites perform traffic analysis against the log files that their web servers generate. With a Request Logging profile, you can specify the data and the format for HTTP requests and responses that you want to include in a log file. If you prefer, you can tailor the information that appears in the logs so that the logs work seamlessly with whatever analysis tools you use for your origin web server's HTTP log files. You can use a request logging profile to log specific data, and then use that information for analysis and troubleshooting.

1. On the Main tab, click Local Traffic > Profiles > Other > Request Logging. The Request Logging profile list screen opens.
2. Click Create. The New Request Logging Profile screen opens.
3. From the Parent Profile list, select a profile from which the new profile inherits properties.
4. Above the Request Settings area, select the Custom check box. This enables all settings in the Request Settings area, making them available for change.
5. From the Request Logging list, select Enabled.
6. In the Template field, type the request logging parameters for the entries that you want to include in the log file.
7. From the HSL Protocol list, select a high-speed logging protocol.
8. From the Pool Name list, select the pool that includes the logging server as a pool member.
9. Optional: You can also configure the error response settings.
    a. From the Respond On Error list, select Enabled.
    b. In the Error Response field, type the error response strings that you want to include in the log file. These strings must be well-formed for the protocol serving the strings.
    c. Select the Close On Error check box to drop the request and close the connection if logging fails.

10. Optional: You can also configure the logging request errors settings.
    a. From the Log Logging Errors list, select Enabled.
    b. In the Error Template field, type the request logging parameters for the entries that you want to include in the log file.
    c. From the HSL Error Protocol list, select a high-speed logging error protocol.
    d. From the Error Pool Name list, select a pool that includes the node for the error logging server as a pool member.
11. Click Update.

This configures a request logging profile to log specified data for HTTP requests.

1.07b - Identify the location of custom alert configuration files

Configuring Custom SNMP Traps
Configuring the BIG-IP System to Deliver Locally-Generated Email Messages (11.x)

SNMP trap configuration files

Standard, pre-configured SNMP traps are contained in the /etc/alertd/alert.conf file. F5 does not recommend, or support, the addition or removal of traps or any other changes to the alert.conf file.

Custom, user-defined SNMP traps should be defined in the /config/user_alert.conf file.

When the alertd process starts, it creates a dynamic configuration file by appending the /config/user_alert.conf file to the /etc/alertd/alert.conf file. The system searches the dynamic configuration file sequentially for matches. Once a match is found, the trap is generated and no further matches are attempted.

> **Note:** All files in the /config directory, including any customizations to the /config/user_alert.conf file, are automatically included in the UCS archive by default.

SMTP configuration

In BIG-IP 11.0.0 and later versions, you can use ssmtp to configure the system to send locally generated email messages to a configured Simple Mail Transfer Protocol (SMTP) mail hub. The ssmtp program accepts a mail stream on standard input with email recipients specified on the command line.

> **Note:** In versions prior to BIG-IP 11.0.0, you can use the Postfix software to deliver locally generated email messages. Starting in BIG-IP 11.0.0, F5 removed Postfix from the BIG-IP software. For more information, refer to SOL13182: Postfix has been removed from BIG-IP software.

You can configure the BIG-IP system to deliver locally-generated email messages using ssmtp by performing one of the following procedures, depending on the BIG-IP version.

Procedures

Configuring ssmtp to send locally generated email messages (11.0.0 - 11.4.1)

Impact of procedure: Performing the following procedure should not have a negative impact on your system.

1. Log in to the command line.
2. Change directories to the /etc/ssmtp directory by typing the following command:
   `cd /etc/ssmtp`
3. Using a text editor, edit the ssmtp.conf file.
4. Edit the mailhub variable to be the host name or IP address of a valid SMTP mail host.

   For example:
   `mailhub=smtp-mail.domain.com`

   Note: If the BIG-IP system is not configured to resolve host names, you can configure a static host name for the mail server by logging in to the Configuration utility and navigating to System > Configuration > Device > Hosts.
5. Optional: If the BIG-IP system has custom alerts configured in /config/user_alert.conf to send an email notification using the fromaddress parameter, you must clear the comment in the following line in the /etc/ssmtp/ssmtp.conf file:
   `FromLineOverride=YES`
6. Optional: If the SMTP mail host uses Secure Socket Layer (SSL)/TLS authentication, you must add the following three additional parameters into the /etc/ssmtp/ssmtp.conf configuration file:

   `UseTLS=yes`
   `AuthUser=<SMTP-Account-Username>`
   `AuthPass=<SMTP-Account-Password>`
   <SMTP-Account-Username> refers to the user name of a valid account on the SMTP mail host server.
   <SMTP-Account-Password> refers to the password of the user account on the SMTP mail host server.
7. Save the changes, and then exit the text editor.
8. Verify that ssmtp is configured properly and can send an email message.

To do so, send a test email message using a command that appears similar to the following example:

```
echo "ssmtp test mail" | mail -vs "Test email for SOL13180" myemail@mydomain.com
```

9. Verify that you receive the email message.

1.07c - Identify the available levels for local traffic logging

Event Logging

Log Levels Related to Traffic Management Events

Traffic Management events use levels to distinguish the severity of the event. Traffic Management uses these severity levels when designating log levels. Log levels set the threshold at which Traffic Management event messages start accruing in the log files. Traffic Management event messages equal to and greater than the specified log level are written to the log file.

> **Important:** Use caution when changing a log level from its default setting. Refer to the event mapping files in the /etc/alertd/ directory to determine which Traffic Management event messages are associated with each log level. Repeat this procedure after each upgrade, as some Traffic Management event messages and their associated log levels may differ between software releases.

The log levels that you can set on certain types of events, ordered from highest severity to lowest severity, are:

- Emergency
- Alert
- Critical
- Error
- Warning
- Notice
- Informational
- Debug

For example, if you set the minimum log level for iRules events to Error, then the system only logs messages that have a severity of Error or higher for those events. If you retain the default minimum log level (Informational), then the system logs all messages that have a severity of Informational or higher (that is, all messages except Debug messages).

There are many different types of local traffic events for which you can set a minimum log level. Table 14.5, shows the types of local traffic events and the minimum log levels that you can configure for them. Because not all log levels are available for every local-traffic event type, the table shows the specific log levels you can set on each event type. Following the table is the procedure for setting the minimum log level on a local traffic event type.

Note: The default setting is indicated by the wildcard character, an asterisk (*).

Note: Not all log levels are available for all objects.

Local-Traffic Event Type	Available Minimum Log Levels	Default Value
ARP/NDP	Error, Warning, Notice, Informational, Debug	Warning
HTTP	Error, Debug	Error
HTTP Compression	Error, Debug	Error
IP	Warning, Notice	Warning
Layer 4	Notice, Informational, Debug	Notice
MCP	Emergency, Alert, Critical, Error, Warning, Notice, Informational, Debug	Notice
Network	Critical, Error, Warning, Notice, Informational, Debug	Warning
Packet Velocity ASIC	Informational, Debug	Informational
iRules	Error, Informational, Debug	Informational
SSL	Error, Warning	Warning
Traffic Management OS	Emergency, Critical, Error, Notice, Informational	Notice
LIND	Emergency, Alert, Critical, Error, Warning, Notice, Informational, Debug	Notice
CSYNCD	Emergency, Alert, Critical, Error, Warning, Notice, Informational, Debug	Notice
CLUSTERD	Emergency, Alert, Critical, Error, Warning, Notice, Informational, Debug	Notice

In addition to Traffic Management events, you have the ability to control the level of Audit logging performed. Audit logging allows you to track configuration modifications.

Audit - Config.Auditing

Specifies the log level for configuration changes on the system.

Log level	Configuration changes
Disable	This turns audit logging off. This is the default value.
Enable	This causes the system to log messages for user-initiated configuration changes only.
Verbose	This causes the system to log messages for user-initiated configuration changes and any loading of configuration data.
Debug (All)	This causes the system to log messages for all user-initiated and system-initiated configuration changes.

1.07d - Identify the steps to trigger custom alerts for testing purposes

Testing SNMP Traps on the BIG-IP System (9.4.x - 11.x)

How to trigger custom alerts for testing purposes

SNMP traps are triggered when the alertd process receives input from the syslog-ng utility that matches an alert code. The alertd process then performs the action specified in the /etc/alertd/alert.conf file, such as sending an SNMP trap.

When configuring or troubleshooting SNMP traps on the BIG-IP system, you may want to generate test syslog-ng messages that the BIG-IP system will send as SNMP traps.

Using the logger utility to generate test SNMP traps

To generate test syslog-ng messages, you can use the logger utility, which is a shell command interface, to the syslog-ng system log module. You can use the logger utility to make entries in the syslog-ng system log.

1. Log in to the command line.
2. To view pre-configured SNMP traps in the /etc/alertd/alert.conf file, type the following command:

   ```
   less /etc/alertd/alert.conf
   ```

3. Find one of the pre-configured traps. For example, pre-configured traps contain the word snmptrap, and appear similar to the following example:

   ```
   alert BIGIP_MCPD_MCPDERR_NODE_ADDRESS_MON_STATUS {
        snmptrap OID=".1.3.6.1.4.1.3375.2.4.0.12"
     }
   ```

4. The alert BIGIP_MCPD_MCPDERR_NODE_ADDRESS_MON_STATUS entry indicates that the mcpd process will report this information, for which the bigip_mcpd_error_maps.h file contains all of the corresponding alert code reported by mcpd, and the message format.

   For example:

   ```
   /etc/alertd/bigip_mcpd_error_maps.h

   # cat bigip_mcpd_error_maps.h | grep NODE_ADDRESS

   0 LOG_DEBUG    01070587 BIGIP_MCPD_MCPDERR_MONITOR_RULE_NODE_ADDRESS_ONLY
  "The requested monitor rule (%s on pool %s) can only be applied to node
   addresses."

   0 LOG_NOTICE    01070640 BIGIP_MCPD_MCPDERR_NODE_ADDRESS_MON_STATUS "Node %s
   monitor status %s."

   0 LOG_NOTICE    01070641 BIGIP_MCPD_MCPDERR_NODE_ADDRESS_SESS_STATUS "Node
   %s session status %s."

   0 LOG_ERR    01070694 BIGIP_MCPD_MCPDERR_NODE_ADDRESS_RATIO_INVALID "The
   requested ratio for the node address (%s) is invalid. The ratio must be
   greater than zero."

   0 LOG_NOTICE    01070728 BIGIP_MCPD_MCPDERR_NODE_ADDRESS_MON_STATUS_UP "Node
   %s monitor status up."

   The command output includes all of the alert codes and descriptions that
   deal directly with NODE_ADDRESS reported by the mcpd process. For this
   example, only the second entry is required since it matched the predefined
   SNMP trap entry in the /etc/alertd/alert.conf file:

   0 LOG_NOTICE    01070640 BIGIP_MCPD_MCPDERR_NODE_ADDRESS_MON_STATUS "Node %s
   monitor status %s."
   ```

   Note: For more information about the map files and how the alertd process receives information from the syslog-ng utility, refer to SOL6420: The /var/run/bigip_error_maps.dat file maps the alertd process input from the syslog-ng utility to an alert name.

5. Extract the syslog level, alert code, and syslog message string contained in double quotes:

   ```
   0 LOG_NOTICE  01070640 BIGIP_MCPD_MCPDERR_NODE_ADDRESS_MON_STATUS "Node %s
   monitor status %s."
   ```

For example:

```
Log level ID: LOG_NOTICE
Alert Code: 01070640
Syslog Message String: "Node %s monitor status %s."
```

6. Use the extracted information to define your logger command, as follows:
   logger -p <facility>.<level> "<alert code>:<log level>: "<Syslog Message String>"

   For example, using the information extracted in step 5, you would create the following logger syntax:

   ```
   logger -p local0.notice "01070640:5: Node 10.10.64.14 monitor status down."
   ```

   In the previous syntax, note the following:

   - Local0 is the facility specified in the /etc/syslog-ng/syslog-ng.conf file as the /var/log/ltm file.
   - Notice is the log level, which can include one of any eight available log levels.
   - 01070640 is the alert code as indicated in the /etc/alertd/bigip_mcpd_error_maps.h file.
   - 5 is the log level. This level corresponds with the following values listed in the /etc/alertd/alert.conf file:

Log level	Description	Corresponding syslog level
0	System is unusable	emerg
1	Action must be taken immediately	alert
2	Critical Conditions	crit
3	Error Conditions	err
4	Warning Conditions	warning
5	Normal but significant condition	notice
6	Informational	info
7	Debug-level messages	debug

   - Node `10.10.64.14` monitor status down is the syslog message string. Note that the corresponding %s values were replaced with the node IP address and the monitor status.

   This command will output a syslog-ng message to the local0.notice facility (the default location is the /var/log/ltm file), and an SNMP trap for this message will be generated.

   Note: For information about the various log facilities and levels, refer to SOL5531: Configuring the level of information that syslog-ng sends to log files.

   For information about which daemons log to which log files, refer to SOL8035: Overview of BIG-IP daemons.

SECTION 2 - TROUBLESHOOT BASIC HARDWARE ISSUES

Objective - 2.01 Determine which iRule to use to resolve an application issue

2.01a - Determine which iRule events and commands to use

iRules Content on DevCentral
iRules Content on DevCentral
iRules Content on DevCentral
iRules Content on DevCentral
iRules Content on DevCentral
iRules Content on DevCentral
iRules Content on DevCentral

*links on DevCentral require member login

Ref: 1, Chapter 13: iRules.

iRule Events and Commands

An iRule is a powerful and flexible feature within the BIG-IP operating system that you can use to manage your network traffic. The iRules feature not only allows you to select pools based on header data, but also allows you to direct traffic by searching on any type of content data that you define. Thus, the iRules feature significantly enhances your ability to customize your content switching to suit your exact needs.

Events

iRules are event-driven, which means that the LTM system triggers an iRule based on an event that you specify in the iRule. An event declaration is the specification of an event within an iRule that causes the LTM system to trigger that iRule whenever that event occurs. Examples of event declarations that can trigger an iRule are HTTP_REQUEST, which triggers an iRule whenever the system receives an HTTP request, and CLIENT_ACCEPTED, which triggers an iRule when a client has established a connection.

Commands

An iRule command within an iRule causes the LTM system to take some action, such as querying for data, manipulating data, or specifying a traffic destination. The types of commands that you can include within iRules are:

- Statement commands - These commands cause actions such as selecting a traffic destination or assigning a SNAT translation address. An example of a statement command is pool <name>, which directs traffic to the named load balancing pool.
- Query commands - These commands search for header and content data. An example of a query command is IP::remote_addr, which searches for and returns the remote IP address of a connection.
- Data manipulation commands -These commands perform data manipulation such as inserting headers into HTTP requests. An example of a data manipulation command is HTTP::header remove <name>, which removes the last occurrence of the named header from a request or response.
- Utility commands - These commands are functions that are useful for parsing and manipulating content. An example of a utility command is ‘’’decode_uri <string>*, which decodes the named string using HTTP URI encoding and returns the result. For more information on using utility commands, see Using utility commands.

Objective - 2.02 Explain the functionality of a given iRule

2.02a - Interpret information in iRule logs to determine the iRule and iRule events where they occurred

iRules 101 - #09 - Debugging
DevCentral iRules Log

Logging with iRules

The first tool you will want to arm yourself with is the iRules “log” command.

The log statement generates and logs the specified message to the Syslog-ng utility. This command works by performing variable expansion on the message as defined for the HTTP profile Header Insert setting.

The log command can produce large amounts of output. Use with care in production environments, especially where disk space is limited.

The syslog facility is limited to logging 1024 bytes per request. Longer strings will be truncated.

The High Speed Logging feature offers the ability to send TCP or UDP syslog messages from an iRule with very low CPU or memory overhead. Consider using HSL instead of the default log command for remote logging.

The syntax for the log is:

```
log [<facility>.<level>] <message>
```

facility: `"local0", "local1", "local2", "local3", "local4", "local5", "local6", "local7"`

level: `"alert", "crit", "debug", "emerg", "err", "error", "info", "none", "notice", "panic", "warn", "warning"`

While the facility and level parameters are optional, it is good to know that there is a significant behavioral difference when the optional <facility>.<level> is specified. When iRule logs messages without the facility and/or level, they are rate-limited as a class and subsequently logged messages within the rate-limit period may be suppressed even though they are textually different. However, when the <facility> and/or <level> are specified, the log messages are not rate-limited (though syslog-ng will still perform suppression of repeated duplicates).

That is a lot of options. Unless you are doing some customization in syslog-ng regarding the different facilities and levels, you can use with the defaults of "local0" and "error" which are the defaults. Actually, we've made it even easier than that for you, in that you can omit the level parameter and we'll default it for you. In almost every iRule you will see on DevCentral, the following syntax is used and in 99% of the cases, it will be all that you need.

```
log local0. "message goes here"
```

This will ensure that the log messages are not rate limited and go directly to the log files and that they will be stored in the system log file: /var/log/ltm.

Remove Logging in Production

Debug logging is a great tool when testing your application deployments, or even when fixing an issue with production servers. But, log messages do fill up the system logs and the system disks are only so big. In most cases, debug logging should be disabled when you've got all the kinks worked out. This can be done in several ways:

1. Remove the log commands from the iRule. This is probably the easiest to implement, just delete the log lines and click save. This option will reduce the clutter in your iRule and makes it easier to read.
2. Comment out the log commands with a # sign. This will enable you to easily restore the log commands if another situation comes up where you need to figure out a new app error. Just uncomment the log lines, click save, and you are back in business.
3. Use conditional log statements based on global variables. By wrapping log statements with an if statement testing the value of a variable, you can make turning on and off logging as simple as changing a variable.

The method you use will solely depend on your own situation. Options 1 and 2 take no CPU overhead in the log processing, while option 3 still requires performing a Boolean test on a variable. For hundreds of thousands of requests, this can add up.

2.02b - Describe the results of iRule errors

Error Message: 01220001:3: TCL Error

iRule Errors

When an iRule contains an error, such as a missing variable, the system generates a TCL error indicating the missing or incorrect element. A TCL runtime error aborts the affected instance of the iRule, and may cause the associated connection to be reset. The error message can provide valuable information when creating and troubleshooting iRule syntax.

If the error message occurs during operation or while creating an iRule, use the information in the error message to troubleshoot the iRule syntax. If the error occurs after upgrading the BIG-IP system to a new software release, refer to the DevCentral site and verify whether any portion of the iRule syntax (such as an iRule command) was deprecated or changed.

Error Message

```
Error Message: 01220001:3: TCL error
```

For example:

```
01220001:3: TCL error: /Common/broken <RULE_INIT> - can't read "b": no such
variable while executing "set a $b"
```

```
01220001:3: TCL error: MyiRule <HTTP_RESPONSE> - wrong # args: should be ""persist
add uie <key>" while executing "persist add uie [HTTP::cookie "cookie_name"]"
```

```
01220001:3: TCL error: MyiRule - Out of bounds (line 2) invoked from within
"HTTP::payload replace 0 $content_length [string repeat "X" $content_length]"
```

Objective - 2.03 Interpret AVR information to identify performance issues or application attacks

2.03a - Explain how to use AVR to trace application traffic

BIG-IP® Analytics: Implementations
Troubleshooting Applications by Capturing Traffic

AVR to trace application traffic

This implementation describes how to set up the BIG-IP system to collect application traffic so that you can troubleshoot problems that have become apparent by monitoring application statistics. For example, by examining captured requests and responses, you can investigate issues with latency, throughput, or reduced transactions per second to understand what is affecting application performance.

When Application Visibility and Reporting (AVR) is provisioned, you can create an Analytics profile that includes traffic capturing instructions. The system can collect application traffic locally, remotely, or both. If the system is already monitoring applications, you can also update an existing Analytics profile to make it so that it captures traffic.

If logging locally, the system logs the first 1000 transactions and displays charts based on the analysis of those transactions. If logging remotely, the system logs information on that system; log size is limited only by any constraints of the remote logging system. To see updated application statistics, you can clear the existing data to display the current statistics.

Prerequisites for capturing application traffic

After you finish a basic networking configuration of the BIG-IP system, you must complete the following tasks as prerequisites for setting up application statistics collection:

- Provision Application Visibility and Reporting (AVR): System > Resource Provisioning
- Create an iAppsTM application service (go to iApp > Application Services), or configure at least one virtual server with a pool pointing to one or more application servers.
- The Traffic Sampling Ratio must be set to all in the default Analytics profile.

You can set up the system for capturing traffic locally or remotely (or both).

> **Note:** Before setting up traffic capturing, it is a good idea to clear the captured transaction log. On the Captured Transactions screen, click Clear All to clear all previously captured data records.

Capturing traffic for troubleshooting

To set up traffic capturing, the Transaction Sampling Ratio of the default analytics profile must be set to All.

You can configure the BIG-IP system to capture application traffic and store the information locally or remotely (on syslog servers or SIEM devices, such as Splunk). To do this, you create an Analytics profile designed for capturing traffic. The profile instructs the BIG-IP system to collect a portion of application traffic using the Application Visibility and Reporting module.

> **Note:** You typically use traffic capturing if you notice an application issue, such as trouble with throughput or latency, discovered when examining application statistics, and want to troubleshoot the system by examining actual transactions.

1. On the Main tab, click Local Traffic > Profiles > Analytics.

   Tip: If Analytics is not listed, this indicates that Application Visibility and Reporting (AVR) is not provisioned, or you do not have rights to create profiles.

   The Analytics screen opens and lists all Analytics profiles that are on the system, including a default profile called analytics.

2. Click Create.

   The New Analytics Profile screen opens. By default, the settings are initially the same as in the default analytics profile.

3. In the Profile Name field, type a name for the Analytics profile.

4. To the right of the General Configuration area, click the Custom check box.

   The settings in the area become available for modification.

5. For Traffic Capturing Logging Type, specify where to store captured traffic.

- To store traffic locally, click Internal. You can view details on the Statistics: Captured Transactions screen. This option is selected by default.
- To store traffic on a remote logging server, click External and type the Remote Server IP Address and Remote Server Port number.

  Tip: If you specify remote logging for multiple applications, you can use the Remote Server Facility filter to sort the data for each.

6. In the Included Objects area, specify the virtual servers for which to capture application statistics:

   a) For the Virtual Servers setting, click Add.

A popup lists the virtual servers that you can assign to the Analytics profile.

b) From the Select Virtual Server popup list, select the virtual servers to include and click Done.

Note: You need to have previously configured the virtual servers (with an HTTP profile) for them to appear in the list. Also, you can assign only one Analytics profile to a virtual server so the list shows only virtual servers that have not been assigned an Analytics profile.

Special considerations apply if using Analytics on a BIG-IP system with both Application Security ManagerTM and Access Policy ManagerTM, where security settings (in Portal Access WebTop or an iRule) redirect traffic from one virtual server to a second one. In this case, you need to attach the Analytics profile to the second virtual server to ensure that the charts show accurate statistics.

7. In the Statistics Gathering Configuration, for Collected Metrics, select the statistics you want the system to collect:

Option	Description
Server Latency	Tracks how long it takes to get data from the application server to the BIG-IP system (selected by default).
Page Load Time	Tracks how long it takes an application user to get a complete response from the application, including network latency and completed page processing. **Note:** End user response times and latencies can vary significantly based on geography and connection types.
Throughput	Saves information about HTTP request and response throughput (selected by default).
User Sessions	Stores the number of unique user sessions. For Timeout, type the number of minutes of user non-activity to allow before the system considers the session to be over. If using transaction sampling, this option is not available.

8. For Collected Entities, select the entities for which you want the system to collect statistics:

Option	Description
URLs	Collects the requested URLs.
Countries	Saves the name of the country where the request came from based on the client IP address.
Client IP Addresses	Saves the IP address where the request originated. The address saved also depends on whether the request has an XFF (X-forwarded-for) header and whether Trust XFF is selected.
Response Codes	Saves HTTP response codes that the server returned to requesters (selected by default).
User Agents	Saves information about browsers used when making the request.
Methods	Saves HTTP methods in requests (selected by default).

9. In the Capture Filter area, from the Capture Requests and Capture Responses lists, select the options that indicate the part of the traffic to capture.

Option	Description
None	Specifies that the system does not capture request (or response) data.
Headers	Specifies that the system captures request (or response) header data only.
Body	Specifies that the system captures the body of requests (or responses) only.
All	Specifies that the system captures all request (or response) data.

10. Depending on the application, customize the remaining filter settings to capture the portion of traffic to that you need for troubleshooting.

    Tip: By focusing in on the data and limiting the type of information that is captured, you can troubleshoot particular areas of an application more quickly. For example, capture only requests or responses, specific status codes or methods, or headers containing a specific string.

11. Click Finished.

The BIG-IP system captures the application traffic described by the Analytics profile for 1000 transactions locally (or until system limits are reached). If logging remotely, the system logs information on that system; log size is limited only by constraints of the remote logging system.

Note: System performance is affected when traffic is being captured.

Reviewing captured traffic

Before you can review captured traffic details on the BIG-IP system, you need to have created an Analytics profile that is capturing application traffic locally. The settings you enable in the Capture Filter area of the profile determine what information the system captures. You need to associate the Analytics profile with one or more virtual servers, or with an iApps application service.

The system starts capturing application traffic as soon as you enable it on the Analytics profile. You can review the captured transactions locally on the BIG-IP system. The system logs the first 1000 transactions.

1. On the Main tab, click System > Logs > Captured Transactions.

   The Captured Transactions screen opens and lists all of the captured transactions.

2. Optionally, use the time period and filter settings to limit which transactions are listed.

3. In the Captured Traffic area, click any transaction that you want to examine.

   Details of the request will display on the screen.

4. Review the general details of the request.

   Tip: The general details, such as the response code or the size of the request and response, help with troubleshooting.

5. For more information, click Request or Response to view the contents of the actual transaction. Review the data for anything unexpected, and other details that will help with troubleshooting the application.

6. On the Captured Transactions screen, click Clear All to clear all previously captured data records (including those not displayed on the screen) and start collecting transactions again.

   The system captures up to 1000 transactions locally and displays them on the screen. Captured transactions are visible a few seconds after they occur.

2.03b - Explain how latency trends identify application tier bottlenecks

Latency is a classic network performance metric, which at the basic level requires the evaluation of timestamps applied to the same packet as it passes through two locations in the network. By comparing the timestamps, the latency of the network segment can be monitored. Many networked applications and services rely on low latency in order to function correctly.

If you have established latency times for transport traffic in your network and you are seeing latency grow or exceed a threshold that causes user acceptance to drop for an application, you can use it as a basis to look

into changes or setting that may be causing additional latency. Gathering the information and keep track of changes is the key to identifying application tier issues.

Investigating the server latency of applications

Before you can investigate server latency, you need to have created an Analytics profile that is logging statistics internally on the BIG-IP system. The Analytics profile must be associated with one or more virtual servers, or an iApps application service. If your browser is IE8 or earlier, you need to have Adobe Flash Player installed on the computer from which you plan to review the data.

You can review statistics concerning server latency on the Analytics charts. Server latency is how long it takes (in milliseconds) from the time a request reaches the BIG-IP system, for it to proceed to the web application server, and return a response to the BIG-IP system.

1. On the Main tab, click Statistics > Analytics > HTTP. The Overview screen opens.
2. From the Latency menu, click Server Latency. A chart shows the server latency for all applications and virtual servers associated with all Analytics profiles.
3. To view server latency for a specific application, in the Details table, select only that application. The charts show latency only for the selected application.
4. To view server latency for a specific virtual server:
    a. In the View By list, select Virtual Servers. The charts show latency for all virtual servers.
    b. In the Details list near the charts, click the virtual server you are interested in. The charts show latency only for the selected virtual server.
5. If further investigation is needed, in the View By setting, select other entities to view charts that show latency for other collected entities included in the Analytics profile, for example, specific pool members, URLs, countries, or client IP addresses.

    Tip: If you are concerned about server latency, you can configure the Analytics profile so that it sends an alert when the average server latency exceeds a number of milliseconds for some period of time.

2.03c - Explain browser requirements to obtain page load time data

BIG-IP® Analytics: Implementations

Browser requirements to obtain page load times

The system can collect page load times only for clients using browsers that meet the following requirements:

- Supports Navigation Timing by W3C
- Accepts cookies from visited application sites
- Enables JavaScript for the visited application sites

2.03d - Explain how to use advanced filters to narrow output data from AVR

BIG-IP® Analytics: Implementations

AVR advanced filters

Before you can view application page load times, you need to have created an Analytics profile that is logging statistics internally on the BIG-IP system. In the profile, the statistics-gathering configuration must have Page Load Time selected as one of the collected metrics. The Analytics profile also needs to be associated with one or more virtual servers, or an iApps application service.

You can view page load times on the Analytics charts. Page load time is how long (in milliseconds) it takes from the time an end user makes a request for a web page, until the web page from the application server finishes loading on the client-side browser.

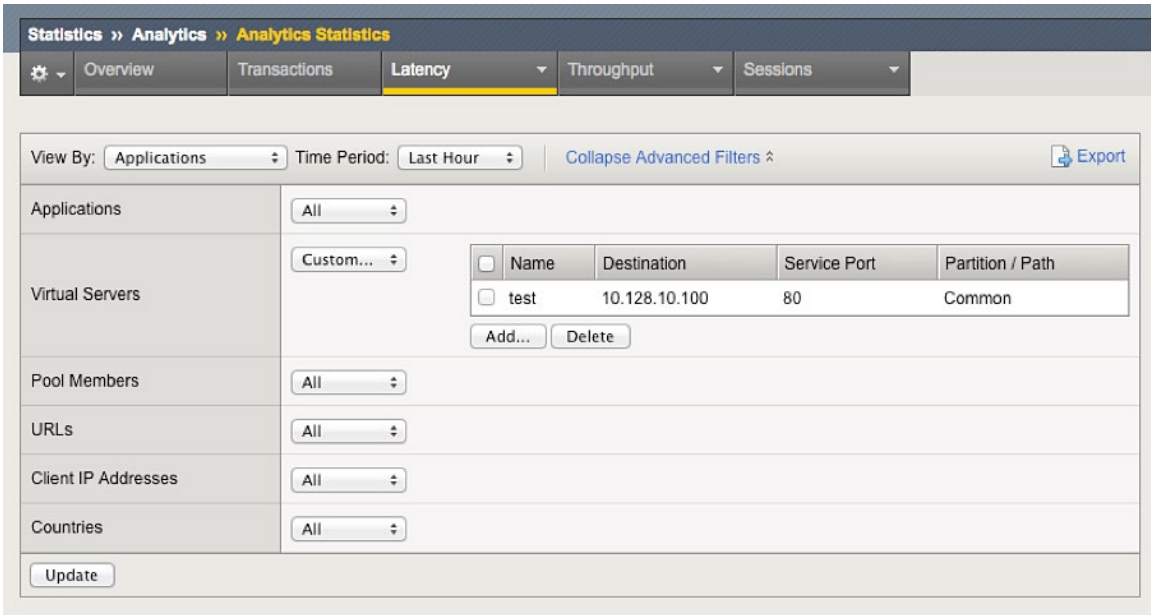
> **Note:** End user response times and latencies can vary significantly based on geography and connection types.

1. On the Main tab, click Statistics > Analytics. The Analytics Overview screen opens.
2. From the Latency menu, choose Page Load Time. Charts show the average page load times in milliseconds for all applications and virtual servers associated with all Analytics profiles.
3. To view average page load time for a specific application, in the Details table, select only that application. The charts refresh and show the page load time only for the selected application.
4. To view page load time for a specific virtual server:
    a. Click Expand Advanced Filters.
    b. For Virtual Servers choose Custom.
    c. Click Add and select the virtual server whose page load times you want to view.

    The charts show page load times for the selected virtual server.

5. To zoom in on page load time during a specific time period, drag your mouse across the chart during the time period you are interested in.

   The system automatically refreshes the chart to display statistics for the time period you selected.



Tip: If you are concerned about maintaining a high level of user experience and productivity, you can configure the Analytics profile so that it sends an alert when the average page load time exceeds a number of milliseconds for some period of time.

2.03e - Explain how to delete AVR data

BIG-IP® Analytics: Implementations

Clearing AVR Data

On the Captured Transactions screen, click Clear All to clear all previously captured data records.

Objective - 2.04 Interpret AVR information to identify LTM device misconfiguration

2.04a - Explain how to use AVR to trace application traffic

See section 2.03a

This topic content is the same as topic 2.03a – simply applied to the 2.04 objective.

2.04b - Explain how latency trends identify application tier bottlenecks

See section 2.03b

This topic content is the same as topic 2.03b – simply applied to the 2.04 objective.

2.04c - Explain how to use advanced filters to narrow output data from AVR

See section 2.03d

This topic content is the same as topic 2.03d – simply applied to the 2.04 objective.

2.04d - Explain how to modify profile settings using information from the AVR

Troubleshooting Applications by Capturing Traffic

Troubleshooting applications by capturing traffic

You can setup the BIG-IP system to collect application traffic so that you can troubleshoot problems that have become apparent by monitoring application statistics. For example, by examining captured requests and responses, you can investigate issues with latency, throughput, or reduced transactions per second to understand what is affecting application performance.

When Application Visibility and Reporting (AVR) is provisioned, you can create an Analytics profile that includes traffic capturing instructions. The system can collect application traffic locally, remotely, or both. If the system is already monitoring applications, you can also update an existing Analytics profile to make it so that it captures traffic.

If logging locally, the system logs the first 1000 transactions and displays charts based on the analysis of those transactions. If logging remotely, the system logs information on that system; log size is limited only by any constraints of the remote logging system. To see updated application statistics, you can clear the existing data to display the current statistics.

If you discover that an application is having issues with latency you may want to look into applying a WAN optimized TCP profile or simply tweak buffer sizes in the existing assigned TCP profile. The possibilities are endless so documenting all of the ways you can use the output of analytics to tune profile settings is not possible for this document, but you will need to spend time looking at the output of AVR so you can correlate what you are seeing to which settings affect that value.

2.04e - Explain how to delete AVR data

See section 2.03e

This topic content is the same as topic 2.03e – simply applied to the 2.04 objective.

Objective - 2.05 Given a scenario, determine the appropriate headers for an HTTP/HTTPS application

Scenario Based Objective

To prepare for scenario based questions the candidate will need to complete hands-on configuration and testing of the configuration on the LTM. This will allow the candidate to better understand how different configurations can produce different results. All F5 exams use scenario-based questions that make the candidate apply what they know to a situation to determine the resulting outcome.

This topic is focused on HTTP and HTTPS application functionality.

2.05a - Explain how to interpret response codes

10 Status Code Definitions

Response Codes

The Status-Code element is a 3-digit integer result code of the attempt to understand and satisfy the request. The Reason-Phrase is intended to give a short textual description of the Status-Code. The Status-Code is intended for use by automata and the Reason-Phrase is intended for the human user. The client is not required to examine or display the Reason-Phrase.

The first digit of the Status-Code defines the class of response. The last two digits do not have any categorization role. There are 5 values for the first digit:

- 1xx: Informational - Request received, continuing process
- 2xx: Success - The action was successfully received, understood, and accepted
- 3xx: Redirection - Further action must be taken in order to complete the request
- 4xx: Client Error - The request contains bad syntax or cannot be fulfilled
- 5xx: Server Error - The server failed to fulfill an apparently valid request

Status Code Definitions

Each Status-Code is described below, including a description of which method(s) it can follow and any meta information required in the response.

Informational 1xx

This class of status code indicates a provisional response, consisting only of the Status-Line and optional headers, and is terminated by an empty line. There are no required headers for this class of status code. Since HTTP/1.0 did not define any 1xx status codes, servers must not send a 1xx response to an HTTP/1.0 client except under experimental conditions.

A client must be prepared to accept one or more 1xx status responses prior to a regular response, even if the client does not expect a 100 (Continue) status message. A user agent MAY ignore an unexpected 1xx status response.

Proxies must forward 1xx responses, unless the connection between the proxy and its client has been closed, or unless the proxy itself requested the generation of the 1xx response. (For example, if a proxy adds a “Expect: 100-continue” field when it forwards a request, then it need not forward the corresponding 100 (Continue) response(s).)

100 Continue

The client should continue with its request. This interim response is used to inform the client that the initial part of the request has been received and has not yet been rejected by the server. The client should continue by sending the remainder of the request or, if the request has already been completed, ignore this response. The server must send a final response after the request has been completed.

101 Switching Protocols

The server understands and is willing to comply with the client's request, via the Upgrade message header field, for a change in the application protocol being used on this connection. The server will switch protocols to those defined by the response's Upgrade header field immediately after the empty line, which terminates the 101 response.

The protocol should be switched only when it is advantageous to do so. For example, switching to a newer version of HTTP is advantageous over older versions, and switching to a real-time, synchronous protocol might be advantageous when delivering resources that use such features.

Successful 2xx

This class of status code indicates that the client's request was successfully received, understood, and accepted.

200 OK

The request has succeeded. The information returned with the response is dependent on the method used in the request, for example:

GET an entity corresponding to the requested resource is sent in the response;

HEAD the entity-header fields corresponding to the requested resource are sent in the response without any message-body;

POST an entity describing or containing the result of the action;

TRACE an entity containing the request message as received by the end server.

201 Created

The request has been fulfilled and resulted in a new resource being created. The newly created resource can be referenced by the URI(s) returned in the entity of the response, with the most specific URI for the resource given by a Location header field. The response should include an entity containing a list of resource characteristics and location(s) from which the user or user agent can choose the one most appropriate. The media type given in the Content-Type header field specifies the entity format. The origin server must create the resource before returning the 201 status code. If the action cannot be carried out immediately, the server should respond with 202 (Accepted) response instead.

A 201 response may contain an ETag response header field indicating the current value of the entity tag for the requested variant just created.

202 Accepted

The request has been accepted for processing, but the processing has not been completed. The request might or might not eventually be acted upon, as it might be disallowed when processing actually takes place. There is no facility for re-sending a status code from an asynchronous operation such as this.

The 202 response is intentionally non-committal. Its purpose is to allow a server to accept a request for some other process (perhaps a batch-oriented process that is only run once per day) without requiring that the user agent's connection to the server persist until the process is completed. The entity returned with this response should include an indication of the request's current status and either a pointer to a status monitor or some estimate of when the user can expect the request to be fulfilled.

203 Non-Authoritative Information

The returned metainformation in the entity-header is not the definitive set as available from the origin server, but is gathered from a local or a third-party copy. The set presented may be a subset or superset of the original version. For example, including local annotation information about the resource might result in a superset of the metainformation known by the origin server. Use of this response code is not required and is only appropriate when the response would otherwise be 200 (OK).

204 No Content

The server has fulfilled the request but does not need to return an entity-body, and might want to return updated metainformation. The response may include new or updated metainformation in the form of entity-headers, which if present should be associated with the requested variant.

If the client is a user agent, it should not change its document view from that which caused the request to be sent. This response is primarily intended to allow input for actions to take place without causing a change to the user agent's active document view, although any new or updated metainformation should be applied to the document currently in the user agent's active view.

The 204 response must not include a message-body, and thus is always terminated by the first empty line after the header fields.

205 Reset Content

The server has fulfilled the request and the user agent should reset the document view, which caused the request to be sent. This response is primarily intended to allow input for actions to take place via user input, followed by a clearing of the form in which the input is given so that the user can easily initiate another input action. The response must not include an entity.

206 Partial Content

The server has fulfilled the partial GET request for the resource. The request must have included a Range header field indicating the desired range, and may have included an If-Range header field to make the request conditional.

The response must include the following header fields:

- Either a Content-Range header field indicating the range included with this response, or a multipart/byteranges Content-Type including Content-Range fields for each part. If a Content-Length header field is present in the response, its value must match the actual number of OCTETs transmitted in the message-body.
- Date
- ETag and/or Content-Location, if the header would have been sent in a 200 response to the same request
- Expires, Cache-Control, and/or Vary, if the field-value might differ from that sent in any previous response for the same variant

If the 206 response is the result of an If-Range request that used a strong cache validator, the response should not include other entity-headers. If the response is the result of an If-Range request that used a weak validator, the response must not include other entity-headers; this prevents inconsistencies between cached entity-bodies and updated headers. Otherwise, the response must include all of the entity-headers that would have been returned with a 200 (OK) response to the same request.

A cache must not combine a 206 response with other previously cached content if the ETag or Last-Modified headers do not match exactly.

A cache that does not support the Range and Content-Range headers must not cache 206 (Partial) responses.

Redirection 3xx

This class of status code indicates that further action needs to be taken by the user agent in order to fulfill the request. The user agent may carry out the action required without interaction with the user if and only if the method used in the second request is GET or HEAD. A client should detect infinite redirection loops, since such loops generate network traffic for each redirection.

> **Note:** previous versions of this specification recommended a maximum of five redirections. Content developers should be aware that there might be clients that implement such a fixed limitation.

300 Multiple Choices

The requested resource corresponds to any one of a set of representations, each with its own specific location, and agent- driven negotiation information is being provided so that the user (or user agent) can select a preferred representation and redirect its request to that location.

Unless it was a HEAD request, the response should include an entity containing a list of resource characteristics and location(s) from which the user or user agent can choose the one most appropriate. The media type given in the Content-Type header field specifies the entity format. Depending upon the format and the capabilities of the user agent, selection of the most appropriate choice may be performed automatically. However, this specification does not define any standard for such automatic selection.

If the server has a preferred choice of representation, it should include the specific URI for that representation in the Location field; user agents may use the Location field value for automatic redirection. This response is cacheable unless indicated otherwise.

301 Moved Permanently

The requested resource has been assigned a new permanent URI and any future references to this resource should use one of the returned URIs. Clients with link editing capabilities ought to automatically re-link references to the Request-URI to one or more of the new references returned by the server, where possible. This response is cacheable unless indicated otherwise.

The Location field in the response should give the new permanent URI. Unless the request method was HEAD, the entity of the response should contain a short hypertext note with a hyperlink to the new URI(s).

If the 301 status code is received in response to a request other than GET or HEAD, the user agent must not automatically redirect the request unless it can be confirmed by the user, since this might change the conditions under which the request was issued.

> **Note:** When automatically redirecting a POST request after receiving a 301 status code, some existing HTTP/1.0 user agents will erroneously change it into a GET request.

302 Found

The requested resource resides temporarily under a different URI. Since the redirection might be altered on occasion, the client should continue to use the Request-URI for future requests. This response is only cacheable if indicated by a Cache-Control or Expires header field.

The temporary URI should be given by the Location field in the response. Unless the request method was HEAD, the entity of the response should contain a short hypertext note with a hyperlink to the new URI(s).

If the 302 status code is received in response to a request other than GET or HEAD, the user agent must not automatically redirect the request unless it can be confirmed by the user, since this might change the conditions under which the request was issued.

> **Note:** RFC 1945 and RFC 2068 specify that the client is not allowed to change the method on the redirected request. However, most existing user agent implementations treat 302 as if it were a 303 response, performing a GET on the Location field-value regardless of the original request method. The status codes 303 and 307 have been added for servers that wish to make unambiguously clear which kind of reaction is expected of the client.

303 See Other

The response to the request can be found under a different URI and should be retrieved using a GET method on that resource. This method exists primarily to allow the output of a POST-activated script to redirect the user agent to a selected resource. The new URI is not a substitute reference for the originally requested resource. The 303 response must not be cached, but the response to the second (redirected) request might be cacheable.

The Location field in the response should give the different URI. Unless the request method was HEAD, the entity of the response should contain a short hypertext note with a hyperlink to the new URI(s).

Note: Many pre-HTTP/1.1 user agents do not understand the 303 status. When interoperability with such clients is a concern, the 302 status code may be used instead, since most user agents react to a 302 response as described here for 303.

304 Not Modified

If the client has performed a conditional GET request and access is allowed, but the document has not been modified, the server should respond with this status code. The 304 response must not contain a message-body, and thus is always terminated by the first empty line after the header fields.

The response must include the following header fields:

- Date, unless its omission is required

  If a clockless origin server obeys these rules, and proxies and clients add their own Date to any response received without one (as already specified by RFC 2068), caches will operate correctly.

- ETag and/or Content-Location, if the header would have been sent in a 200 response to the same request

- Expires, Cache-Control, and/or Vary, if the field-value might differ from that sent in any previous response for the same variant

If the conditional GET used a strong cache validator, the response should not include other entity-headers. Otherwise (i.e., the conditional GET used a weak validator), the response must not include other entity-headers; this prevents inconsistencies between cached entity-bodies and updated headers.

If a 304 response indicates an entity not currently cached, then the cache must disregard the response and repeat the request without the conditional.

If a cache uses a received 304 response to update a cache entry, the cache must update the entry to reflect any new field values given in the response.

305 Use Proxy

The requested resource must be accessed through the proxy given by the Location field. The Location field gives the URI of the proxy. The recipient is expected to repeat this single request via the proxy. 305 responses must only be generated by origin servers.

> **Note:** RFC 2068 was not clear that 305 was intended to redirect a single request, and to be generated by origin servers only. Not observing these limitations has significant security consequences.

306 (Unused)

The 306 status code was used in a previous version of the specification, is no longer used, and the code is reserved.

307 Temporary Redirect

The requested resource resides temporarily under a different URI. Since the redirection may be altered on occasion, the client should continue to use the Request-URI for future requests. This response is only cacheable if indicated by a Cache-Control or Expires header field.

The temporary URI should be given by the Location field in the response. Unless the request method was HEAD, the entity of the response should contain a short hypertext note with a hyperlink to the new URI(s), since many pre-HTTP/1.1 user agents do not understand the 307 status. Therefore, the note should contain the information necessary for a user to repeat the original request on the new URI.

If the 307 status code is received in response to a request other than GET or HEAD, the user agent must not automatically redirect the request unless it can be confirmed by the user, since this might change the conditions under which the request was issued.

Client Error 4xx

The 4xx class of status code is intended for cases in which the client seems to have erred. Except when responding to a HEAD request, the server should include an entity containing an explanation of the error

situation, and whether it is a temporary or permanent condition. These status codes are applicable to any request method. User agents should display any included entity to the user.

If the client is sending data, a server implementation using TCP should be careful to ensure that the client acknowledges receipt of the packet(s) containing the response, before the server closes the input connection. If the client continues sending data to the server after the close, the server's TCP stack will send a reset packet to the client, which may erase the client's unacknowledged input buffers before they can be read and interpreted by the HTTP application.

400 Bad Request

The server due to malformed syntax could not understand the request. The client should not repeat the request without modifications.

401 Unauthorized

The request requires user authentication. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource. The client MAY repeat the request with a suitable Authorization header field. If the request already included Authorization credentials, then the 401 response indicates that authorization has been refused for those credentials. If the 401 response contains the same challenge as the prior response, and the user agent has already attempted authentication at least once, then the user should be presented the entity that was given in the response, since that entity might include relevant diagnostic information. HTTP access authentication is explained in "HTTP Authentication: Basic and Digest Access Authentication".

402 Payment Required

This code is reserved for future use.

403 Forbidden

The server understood the request, but is refusing to fulfill it. Authorization will not help and the request should not be repeated. If the request method was not HEAD and the server wishes to make public why the request has not been fulfilled, it should describe the reason for the refusal in the entity. If the server does not wish to make this information available to the client, the status code 404 (Not Found) can be used instead.

404 Not Found

The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent. The 410 (Gone) status code should be used if the server knows, through some internally configurable mechanism, that an old resource is permanently unavailable and has no forwarding

address. This status code is commonly used when the server does not wish to reveal exactly why the request has been refused, or when no other response is applicable.

405 Method Not Allowed

The method specified in the Request-Line is not allowed for the resource identified by the Request-URI. The response must include an Allow header containing a list of valid methods for the requested resource.

406 Not Acceptable

The resource identified by the request is only capable of generating response entities which have content characteristics not acceptable according to the accept headers sent in the request.

Unless it was a HEAD request, the response should include an entity containing a list of available entity characteristics and location(s) from which the user or user agent can choose the one most appropriate. The media type given in the Content-Type header field specifies the entity format. Depending upon the format and the capabilities of the user agent, selection of the most appropriate choice MAY be performed automatically. However, this specification does not define any standard for such automatic selection.

> **Note:** HTTP/1.1 servers are allowed to return responses which are not acceptable according to the accept headers sent in the request. In some cases, this may even be preferable to sending a 406 response. User agents are encouraged to inspect the headers of an incoming response to determine if it is acceptable.

If the response could be unacceptable, a user agent should temporarily stop receipt of more data and query the user for a decision on further actions.

407 Proxy Authentication Required

This code is similar to 401 (Unauthorized), but indicates that the client must first authenticate itself with the proxy. The proxy must return a Proxy-Authenticate header field containing a challenge applicable to the proxy for the requested resource. The client may repeat the request with a suitable Proxy-Authorization header field. HTTP access authentication is explained in “HTTP Authentication: Basic and Digest Access Authentication”.

408 Request Timeout

The client did not produce a request within the time that the server was prepared to wait. The client may repeat the request without modifications at any later time.

409 Conflict

The request could not be completed due to a conflict with the current state of the resource. This code is only allowed in situations where it is expected that the user might be able to resolve the conflict and resubmit the request. The response body should include enough information for the user to recognize the source of the conflict. Ideally, the response entity would include enough information for the user or user agent to fix the problem; however, that might not be possible and is not required.

Conflicts are most likely to occur in response to a PUT request. For example, if versioning were being used and the entity being PUT included changes to a resource which conflict with those made by an earlier (third-party) request, the server might use the 409 response to indicate that it can't complete the request. In this case, the response entity would likely contain a list of the differences between the two versions in a format defined by the response Content-Type.

410 Gone

The requested resource is no longer available at the server and no forwarding address is known. This condition is expected to be considered permanent. Clients with link editing capabilities should delete references to the Request-URI after user approval. If the server does not know, or has no facility to determine, whether or not the condition is permanent, the status code 404 (Not Found) should be used instead. This response is cacheable unless indicated otherwise.

The 410 response is primarily intended to assist the task of web maintenance by notifying the recipient that the resource is intentionally unavailable and that the server owners desire that remote links to that resource be removed. Such an event is common for limited-time, promotional services and for resources belonging to individuals no longer working at the server's site. It is not necessary to mark all permanently unavailable resources as "gone" or to keep the mark for any length of time -- that is left to the discretion of the server owner.

411 Length Required

The server refuses to accept the request without a defined Content- Length. The client may repeat the request if it adds a valid Content-Length header field containing the length of the message-body in the request message.

412 Precondition Failed

The precondition given in one or more of the request-header fields evaluated to false when it was tested on the server. This response code allows the client to place preconditions on the current resource metainformation (header field data) and thus prevent the requested method from being applied to a resource other than the one intended.

413 Request Entity Too Large

The server is refusing to process a request because the request entity is larger than the server is willing or able to process. The server may close the connection to prevent the client from continuing the request.

If the condition is temporary, the server should include a Retry-After header field to indicate that it is temporary and after what time the client may try again.

414 Request-URI Too Long

The server is refusing to service the request because the Request-URI is longer than the server is willing to interpret. This rare condition is only likely to occur when a client has improperly converted a POST request to a GET request with long query information, when the client has descended into a URI "black hole" of redirection (e.g., a redirected URI prefix that points to a suffix of itself), or when the server is under attack by a client attempting to exploit security holes present in some servers using fixed-length buffers for reading or manipulating the Request-URI.

415 Unsupported Media Type

The server is refusing to service the request because the entity of the request is in a format not supported by the requested resource for the requested method.

416 Requested Range Not Satisfiable

A server should return a response with this status code if a request included a Range request-header field, and none of the range-specifier values in this field overlap the current extent of the selected resource, and the request did not include an If-Range request-header field. (For byte-ranges, this means that the first- byte-pos of all of the byte-range-spec values were greater than the current length of the selected resource.)

When this status code is returned for a byte-range request, the response should include a Content-Range entity-header field specifying the current length of the selected resource. This response must not use the multipart/byteranges content- type.

417 Expectation Failed

This server could not meet the expectation given in an Expect request-header field, or, if the server is a proxy, the server has unambiguous evidence that the next-hop server could not meet the request.

Server Error 5xx

Response status codes beginning with the digit "5" indicate cases in which the server is aware that it has erred or is incapable of performing the request. Except when responding to a HEAD request, the server

should include an entity containing an explanation of the error situation, and whether it is a temporary or permanent condition. User agents should display any included entity to the user. These response codes are applicable to any request method.

500 Internal Server Error

The server encountered an unexpected condition, which prevented it from fulfilling the request.

501 Not Implemented

The server does not support the functionality required to fulfill the request. This is the appropriate response when the server does not recognize the request method and is not capable of supporting it for any resource.

502 Bad Gateway

The server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed in attempting to fulfill the request.

503 Service Unavailable

The server is currently unable to handle the request due to a temporary overloading or maintenance of the server. The implication is that this is a temporary condition, which will be alleviated after some delay. If known, the length of the delay may be indicated in a Retry-After header. If no Retry-After is given, the client should handle the response as it would for a 500 response.

> **Note:** The existence of the 503 status code does not imply that a server must use it when becoming overloaded. Some servers may wish to simply refuse the connection.

504 Gateway Timeout

The server, while acting as a gateway or proxy, did not receive a timely response from the upstream server specified by the URI (e.g. HTTP, FTP, LDAP) or some other auxiliary server (e.g. DNS) it needed to access in attempting to complete the request.

> **Note:** Note to implementers: some deployed proxies are known to return 400 or 500 when DNS lookups time out.

505 HTTP Version Not Supported

The server does not support, or refuses to support, the HTTP protocol version that was used in the request message. The server is indicating that it is unable or unwilling to complete the request using the same major version as the client, as described in section 3.1, other than with this error message. The response should contain an entity describing why that version is not supported and what other protocols that server supports.

2.05b - Explain the function of HTTP headers within different HTTP applications (Cookies, Cache Control, Vary, Content Type & Host)

Fundamentals of HTTP
BIG-IP HTTP Cache and HTTP Vary Header Functionality

HTTP Headers

HTTP headers carry information about behavior and application state between the browser and the server. These headers can be modified and examined by the browser and the server, as well as intermediary devices such as web acceleration solutions and application delivery controllers. The headers sent by the browser notify the web server of the browser's capabilities. The headers sent by the web server tell the browser how to treat the content.

The most important browser headers, in terms of end-user performance, are:

1. HTTP version (HTTP/1.0 or HTTP/1.1)
2. Accept-Encoding: gzip, deflate
3. Connection: Keep-Alive
4. If-* headers
5. Cache-Control or Pragma no-cache

The first three items are interrelated. HTTP 1.0 does not include compression–indicated by the Accept-Encoding: gzip, deflate header, or connection keep-alives. Compression can reduce the byte count of text by 6:1 to 8:1. This often translates into a 40-50 percent reduction in size for a page. Connection: Keep-Alive will reuse TCP connections for subsequent requests and will save on the latency incurred by the 3-way hand-shake, and 4-way tear-down required for TCP connections on every request. Keeping connections open is important in emerging web-based applications that utilize Web 2.0 technology such as AJAX (Asynchronous JavaScript and XML) to perform real-time updates of content because it reduces the overhead associated with opening and closing TCP connections.

The various If-* headers, such as If-Modified-Since, will enable the web server to send a response that indicates the content has not been modified if this is true. This can potentially turn a 200KB download into a 1KB download, as the browser will respond to the 304 Not Modified response by loading the referenced content from the browser's cache. However, a lot of If-* requests for static content can result in unnecessary round trips. This can really slow end-user performance. The no-cache header and its relatives—no-store, private, must-revalidate, and proxy-revalidate—request that proxies and, sometimes, web servers not cache the response to the request. Honoring those requests can cause the servers to do a lot more work because they must always return the full content rather than enable the browser to use a cached version.

The most important web server headers, in terms of end-user performance, are:

1. The HTTP version (either HTTP/1.0 or HTTP/1.1) at the beginning of the status line
2. Connection: Keep-Alive/Close
3. Encoding: gzip, deflate
4. The various cache-control headers, especially max-age
5. Content-Type:
6. Date:
7. Accept-Ranges: bytes

Again, the first three items are inter-related and are meant to impart the same information as when sent by the browser. The cache-control headers are very important because they can be used to store items in the browser cache and avoid future HTTP requests altogether. However, using cached data runs the risk of using out-dated data if the content changes before the cached object expires. Content-type is important for telling the browser how to handle the object. This is most important for content that the browser hands off to plug-ins (Flash, Microsoft Office documents, etc.). It is also the biggest clue to the true function of that object in the web application. Improper content types will often result in slower, but not broken web applications. The Date header is very important because it affects how the browser interprets the cache-control headers. It is important to make sure the date on the server is set correctly so that this field is accurate. The Accept-Ranges header is only important when downloading PDF documents. It enables the browser to know that it can request the PDF document one page at a time.

Cookies

Cookies are sent by the web server to the browser as an HTTP header and used to store all sorts of information about a user's interaction with the site. Generally speaking the use of cookies will not affect the performance of an application, unless they are encrypted for security purposes. The reason encrypted cookies can affect performance is because the web server needs to decrypt them before use, and the encryption/decryption process is resource intensive. The more encrypted cookies that are used by a site, the longer it takes for the web server to process them into a readable format.

Vary

The HTTP Vary header, documented in RFC2616, is set by an origin web server (OWS) and contains request-header information. This information is used to determine whether a proxy server is permitted to reply to a subsequent request without re-validating the content from the OWS.

The BIG-IP HTTP cache (referred to as RAM Cache in BIG-IP versions prior to 11.0.0) uses the information from the Vary header to cache responses from the OWS. The OWS can include information within the Vary header to determine which resource the server returns in its response. For example, if a page is optimized for a particular web browser, the OWS response may return the Vary: User-Agent HTTP header. The proxy server then uses this information to determine whether to return a cached copy of the response to subsequent requests, or to query the OWS for the resource again (a subsequent client request containing a different User-Agent value forces the proxy to query the OWS for the resource again).

This behavior can require a proxy server (including the BIG-IP HTTP cache) to use up excess disk space to cache the same response.

For example:

Client A's request for a URI contains the following header:

```
User-Agent: agent1
```

The server's response includes the following headers:

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

The BIG-IP system then stores the page, noting the User-Agent and Accept-Encoding headers from the client's request.

Client B then requests the same URI, but the request has a User-Agent header containing agent2. The BIG-IP system ignores the existing cache entry (since the User-Agent is different), forwards the request to the server, and caches the response as a separate entry.

Beginning with BIG-IP 9.2, you can use the iRule CACHE::userkey <keystring> command to instruct the cache to cache the information based on the parameter that the administrator specifies. You can use this command to prevent multiple caches of the same information. Additionally, you can use the CACHE::useragent and CACHE::acceptencoding commands to override the behavior described in the previous example, such as, have a cache based on a group of User-Agent values rather than store an entry for each User-Agent header seen, and cause duplication.

For example, the following iRule sets the cache behavior based on the information that the User-Agent has on the customer's initial request, not on honoring User-Agent or Accept-Encoding when found in the server's Vary header:

```
when HTTP_REQUEST { set user_key "[HTTP::header User-Agent]" CACHE::userkey $user_
key }
```

> **Note:** The user_key can be defined as any string found in the HTTP request that the administrator wants to use to build cache responses.

You can use the previously listed iRule commands, even when the server does not set a Vary header, which allows the administrator to control the behavior outside of the server.

Content-Type

The MIME type of the body of the request (used with POST and PUT requests)

Host

The host value is represented by the domain name of the server (for virtual hosting), and the TCP port number on which the server is listening. The port number may be omitted if the port is the standard port for the service requested.

2.05c - Explain HTTP methods (GET, POST, etc.)

Fundamentals of HTTP
Hypertext Transfer Protocol – HTTP/1.1

HTTP Methods

When you open up a browser and request a web page (either by setting a default page or by entering a Uniform Resource Locater or URL), the first thing that happens is that the browser relies upon the operating system to resolve the host name in the URL to an IP address. Normally this is done via a DNS (Domain Name System) query over UDP (User Datagram Protocol) on port 53. However, if the host is listed in the local hosts file, the operating system will not make a DNS query.

When the IP address is obtained, the browser will attempt to open a TCP (Transmission Control Protocol) connection to the web server, usually on port 80. Once the TCP connection is made, the browser will issue an HTTP request to the server using the connection. The request comprises a header section, and possibly a body section (this is where things like POST data go). Once the request is sent, the browser will wait for the response. When the web server has assembled the response, it is sent back to the browser for rendering.

The base request comprises a method, the URI (Uniform Resource Indicator) of the web page or resource being requested, and the HTTP version desired (1.0 or 1.1). The method may be one of:

- Get
- Post
- Put

- Delete
- Head

Web servers almost universally support GET and POST, with the difference between them being the way in which query parameters are represented. With the GET method, all query parameters are part of the URI. This restricts the length of the parameters because a URI is generally limited to a set number of characters. Conversely, all parameters are included within the body of the request when using the POST method and there is usually no limit on the length of the body. PUT and DELETE, though considered important for emerging technology architectures such as REST (Representational State Transfer), are considered potentially dangerous as they enable the user to modify resources on the web server. These methods are generally disabled on web servers and not supported by modern web browsers.

The HTTP response consists of a header section and a body. The header section tells the browser how to treat the body content and the browser renders the content for viewing. Each HTTP response includes a status code, which indicates the status of the request. The most common status codes are:

200 OK. This indicates success

304 Not Modified. This shows that the resource in question has not changed and the browser should load it from its cache instead. This is only used when the browser performs a conditional GET request.

404 Not Found. This suggests that the resource requested cannot be found on the server.

401 Authorization Required. This indicates that the resource is protected and requires valid credentials before the server can grant access.

500 Internal Error. This signifies that the server had a problem processing the request.

While most developers do not need to know these status codes as they are not used within D/HTML, AJAX (Asynchronous Javascript and XML) developers may need to recognize these codes as part of their development efforts.

Most HTTP responses will also contain references to other objects within the body that will cause the browser to automatically request these objects as well. Web pages often contain more than 30 other object references required to complete the page.

When retrieving these referenced objects, the default browser behavior is to open two TCP connections per host seen in the references. With Internet Explorer there is a Windows registry setting that limits this to a total of eight TCP connections. There is a similar setting in Firefox, but its maximum is 24 TCP connections.

Get

The GET method means retrieve whatever information (in the form of an entity) is identified by the Request-URI. If the Request-URI refers to a data-producing process, it is the produced data, which shall be returned as the entity in the response and not the source text of the process, unless that text happens to be the output of the process.

The semantics of the GET method change to a "conditional GET" if the request message includes an If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match, or If-Range header field. A conditional GET method requests that the entity be transferred only under the circumstances described by the conditional header field(s). The conditional GET method is intended to reduce unnecessary network usage by allowing cached entities to be refreshed without requiring multiple requests or transferring data already held by the client.

The semantics of the GET method change to a "partial GET" if the request message includes a Range header field. A partial GET requests that only part of the entity be transferred. The partial GET method is intended to reduce unnecessary network usage by allowing partially retrieved entities to be completed without transferring data already held by the client.

The response to a GET request is cacheable if and only if it meets the requirements for HTTP caching.

Put

The PUT method requests that the enclosed entity be stored under the supplied Request-URI. If the Request-URI refers to an already existing resource, the enclosed entity SHOULD be considered as a modified version of the one residing on the origin server. If the Request-URI does not point to an existing resource, and that URI is capable of being defined as a new resource by the requesting user agent, the origin server can create the resource with that URI. If a new resource is created, the origin server MUST inform the user agent via the 201 (Created) response. If an existing resource is modified, either the 200 (OK) or 204 (No Content) response codes SHOULD be sent to indicate successful completion of the request. If the resource could not be created or modified with the Request-URI, an appropriate error response SHOULD be given that reflects the nature of the problem. The recipient of the entity MUST NOT ignore any Content-* (e.g. Content-Range) headers that it does not understand or implement and MUST return a 501 (Not Implemented) response in such cases.

If the request passes through a cache and the Request-URI identifies one or more currently cached entities, those entries SHOULD be treated as stale. Responses to this method are not cacheable.

The fundamental difference between the POST and PUT requests is reflected in the different meaning of the Request-URI. The URI in a POST request identifies the resource that will handle the enclosed entity. That resource might be a data-accepting process, a gateway to some other protocol, or a separate entity that accepts annotations. In contrast, the URI in a PUT request identifies the entity enclosed with the request – the user agent knows what URI is intended and the server MUST NOT attempt to apply the request to some other resource. If the server desires that the request be applied to a different URI,

It MUST send a 301 (Moved Permanently) response; the user agent MAY then make its own decision regarding whether or not to redirect the request.

Many different URIs MAY identify a single resource. For example, an article might have a URI for identifying "the current version" which is separate from the URI identifying each particular version. In this case, a PUT request on a general URI might result in several other URIs being defined by the origin server.

HTTP/1.1 does not define how a PUT method affects the state of an origin server.

Unless otherwise specified for a particular entity-header, the entity-headers in the PUT request SHOULD be applied to the resource created or modified by the PUT.

2.05d - Explain the difference between HTTP versions (i.e., 1.0 and 1.1)

Key Differences between HTTP/1.0 and HTTP/1.1

HTTP 1.0 and HHTP 1.1 Differences

The HTTP Protocol was not written by F5 so there is not much in-house documentation on topics like this one comparing the different versions. So I found this site document that does a fantastic job with comparing HTTP1.0 and HTTP 1.1. There are many references in the linked site document that you can go check out as well but for the sake of flow I removed them from this section.

In this section we describe the major changes between the HTTP/1.0 and HTTP/1.1 protocols. The HTTP/1.1 specification is almost three times as long as RFC1945, reflecting an increase in complexity, clarity, and specificity. Even so, the HTTP/1.1 specification, rather than being explicitly stated implies numerous rules. While some attempts have been made to document the differences between HTTP/1.0 and HTTP/1.1 ([Mar97,Yap97], Section 19.6.1 of [FGM+98]), we know of no published analysis that covers major differences and the rationale behind them, and that reflects the most recent (and probably near-final) revision of the HTTP/1.1 specification. Because the HTTP-WG, a large and international group of researchers and developers, conducted most of its discussions via its mailing list, the archive of that list [CLF98] documents the history of the HTTP/1.1 effort. But that archive contains over 8500 messages, rendering it opaque to all but the most determined protocol historian.

We structure our discussion by (somewhat arbitrarily) dividing the protocol changes into nine major areas:

1. Extensibility
2. Caching
3. Bandwidth optimization
4. Network connection management

5. Message transmission
6. Internet address conservation
7. Error notification
8. Security, integrity, and authentication
9. Content negotiation

We devote a section to each area, including the motivation for changes and a description of the corresponding new features.

Extensibility

The HTTP/1.1 effort assumed, from the outset, that compatibility with the installed base of HTTP implementations was mandatory. It seemed unlikely that most software vendors or Web site operators would deploy systems that failed to interoperate with the millions of existing clients, servers, and proxies.

Because the HTTP/1.1 effort took over four years, and generated numerous interim draft documents, many implementers deployed systems using the `HTTP/1.1'' protocol version before the final version of the specification was finished. This created another compatibility problem: the final version had to be substantially compatible with these pseudo-HTTP/1.1 versions, even if the interim drafts turned out to have errors in them.

These absolute requirements for compatibility with poorly specified prior versions led to a number of idiosyncrasies and non-uniformities in the final design. It is not possible to understand the rationale for all of the HTTP/1.1 features without recognizing this point.

The compatibility issue also underlined the need to include, in HTTP/1.1, as much support as possible for future extensibility. That is, if a future version of HTTP were to be designed, it should not be hamstrung by any additional compatibility problems.

Note that HTTP has always specified that if an implementation receives a header that it does not understand, it must ignore the header. This rule allows a multitude of extensions without any change to the protocol version, although it does not by itself support all possible extensions.

Version numbers

In spite of the confusion over the meaning of the `HTTP/1.1'' protocol version token (does it imply compatibility with one of the interim drafts, or with the final standard?), in many cases the version number in an HTTP message can be used to deduce the capabilities of the sender. A companion document to the HTTP specification clearly specified the ground rules for the use and interpretation of HTTP version numbers.

The version number in an HTTP message refers to the hop-by-hop sender of the message, not the end-to-end sender. Thus the message's version number is directly useful in determining hop-by-hop message-level

capabilities, but not very useful in determining end-to-end capabilities. For example, if an HTTP/1.1 origin server receives a message forwarded by an HTTP/1.1 proxy, it cannot tell from that message whether the ultimate client uses HTTP/1.0 or HTTP/1.1.

For this reason, as well as to support debugging, HTTP/1.1 defines a Via header that describes the path followed by a forwarded message. The path information includes the HTTP version numbers of all senders along the path and is recorded by each successive recipient. (Only the last of multiple consecutive HTTP/1.0 senders will be listed, because HTTP/1.0 proxies will not add information to the Via header.)

The OPTIONS method

HTTP/1.1 introduces the OPTIONS method, a way for a client to learn about the capabilities of a server without actually requesting a resource. For example, a proxy can verify that the server complies with a specific version of the protocol. Unfortunately, the precise semantics of the OPTIONS method were the subject of an intense and unresolved debate, and we believe that the mechanism is not yet fully specified.

Upgrading to other protocols

In order to ease the deployment of incompatible future protocols, HTTP/1.1 includes the new Upgrade request-header. By sending the Upgrade header, a client can inform a server of the set of protocols it supports as an alternate means of communication. The server may choose to switch protocols, but this is not mandatory.

Caching

Web developers recognized early on that the caching of responses was both possible and highly desirable. Caching is effective because a few resources are requested often by many users, or repeatedly by a given user. Caches are employed in most Web browsers and in many proxy servers; occasionally they are also employed in conjunction with certain origin servers. Web caching products, such as Cisco's cache engine [Cis] and Inktomi's Traffic Server [Ink] (to name two), are now a major business.

Many researchers have studied the effectiveness of HTTP caching. Caching improves user-perceived latency by eliminating the network communication with the origin server. Caching also reduces bandwidth consumption, by avoiding the transmission of unnecessary network packets. Reduced bandwidth consumption also indirectly reduces latency for uncached interactions, by reducing network congestion. Finally, caching can reduce the load on origin servers (and on intermediate proxies), further improving latency for uncached interactions.

One risk with caching is that the caching mechanism might not be ``semantically transparent'': that is, it might return a response different from what would be returned by direct communication with the origin server. While some applications can tolerate non-transparent responses, many Web applications (electronic commerce, for example) cannot.

Caching in HTTP/1.0

HTTP/1.0 provided a simple caching mechanism. An origin server may mark a response, using the Expires header, with a time until which a cache could return the response without violating semantic transparency. Further, a cache may check the current validity of a response using what is known as a conditional request: it may include an If-Modified-Since header in a request for the resource, specifying the value given in the cached response's Last-Modified header. The server may then either respond with a 304 (Not Modified) status code, implying that the cache entry is valid, or it may send a normal 200 (OK) response to replace the cache entry.

HTTP/1.0 also included a mechanism, the Pragma: no-cache header, for the client to indicate that a request should not be satisfied from a cache.

The HTTP/1.0 caching mechanism worked moderately well, but it had many conceptual shortcomings. It did not allow either origin servers or clients to give full and explicit instructions to caches; therefore, it depended on a body of heuristics that were not well specified. This led to two problems: incorrect caching of some responses that should not have been cached, and failure to cache some responses that could have been cached. The former causes semantic problems; the latter causes performance problems.

Caching in HTTP/1.1

HTTP/1.1 attempts to clarify the concepts behind caching, and to provide explicit and extensible protocol mechanisms for caching. While it retains the basic HTTP/1.0 design, it augments that design both with new features, and with more careful specifications of the existing features.

In HTTP/1.1 terminology, a cache entry is fresh until it reaches its expiration time, at which point it becomes stale. A cache need not discard a stale entry, but it normally must revalidate it with the origin server before returning it in response to a subsequent request. However, the protocol allows both origin servers and end-user clients to override this basic rule.

In HTTP/1.0, a cache revalidated an entry using the If-Modified-Since header. This header uses absolute timestamps with one-second resolution, which could lead to caching errors either because of clock synchronization errors, or because of lack of resolution. Therefore, HTTP/1.1 introduces the more general concept of an opaque cache validator string, known as an entity tag. If two responses for the same resource have the same entity tag, then they must (by specification) be identical. Because an entity tag is opaque, the origin server may use any information it deems necessary to construct it (such as a fine-grained timestamp or an internal database pointer), as long as it meets the uniqueness requirement. Clients may compare entity tags for equality, but cannot otherwise manipulate them. HTTP/1.1 servers attach entity tags to responses using the ETag header.

HTTP/1.1 includes a number of new conditional request-headers, in addition to If-Modified-Since. The most basic is If-None-Match, which allows a client to present one or more entity tags from its cache entries for

a resource. If none of these matches the resource's current entity tag value, the server returns a normal response; otherwise, it may return a 304 (Not Modified) response with an ETag header that indicates which cache entry is currently valid. Note that this mechanism allows the server to cycle through a set of possible responses, while the If-Modified-Since mechanism only generates a cache hit if the most recent response is valid.

HTTP/1.1 also adds new conditional headers called If-Unmodified-Since and If-Match, creating other forms of preconditions on requests. These preconditions are useful in more complex situations; in particular, see the discussion in Section 4.1 of Range requests.

The Cache-Control header

In order to make caching requirements more explicit, HTTP/1.1 adds the new Cache-Control header, allowing an extensible set of cache-control directives to be transmitted in both requests and responses. The set defined by HTTP/1.1 is quite large, so we concentrate on several notable members.

Because the absolute timestamps in the HTTP/1.0 Expires header can lead to failures in the presence of clock skew (and observations suggest that serious clock skew is common), HTTP/1.1 can use relative expiration times, via the max-age directive. (It also introduces an Age header, so that caches can indicate how long a response has been sitting in caches along the way.)

Because some users have privacy requirements that limit caching beyond the need for semantic transparency, the private and no-store directives allow servers and clients to prevent the storage of some or all of a response. However, this does not guarantee privacy; only cryptographic mechanisms can provide true privacy.

Some proxies transform responses (for example, to reduce image complexity before transmission over a slow link), but because some responses cannot be blindly transformed without losing information, the no-transform directive may be used to prevent transformations.

The Vary header

A cache finds a cache entry by using a key value in a lookup algorithm. The simplistic caching model in HTTP/1.0 uses just the requested URL as the cache key. However, the content negotiation mechanism (described in Section 10) breaks this model, because the response may vary not only based on the URL, but also based on one or more request-headers (such as Accept-Language and Accept-Charset).

To support caching of negotiated responses, and for future extensibility, HTTP/1.1 includes the Vary response-header. This header field carries a list of the relevant selecting request-header fields that participated in the selection of the response variant. In order to use the particular variant of the cached response in replying to a subsequent request, the selecting request-headers of the new request must exactly match those of the original request.

This simple and elegant extension mechanism works for many cases of negotiation, but it does not allow for much intelligence at the cache. For example, a smart cache could, in principle, realize that one request header

value is compatible with another, without being equal. The HTTP/1.1 development effort included an attempt to provide so-called `transparent content negotiation'' that would allow caches some active participation, but ultimately no consensus developed, and this attempt was separated from the HTTP/1.1 specification.

Bandwidth optimization

Network bandwidth is almost always limited. Both by intrinsically delaying the transmission of data, and through the added queuing delay caused by congestion, wasting bandwidth increases latency. HTTP/1.0 wastes bandwidth in several ways that HTTP/1.1 addresses. A typical example is a server's sending an entire (large) resource when the client only needs a small part of it. There was no way in HTTP/1.0 to request partial objects. Also, it is possible for bandwidth to be wasted in the forward direction: if a HTTP/1.0 server could not accept large requests, it would return an error code after bandwidth had already been consumed. What was missing was the ability to negotiate with a server and to ensure its ability to handle such requests before sending them.

Range requests

A client may need only part of a resource. For example, it may want to display just the beginning of a long document, or it may want to continue downloading a file after a transfer was terminated in mid-stream. HTTP/1.1 range requests allow a client to request portions of a resource. While the range mechanism is extensible to other units (such as chapters of a document, or frames of a movie), HTTP/1.1 supports only ranges of bytes. A client makes a range request by including the Range header in its request, specifying one or more contiguous ranges of bytes. The server can either ignore the Range header, or it can return one or more ranges in the response.

If a response contains a range, rather than the entire resource, it carries the 206 (Partial Content) status code. This code prevents HTTP/1.0 proxy caches from accidentally treating the response as a full one, and then using it as a cached response to a subsequent request. In a range response, the Content-Range header indicates the offset and length of the returned range, and the new multipart/byteranges MIME type allows the transmission of multiple ranges in one message.

Range requests can be used in a variety of ways, such as:

1. to read the initial part of an image, to determine its geometry and therefore do page layout without loading the entire image.
2. to complete a response transfer that was interrupted (either by the user or by a network failure); in other words, to convert a partial cache entry into a complete response.
3. to read the tail of a growing object.

Some of these forms of range request involve cache conditionals. That is, the proper response may depend on the validity of the client's cache entry (if any).

For example, the first kind (getting a prefix of the resource) might be done unconditionally, or it might be done with an If-None-Match header; the latter implies that the client only wants the range if the underlying object has changed, and otherwise will use its cache entry.

The second kind of request, on the other hand, is made when the client does not have a cache entry that includes the desired range. Therefore, the client wants the range only if the underlying object has not changed; otherwise, it wants the full response. This could be accomplished by first sending a range request with an If-Match header, and then repeating the request without either header if the first request fails. However, since this is an important optimization, HTTP/1.1 includes an If-Range header, which effectively performs that sequence in a single interaction.

Range requests were originally proposed by Ari Luotonen and John Franks [FL95], using an extension to the URL syntax instead of a separate header field. However, this approach proved less general than the approach ultimated used in HTTP/1.1, especially with respect to conditional requests.

Expect and 100 (Continue)

Some HTTP requests (for example, the PUT or POST methods) carry request bodies, which may be arbitrarily long. If, the server is not willing to accept the request, perhaps because of an authentication failure, it would be a waste of bandwidth to transmit such a large request body.

HTTP/1.1 includes a new status code, 100 (Continue), to inform the client that the request body should be transmitted. When this mechanism is used, the client first sends its request headers, then waits for a response. If the response is an error code, such as 401 (Unauthorized), indicating that the server does not need to read the request body, the request is terminated. If the response is 100 (Continue), the client can then send the request body, knowing that the server will accept it.

However, HTTP/1.0 clients do not understand the 100 (Continue) response. Therefore, in order to trigger the use of this mechanism, the client sends the new Expect header, with a value of 100-continue. (The Expect header could be used for other, future purposes not defined in HTTP/1.1.)

Because not all servers use this mechanism (the Expect header is a relatively late addition to HTTP/1.1, and early ``HTTP/1.1'' servers did not implement it), the client must not wait indefinitely for a 100 (Continue) response before sending its request body. HTTP/1.1 specifies a number of somewhat complex rules to avoid either infinite waits or wasted bandwidth. We lack sufficient experience based on deployed implementations to know if this design will work efficiently.

Compression

One well-known way to conserve bandwidth is through the use of data compression. While most image formats (GIF, JPEG, MPEG) are pre-compressed, many other data types used in the Web are not. One study showed that aggressive use of additional compression could save almost 40% of the bytes sent via HTTP.

While HTTP/1.0 included some support for compression, it did not provide adequate mechanisms for negotiating the use of compression, or for distinguishing between end-to-end and hop-by-hop compression.

HTTP/1.1 makes a distinction between content-codings, which are end-to-end encodings that might be inherent in the native format of a resource, and transfer-codings, which are always hop-by-hop. Compression can be done either as a content-coding or as a transfer-coding. To support this choice, and the choice between various existing and future compression codings, without breaking compatibility with the installed base, HTTP/1.1 had to carefully revise and extend the mechanisms for negotiating the use of codings.

HTTP/1.0 includes the Content-Encoding header, which indicates the end-to-end content-coding(s) used for a message; HTTP/1.1 adds the Transfer-Encoding header, which indicates the hop-by-hop transfer-coding(s) used for a message.

HTTP/1.1 (unlike HTTP/1.0) carefully specifies the Accept-Encoding header, used by a client to indicate what content-codings it can handle, and which ones it prefers. One tricky issue is the need to support ``robot'' clients that are attempting to create mirrors of the origin server's resources; another problem is the need to interoperate with HTTP/1.0 implementations, for which Accept-Encoding was poorly specified.

HTTP/1.1 also includes the TE header, which allows the client to indicate which transfer-codings are acceptable, and which are preferred. Note that one important transfer-coding, Chunked, has a special function (not related to compression), and is discussed further in Section 6.1.

Network connection management

HTTP almost always uses TCP as its transport protocol. TCP works best for long-lived connections, but the original HTTP design used a new TCP connection for each request, so each request incurred the cost of setting up a new TCP connection (at least one round-trip time across the network, plus several overhead packets). Since most Web interactions are short (the median response message size is about 4 Kbytes), the TCP connections seldom get past the ``slow-start'' region [Jac88] and therefore fail to maximize their use of the available bandwidth.

Web pages frequently have embedded images; sometimes many of them, and each image is retrieved via a separate HTTP request. The use of a new TCP connection for each image retrieval serializes the display of the entire page on the connection-setup latencies for all of the requests. Netscape introduced the use of parallel TCP connections to compensate for this serialization, but the possibility of increased congestion limits the utility of this approach.

To resolve these problems you can incorporate use of persistent connections and the pipelining of requests on a persistent connection.

The Connection header

Before discussing persistent connections, we address a more basic issue. Given the use of intermediate proxies, HTTP makes a distinction between the end-to-end path taken by a message, and the actual hop-by-hop connection between two HTTP implementations.

HTTP/1.1 introduces the concept of hop-by-hop headers: message headers that apply only to a given connection, and not to the entire path. (For example, we have already described the hop-by-hop Transfer-Encoding and TE headers.) The use of hop-by-hop headers creates a potential problem: if such a header were to be forwarded by a naive proxy, it might mislead the recipient.

Therefore, HTTP/1.1 includes the Connection header. This header lists all of the hop-by-hop headers in a message, telling the recipient that these headers must be removed from that message before it is forwarded. This extensible mechanism allows the future introduction of new hop-by-hop headers; the sender need not know whether the recipient understands a new header in order to prevent the recipient from forwarding the header.

Because HTTP/1.0 proxies do not understand the Connection header, however, HTTP/1.1 imposes an additional rule. If a Connection header is received in an HTTP/1.0 message, then an HTTP/1.0 proxy must have incorrectly forwarded it. Therefore, all of the headers it lists were also incorrectly forwarded, and must be ignored.

The Connection header may also list connection-tokens, which are not headers but rather per-connection Boolean flags. For example, HTTP/1.1 defines the token close to permit the peer to indicate that it does not want to use a persistent connection. Again, the Connection header mechanism prevents these tokens from being forwarded.

Persistent Connections

HTTP/1.0, in its documented form, made no provision for persistent connections. Some HTTP/1.0 implementations, however, use a Keep-Alive header to request that a connection persist. This design did not interoperate with intermediate proxies; HTTP/1.1 specifies a more general solution.

In recognition of their desirable properties, HTTP/1.1 makes persistent connections the default. HTTP/1.1 clients, servers, and proxies assume that a connection will be kept open after the transmission of a request and its response. The protocol does allow an implementation to close a connection at any time, in order to manage its resources, although it is best to do so only after the end of a response.

Because an implementation may prefer not to use persistent connections if it cannot efficiently scale to large numbers of connections or may want to cleanly terminate one for resource-management reasons, the protocol permits it to send a Connection: close header to inform the recipient that the connection will not be reused.

Pipelining

Although HTTP/1.1 encourages the transmission of multiple requests over a single TCP connection, each request must still be sent in one contiguous message, and a server must send responses (on a given connection) in the order that it received the corresponding requests. However, a client need not wait to receive the response for one request before sending another request on the same connection. In fact, a client could send an arbitrarily large number of requests over a TCP connection before receiving any of the responses. This practice, known as pipelining, can greatly improve performance. It avoids the need to wait for network round-trips, and it makes the best possible use of the TCP protocol.

Message transmission

HTTP messages may carry a body of arbitrary length. The recipient of a message needs to know where the message ends. The sender can use the Content-Length header, which gives the length of the body. However, many responses are generated dynamically, by CGI [CGI98] processes and similar mechanisms. Without buffering the entire response (which would add latency), the server cannot know how long it will be and cannot send a Content-Length header.

When not using persistent connections, the solution is simple: the server closes the connection. This option is available in HTTP/1.1, but it defeats the performance advantages of persistent connections.

The Chunked transfer-coding

HTTP/1.1 resolves the problem of delimiting message bodies by introducing the Chunked transfer-coding. The sender breaks the message body into chunks of arbitrary length, and each chunk is sent with its length prepended; it marks the end of the message with a zero-length chunk. The sender uses the Transfer-Encoding: chunked header to signal the use of chunking.

This mechanism allows the sender to buffer small pieces of the message, instead of the entire message, without adding much complexity or overhead. All HTTP/1.1 implementations must be able to receive chunked messages.

The Chunked transfer-coding solves another problem, not related to performance. In HTTP/1.0, if the sender does not include a Content-Length header, the recipient cannot tell if the message has been truncated due to transmission problems. This ambiguity leads to errors, especially when truncated responses are stored in caches.

Trailers

Chunking solves another problem related to sender-side message buffering. Some header fields, such as Content-MD5 (a cryptographic checksum over the message body), cannot be computed until after the message body is generated. In HTTP/1.0, the use of such header fields required the sender to buffer the entire message.

In HTTP/1.1, a chunked message may include a trailer after the final chunk. A trailer is simply a set of one or more header fields. By placing them at the end of the message, the sender allows itself to compute them after generating the message body.

The sender alerts the recipient to the presence of message trailers by including a Trailer header, which lists the set of headers deferred until the trailer. This alert, for example, allows a browser to avoid displaying a prefix of the response before it has received authentication information carried in a trailer.

HTTP/1.1 imposes certain conditions on the use of trailers, to prevent certain kinds of interoperability failure. For example, if a server sends a lengthy message with a trailer to an HTTP/1.1 proxy that is forwarding the response to an HTTP/1.0 client, the proxy must either buffer the entire message or drop the trailer. Rather than insist that proxies buffer arbitrarily long messages, which would be infeasible, the protocol sets rules that should prevent any critical information in the trailer (such as authentication information) from being lost because of this problem. Specifically, a server cannot send a trailer unless either the information it contains is purely optional, or the client has sent a TE: trailers header, indicating that it is willing to receive trailers (and, implicitly, to buffer the entire response if it is forwarding the message to an HTTP/1.0 client).

Transfer-length issues

Several HTTP/1.1 mechanisms, such as Digest Access Authentication (see Section 9.1), require end-to-end agreement on the length of the message body; this is known as the entity-length. Hop-by-hop transfer-codings, such as compression or chunking, can temporarily change the transfer-length of a message. Before this distinction was clarified, some earlier implementations used the Content-Length header indiscriminately.

Therefore, HTTP/1.1 gives a lengthy set of rules for indicating and inferring the entity-length of a message. For example, if a non-identity transfer-coding is used (so the transfer-length and entity-length differ), the sender is not allowed to use the Content-Length header at all. When a response contains multiple byte ranges, using Content-Type: multipart/byteranges, then this self-delimiting format defines the transfer-length.

Internet address conservation

Companies and organizations use URLs to advertise themselves and their products and services. When a URL appears in a medium other than the Web itself, people seem to prefer ``pure hostname'' URLs; i.e., URLs without any path syntax following the hostname. These are often known as ``vanity URLs,'' but in spite of the implied disparagement, it's unlikely that non-purist users will abandon this practice, which has led to the continuing creation of huge numbers of hostnames.

IP addresses are widely perceived as a scarce resource (pending the uncertain transition to IPv6). The Domain Name System (DNS) allows multiple host names to be bound to the same IP address. Unfortunately, because the original designers of HTTP did not anticipate the ``success disaster'' they were enabling, HTTP/1.0 requests do not pass the hostname part of the request URL. For example, if a user makes a request for the resource at URL http://example1.org/home.html, the browser sends a message with the Request-Line

GET /home.html HTTP/1.0

to the server at example1.org. This prevents the binding of another HTTP server hostname, such as exampleB.org to the same IP address, because the server receiving such a message cannot tell which server the message is meant for. Thus, the proliferation of vanity URLs causes a proliferation of IP address allocations.

The Internet Engineering Steering Group (IESG), which manages the IETF process, insisted that HTTP/1.1 take steps to improve conservation of IP addresses. Since HTTP/1.1 had to interoperate with HTTP/1.0, it could not change the format of the Request-Line to include the server hostname. Instead, HTTP/1.1 requires requests to include a Host header, first proposed by John Franks [Fra94], that carries the hostname. This converts the example above to:

GET /home.html HTTP/1.1

Host: example1.org

If the URL references a port other than the default (TCP port 80), this is also given in the Host header.

Clearly, since HTTP/1.0 clients will not send Host headers, HTTP/1.1 servers cannot simply reject all messages without them. However, the HTTP/1.1 specification requires that an HTTP/1.1 server must reject any HTTP/1.1 message that does not contain a Host header.

The intent of the Host header mechanism, and in particular the requirement that enforces its presence in HTTP/1.1 requests, is to speed the transition away from assigning a new IP address for every vanity URL. However, as long as a substantial fraction of the users on the Internet use browsers that do not send Host, no Web site operator (such as an electronic commerce business) that depends on these users will give up a vanity-URL IP address. The transition, therefore, may take many years. It may be obviated by an earlier transition to IPv6, or by the use of market mechanisms to discourage the unnecessary consumption of IPv4 addresses.

Error notification

HTTP/1.0 defined a relatively small set of sixteen status codes, including the normal 200 (OK) code. Experience revealed the need for finer resolution in error reporting.

The Warning header

HTTP status codes indicate the success or failure of a request. For a successful response, the status code cannot provide additional advisory information, in part because the placement of the status code in the Status-Line, instead of in a header field, prevents the use of multiple status codes.

HTTP/1.1 introduces a Warning header, which may carry any number of subsidiary status indications. The intent is to allow a sender to advise the recipient that something may be unsatisfactory about an ostensibly successful response.

HTTP/1.1 defines an initial set of Warning codes, mostly related to the actions of caches along the response path. For example, a Warning can mark a response as having been returned by a cache during disconnected operation, when it is not possible to validate the cache entry with the origin server.

The Warning codes are divided into two classes, based on the first digit of the 3-digit code. One class of warnings must be deleted after a successful revalidation of a cache entry; the other class must be retained with a revalidated cache entry. Because this distinction is made based on the first digit of the code, rather than through an exhaustive listing of the codes, it is extensible to Warning codes defined in the future.

Other new status codes

There are 24 new status codes in HTTP/1.1; we have discussed 100 (Continue), 206 (Partial Content), and 300 (Multiple Choices) elsewhere in this paper. A few of the more notable additions include

- 409 (Conflict), returned when a request would conflict with the current state of the resource. For example, a PUT request might violate a versioning policy.
- 410 (Gone), used when a resource has been removed permanently from a server, and to aid in the deletion of any links to the resource.

Most of the other new status codes are minor extensions.

Security, integrity, and authentication

In recent years, the IETF has heightened its sensitivity to issues of privacy and security. One special concern has been the elimination of passwords transmitted ``in the clear.'' This increased emphasis has manifested itself in the HTTP/1.1 specification (and other closely related specifications).

Digest access authentication

HTTP/1.0 provides a challenge-response access control mechanism, Basic authentication. The origin server responds to a request for which it needs authentication with a WWW-Authenticate header that identifies the authentication scheme (in this case, ``Basic'') and realm. (The realm value allows a server to partition sets of resources into ``protection spaces,'' each with its own authorization database.)

The client (user agent) typically queries the user for a username and password for the realm, then repeats the original request, this time including an Authorization header that contains the username and password. Assuming these credentials are acceptable to it, the origin server responds by sending the expected content. A client may continue to send the same credentials for other resources in the same realm on the same server, thus eliminating the extra overhead of the challenge and response.

A serious flaw in Basic authentication is that the username and password in the credentials are unencrypted and therefore vulnerable to network snooping. The credentials also have no time dependency, so they could be collected at leisure and used long after they were collected. Digest access authentication provides a

simple mechanism that uses the same framework as Basic authentication while eliminating many of its flaws. (Digest access authentication, being largely separable from the HTTP/1.1 specification, has developed in parallel with it.)

The message flow in Digest access authentication mirrors that of Basic and uses the same headers, but with a scheme of ``Digest.'' The server's challenge in Digest access authentication uses a nonce (one-time) value, among other information. To respond successfully, a client must compute a checksum (MD5, by default) of the username, password, nonce, HTTP method of the request, and the requested URI. Not only is the password no longer unencrypted, but the given response is correct only for a single resource and method. Thus, an attacker that can snoop on the network could only replay the request, the response for which he has already seen. Unlike with Basic authentication, obtaining these credentials does not provide access to other resources.

As with Basic authentication, the client may make further requests to the same realm and include Digest credentials, computed with the appropriate request method and request-URI. However, the origin server's nonce value may be time-dependent. The server can reject the credentials by saying the response used a stale nonce and by providing a new one. The client can then recompute its credentials without needing to ask the user for username and password again.

In addition to the straightforward authentication capability, Digest access authentication offers two other features: support for third-party authentication servers, and a limited message integrity feature (through the Authentication-Info header).

Proxy authentication

Some proxy servers provide service only to properly authenticated clients. This prevents, for example, other clients from stealing bandwidth from an unsuspecting proxy.

To support proxy authentication, HTTP/1.1 introduces the Proxy-Authenticate and Proxy-Authorization headers. They play the same role as the WWW-Authenticate and Authorization headers in HTTP/1.0, except that the new headers are hop-by-hop, rather than end-to-end. Proxy authentication may use either of the Digest or Basic authentication schemes, but the former is preferred.

A proxy server sends the client a Proxy-Authenticate header, containing a challenge, in a 407 (Proxy Authentication Required) response. The client then repeats the initial request, but adds a Proxy-Authorization header that contains credentials appropriate to the challenge. After successful proxy authentication, a client typically sends the same Proxy-Authorization header to the proxy with each subsequent request, rather than wait to be challenged again.

Protecting the privacy of URIs

The URI of a resource often represents information that some users may view as private. Users may prefer not to have it widely known that they have visited certain sites.

The Referer [sic] header in a request provides the server with the URI of the resource from which the request-URI was obtained. This gives the server information about the user's previous page-view. To protect against unexpected privacy violations, the HTTP/1.1 specification takes pains to discourage sending the Referer header inappropriately; for example, when a user enters a URL from the keyboard, the application should not send a Referer header describing the currently-visible page, nor should a client send the Referer header in an insecure request if the referring page had been transferred securely.

The use of a GET-based HTML form causes the encoding of form parameters in the request-URI. Most proxy servers log these request-URIs. To protect against revealing sensitive information, such as passwords or credit-card numbers, in a URI, the HTTP/1.1 specification strongly discourages the use of GET-based forms for submitting such data. The use of POST-based forms prevents the form parameters from appearing in a request-URI, and therefore from being logged inappropriately.

Content-MD5

MIME's Content-MD5 header contains the MD5 digest of the entity being sent [MR95]. The HTTP/1.1 specification provides specific rules for the use of this header in the Web, which differ slightly from its use in MIME (electronic mail). The sender may choose to send Content-MD5 so the recipient can detect accidental changes to the entity during its transmission. Content-MD5 is a good example of a header that a server might usefully send as a trailer.

Clearly the Content-MD5 value could easily be spoofed and cannot serve as a means of security. Also, because Content-MD5 covers just the entity in one message, it cannot be used to determine if a full response has been successfully reassembled from a number of partial (range) responses, or whether response headers have been altered.

State management

HTTP requests are stateless. That is, from a server's perspective, each request can ordinarily be treated as independent of any other. For Web applications, however, state can sometimes be useful. For example, a shopping application would like to keep track of what is in a user's ``shopping basket,'' as the basket's contents change over the course of a sequence of HTTP requests.

Netscape introduced ``cookies'' [Net] in version 1.1 of their browser as a state management mechanism. The IETF subsequently standardized cookies in RFC2109 [KM97]. (The cookie specification is another example of how HTTP can be extended by a separate specification without affecting the core protocol. Cookie support is optional in servers and user agents, although some Web-based services will not work in their absence.)

The basic cookie mechanism is simple. An origin server sends an arbitrary piece of (state) information to the client in its response. The client is responsible for saving the information and returning it with its next request to the origin server. RFC2109 and Netscape's original specification relax this model so that a cookie can be returned to any of a collection of related servers, rather than just to one. The specifications also restricts for

which URIs on a given server the cookie may be returned. A servers may assign a lifetime to a cookie, after which it is no longer used.

Cookies have both privacy and security implications. Because their content is arbitrary, cookies may contain sensitive application-dependent information. For example, they could contain credit card numbers, user names and passwords, or other personal information. Applications that send such information over unencrypted connections leave it vulnerable to snooping, and cookies stored at a client system might reveal sensitive information to another user of (or intruder into) that client.

RFC2109 proved to be controversial, primarily because of restrictions that were introduced to protect privacy. Probably the most controversial of these has to do with `unverifiable transactions'' and `third-party cookies.'' Consider this scenario.

1. The user visits http://www.example1.com/home.html.
2. The returned page contains an IMG (image) tag with a reference to http://ad.example.com/adv1.gif, an advertisement.
3. The user's browser automatically requests the image. The response includes a cookie from ad.example.com.
4. The user visits http://www.exampleB.com/home.html.
5. The returned page contains an IMG tag with a reference to http://ad.example.com/adv2.gif.
6. The user's browser automatically requests the image, sending the previously received cookie to ad.example.com in the process. The response includes a new cookie from ad.example.com.

Privacy advocates, and others, worried that:

- the user receives, in step 3, a (`third-party'') cookie from ad.example.com, a site she didn't even know she was going to visit (an `unverifiable transaction''); and
- that first cookie gets returned to ad.example.com in the second image request (step 6).

If a Referer header is sent with each of the image requests to ad.example.com, then that site can begin to accumulate a profile of the user's interests from the sites she visited, here http://www.example1.com/home.html and http://www.exampleB.com/home.html. Such an advertising site could potentially select advertisements that are likely to be interesting to her. While that profiling process is relatively benign in isolation, it could become more personal if the profile can also be tied to a specific real person, not just a persona. For example, this might happen if the user goes through some kind of registration at www.example1.com.

RFC2109 sought to limit the possible pernicious effects of cookies by requiring user agents to reject cookies that arrive from the responses to unverifiable transactions. RFC2109 further stated that user agents could be

configured to accept such cookies, provided that the default was not to accept them. This default setting was a source of concern for advertising networks (companies that run sites like ad.example.com in the example) whose business model depended on cookies, and whose business blossomed in the interval between when the specification was essentially complete (July, 1996) and the time it appeared as an RFC (February, 1997). RFC2109 has undergone further refinement in response to comments, both political and technical.

Content Negotiation

Web users speak many languages and use many character sets. Some Web resources are available in several variants to satisfy this multiplicity. HTTP/1.0 included the notion of content negotiation, a mechanism by which a client can inform the server which language(s) and/or character set(s) are acceptable to the user.

Content negotiation has proved to be a contentious and confusing area. Some aspects that appeared simple at first turned out to be quite difficult to resolve. For example, although current IETF practice is to insist on explicit character set labeling in all relevant contexts, the existing HTTP practice has been to use a default character set in most contexts, but not all implementations chose the same default. The use of unlabeled defaults greatly complicates the problem of internationalizing the Web.

HTTP/1.0 provided a few features to support content negotiation, but RFC1945 [BLFF96] never uses that term and devotes less than a page to the relevant protocol features. The HTTP/1.1 specification specifies these features with far greater care, and introduces a number of new concepts.

The goal of the content negotiation mechanism is to choose the best available representation of a resource. HTTP/1.1 provides two orthogonal forms of content negotiation, differing in where the choice is made:

1. In server-driven negotiation, the more mature form, the client sends hints about the user's preferences to the server, using headers such as Accept-Language, Accept-Charset, etc. The server then chooses the representation that best matches the preferences expressed in these headers.

2. In agent-driven negotiation, when the client requests a varying resource, the server replies with a 300 (Multiple Choices) response that contains a list of the available representations and a description of each representation's properties (such as its language and character set). The client (agent) then chooses one representation, either automatically or with user intervention, and resubmits the request, specifying the chosen variant.

Although the HTTP/1.1 specification reserves the Alternates header name for use in agent-driven negotiation, the HTTP working group never completed a specification of this header, and server-driven negotiation remains the only usable form.

Some users may speak multiple languages, but with varying degrees of fluency. Similarly, a Web resource might be available in its original language, and in several translations of varying faithfulness. HTTP introduces the use of quality values to express the importance or degree of acceptability of various negotiable

parameters. A quality value (or qvalue) is a fixed-point number between 0.0 and 1.0. For example, a native speaker of English with some fluency in French, and who can impose on a Danish-speaking office-mate, might configure a browser to generate requests including

Accept-Language: en, fr;q=0.5, da;q=0.1

Because the content-negotiation mechanism allows qvalues and wildcards, and expresses variation across many dimensions (language, character-set, content-type, and content-encoding) the automated choice of the ``best available'' variant can be complex and might generate unexpected outcomes. These choices can interact with caching in subtle ways; see the discussion in Section 3.4.

Content negotiation promises to be a fertile area for additional protocol evolution. For example, the HTTP working group recognized the utility of automatic negotiation regarding client implementation features, such as screen size, resolution, and color depth. The IETF has created the Content Negotiation working group to carry forward with work in the area.

Objective - 2.06 Given a set of headers or traces, determine the root cause of an HTTP/HTTPS application problem

2.06a - Explain how to interpret response codes

See section 2.05a

This topic content is the same as topic 2.05a – simply applied to the 2.06 objective.

2.06b - Explain the function of HTTP headers within different HTTP applications (Cookies, Cache Control, Vary, Content Type & Host)

See section 2.05b

This topic content is the same as topic 2.05b – simply applied to the 2.06 objective.

2.06c - Explain HTTP methods (GET, POST, etc.)

See section 2.05c

This topic content is the same as topic 2.05c – simply applied to the 2.06 objective.

2.06d - Explain the difference between HTTP versions (i.e., 1.0 and 1.1)

See section 2.05d

This topic content is the same as topic 2.05d – simply applied to the 2.06 objective.

2.06e - Explain how to unencrypt HTTPS data

Overview of Packet Tracing with the ssldump Utility

Decrypting HTTPS Data

The Secure Socket Layer (SSL) protocol is used to encrypt sensitive data for transmission on the Internet. It is commonly used to encrypt HTTP based traffic know as HTTPS traffic. Doing SSL termination on the BIG-IP platform could decrypt HTTPS data, but this function is not always configured. If you capture HTTPS traffic and need visibility to solve a technical issue, it may be helpful to decrypt the application data to better understand the issue. The ssldump utility is an SSL/TLS network protocol analyzer, which identifies TCP connections from a chosen packet trace or network interface and attempts to interpret them as SSL/TLS traffic. When the ssldump utility identifies SSL/TLS traffic, it decodes the records and displays them in text to standard output. If provided with the private key that was used to encrypt the connections, the ssldump utility may also be able to decrypt the connections and display the application data traffic.

You can use the ssldump utility to examine, decrypt, and decode SSL-encrypted packet streams managed by the BIG-IP system. The ssldump utility can act on packet streams real-time as they traverse the system, or on a packet capture file saved in the libpcap format, such as that produced by the tcpdump utility. Although it is possible for the ssldump utility to decode and display live traffic real-time as it traverses the BIG-IP system, it is rarely the most effective method to examine the voluminous and complex output of the ssldump utility. Capturing the target traffic to a file using the tcpdump utility, then decoding the file using the ssldump utility offers a better opportunity to examine the traffic in detail.

Examining the decrypted application data

When troubleshooting SSL issues, it may be necessary to examine the decrypted application data. To do this, the ssldump utility must have access to either the (asymmetric) private key from the server you want to debug, or the (symmetric) pre-master secret keys. Using the pre-master secret keys allows you to examine the decrypted application data in Wireshark without having access to the (asymmetric) private key.

> **Warning:** Decrypting the SSL application data may expose sensitive information such as credit card numbers and passwords.

Examining the decrypted application data using the (symmetric) pre-master secret keys

Beginning in BIG-IP 11.2.0, the ssldump -M option allows you to create a pre-master secret (PMS) key log file. You can load the PMS log file into Wireshark (1.6 and later) along with the capture file, and use it to decrypt the application data without having access to the server's private key. This option gives F5 Technical Support the ability to fully decrypt sessions in the targeted capture file without revealing sensitive information about the private key.

To run ssldump using the -M option to create a pre-master secret key log file, perform the following procedure:

1. Log in to the BIG-IP command line.
2. Capture the packet trace containing the SSL traffic (see the Capturing the target traffic section).
3. To create a pre-master secret key log file, use the following ssldump syntax:

   ```
   ssldump -r /path/to/capture_file -k /path/to/private_key -M /path/to/pre-master-key_log_file
   ```

   For example, the following ssldump command reads the www-ssl-client1.cap capture file using the test.org key file to decrypt the session, and creates the PMS log file called client1.pms:

   ```
   ssldump -r /var/tmp/www-ssl-client1.cap -k /config/filestore/files_d/Common_d/certificate_key_d/\:Common\:test.org.key_1 -M /var/tmp/client1.pms
   ```

   You can now load the capture file and the PMS log file into Wireshark (1.6 and later).

   Note: To load the pre-master secret (PMS) key log file in Wireshark, navigate to Edit > Preferences > Protocols > SSL, and enter the path and filename of the PMS key in the (Pre)-Master-Secret log filename field.

   Note: For F5 Technical Support to examine the decrypted application data, you must submit the capture file and PMS key log file.

Examining the decrypted application data using the (asymmetric) private key

To decrypt display application data, the ssldump utility will need a copy of the private key from the server you want to debug. If you do not have the key, the application data cannot be decrypted, and you will only be able to examine and decode the SSL handshake packets.

Important: Not all ciphers provide the ability to decrypt SSL traffic using a utility such as ssldump. Depending on the cipher negotiated, the ssldump utility may not be able to derive enough information from the SSL handshake and the server's private key to decrypt the application data. Examples of such SSL ciphers would be the Diffie-Hellman Ephemeral (DHE) cipher suites and export-grade RSA

cipher suites. If it is necessary to decrypt application data for a virtual server, you can change the cipher suite used in your SSL profile to accept only those ciphers. To do so, make a note of the cipher string currently configured in the SSL profile, then temporarily modify the SSL profile to specify NONE:RC4+RSA as the custom cipher string. For specific configuration steps, refer to the examples in SOL7815: Configuring the cipher strength for SSL profiles.

In the previous example, the ssldump command that is provided prints the application_data record type but does not display the application data itself. Since no key was provided, the application data has not been decrypted. To print the decrypted application data, use the -k option to specify the path and name of the file containing the server's private key.

For example:

```
ssldump -Aed -nr /var/tmp/www-ssl-client.cap -k /config/ssl/ssl.key/www-ssl.key
```

Note: In BIG-IP 11.x, the SSL profile keys are stored in the `/config/filestore/files_d/<partition_name>_d/certificate_key_d/` directory.

Decoded application data records printed by ssldump appear similar to the following example:

```
 3 7 1247932735.6286 (0.0031) C>SV3.1(436) application_data

   ---------------------------------------------------

   GET / HTTP/1.1

   Accept: image/gif, image/jpeg, image/pjpeg, image/pjpeg,
application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-
powerpoint, application/msword, */*

   Accept-Language: en-us

   User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0;
InfoPath.2; .NET CLR 1.1.4322; msn OptimizedIE8;ENUS)

   Accept-Encoding: gzip, deflate

   Host: 172.24.72.169

   Connection: Keep-Alive

 3 8 1247932735.8015 (0.1728) S>CV3.1(1540) application_data

   ---------------------------------------------------
```

```
HTTP/1.1 200 OK

Date: Sat, 18 Jul 2009 15:25:37 GMT

Server: Apache/2.2.3 (Debian) PHP/4.4.4-8+etch4 mod_ssl/2.2.3 OpenSSL/0.9.8g
mod_perl/2.0.2 Perl/v5.8.8

Last-Modified: Tue, 08 Jan 2008 22:59:06 GMT

ETag: "3306ee-8be-ec750280"

Accept-Ranges: bytes

Content-Length: 2238

Keep-Alive: timeout=15, max=100

Connection: Keep-Alive

Content-Type: text/html
```

Locating a BIG-IP virtual server's private key

The private key that the BIG-IP system uses for a virtual server is specified in the Client SSL profile applied to the virtual server. The key file locations are listed below:

- BIG-IP 9.x - 10.x:
  `/config/ssl/ssl.key/`

  For example:
  `/config/ssl/ssl.key/www.site.com.crt`

- BIG-IP 11. x:
  `/config/filestore/files_d/<partition_name>_d/certificate_key_d/`

  For example:
  `/config/filestore/files_d/Common_d/certificate_key_d/:Common:www.site.com.key_1`

To determine the key used for a particular virtual server, examine the virtual server configuration to determine the name of the SSL profiles applied to the virtual server and then examine the SSL profile configuration to determine the name of the key file.

2.06f - Explain how to decode POST data

Fully Decode URI

Decode POST data

Decoding POST data has to be done using an iRule. We can use the URI::decode command which will returns a URI decoded version of a given URI.

When checking for special characters in a URI (quotes, percent signs, spaces, etc), one must make sure that the URI has been fully decoded. This could require multiple decoding's to cover for the fact that characters within an encoded character could themselves be encoded.

```
when HTTP_REQUEST {

   # Decode original URI
   set tmpUri [HTTP::uri]

   set uri [URI::decode $tmpUri]

   # For versions lower than 10.2 HF2, set a max number of iterations
   # to handle bug ID 337562
   set max 4
   set cnt 0

   # Repeat decoding until the decoded version equals the previous value
   # or max iterations is reached
   while { $uri ne $tmpUri && $cnt < $max } {
      set tmpUri $uri
      set uri [URI::decode $tmpUri]
      incr cnt
   }
   HTTP::uri $uri

   log local0.  "Original URI: [HTTP::uri]"
   log local0.  "Fully decoded URI: $uri"
}
```

Objective - 2.07 Given a set of headers or traces, determine a solution to an HTTP/HTTPS application problem

2.07a - Explain how to interpret response codes

See section 2.05a

This topic content is the same as topic 2.05a – simply applied to the 2.07 objective.

2.07b - Explain circumstances under which HTTP chunking is required

Overview of HTTP Chunking

HTTP Chunking

Chunking is a technique that HTTP servers use to improve responsiveness. Chunking can help you avoid situations where the server needs to obtain dynamic content from an external source and delays sending the response to the client until receiving all of the content so the server can calculate a Content-Length header.

When chunking is enabled, instead of delaying sending packets to the client until all content is available, the server will:

- Send the response in chunks.
- Add a Transfer-Encoding: chunked HTTP header to the chunks.
- Apply markers within the content to indicate the length of each chunk and whether that particular chunk is the last chunk that the server is sending.

Some operations involve modifying content, such as adding content using an iRule, or applying compression. These operations need to first remove chunking (unchunk), perform the operation, and optionally reapply chunking (rechunk) to the new content.

The HTTP profile has four modes available to specify how the system handles HTTP content with regards to chunking. The behavior in each mode depends on whether the server sends chunked or unchunked responses.

Chunked content

The following specifications apply to content that is chunked:

- Unchunk
  Specifies that the system removes the HTTP transfer encoding headers, removes the chunk headers, processes the HTTP content, and then sends the unchunked response to the client.
- Rechunk
  Specifies that the system unchunks the HTTP content, processes the data, re-adds the chunk headers, and then sends the chunked request or response to the client.
- Selective
  Specifies that the system unchunks the HTTP content, processes the data, re-adds the chunk headers, and then sends the chunked request or response to the client.

  Note: For chunked content, this mode is the same as the Rechunk mode.
- Preserve
  Specifies that the system processes the HTTP content, and sends the response to the client, unchanged.

Unchunked content

The following specifications apply to content that is unchunked:

- Unchunk
  Specifies that the system processes the HTTP content, and sends the response to the client, unchanged.
- Rechunk
  Specifies that the system processes the HTTP content, adds the transfer encoding and chunk headers to the response, and then sends the chunked response to the client.
- Selective
  Specifies that the system processes the HTTP content, and sends the response to the client, unchanged.
- Preserve
  Specifies that the system processes the HTTP content, and sends the response to the client, unchanged.

In BIG-IP LTM 9.4.0 and later, the default response chunking mode in the HTTP profile is selective. Selective mode unchunks and rechunks data only if the payload is being modified.

2.07c - Explain the function of HTTP headers within different HTTP applications (Cookies, Cache Control, Vary, Content Type & Host)

See section 2.05b

This topic content is the same as topic 2.05b – simply applied to the 2.06 objective.

2.07d - Explain HTTP methods (GET, POST, etc.)

See section 2.05c

This topic content is the same as topic 2.05c – simply applied to the 2.06 objective.

2.07e - Explain the difference between HTTP versions (i.e., 1.0 and 1.1)

See section 2.05d

This topic content is the same as topic 2.05d – simply applied to the 2.06 objective.

2.07f - Explain how to unencrypt HTTPS data

See section 2.06e

This topic content is the same as topic 2.06e – simply applied to the 2.06 objective.

2.07g - Explain how to decode POST data

See section 2.06f

This topic content is the same as topic 2.06f – simply applied to the 2.06 objective.

Objective - 2.08 Given a direct trace and a trace through the LTM device, compare the traces to determine the root cause of an HTTP/HTTPS application problem

2.08a - Explain how to interpret response codes

See section 2.05a

This topic content is the same as topic 2.05a – simply applied to the 2.06 objective.

2.08b - Explain the function of HTTP headers within different HTTP applications (Cookies, Cache Control, Vary, Content Type & Host)

See section 2.05b

This topic content is the same as topic 2.05b – simply applied to the 2.06 objective.

2.08c - Explain HTTP methods (GET, POST, etc.)

See section 2.05c

This topic content is the same as topic 2.05c – simply applied to the 2.06 objective.

2.08d - Explain the difference between HTTP versions (i.e., 1.0 and 1.1)

See section 2.05d

This topic content is the same as topic 2.05d – simply applied to the 2.06 objective.

2.08e - Explain how to unencrypt HTTPS data

See section 2.06e

This topic content is the same as topic 2.06e – simply applied to the 2.06 objective.

2.08f - Explain how to decode POST data

See section 2.06f

This topic content is the same as topic 2.06f – simply applied to the 2.06 objective.

2.08g - Given a set of circumstances determine which persistence is required

Session Persistence Profiles

Session Persistence

Using BIG-IP Local Traffic Manager (LTM), you can configure session persistence. When you configure session persistence, LTM tracks and stores session data, such as the specific pool member that serviced a client request. The primary reason for tracking and storing session data is to ensure that client requests are directed to the same pool member throughout the life of a session or during subsequent sessions.

In addition, session persistence can track and store other types of information, such as user preferences or a user name and password.

LTM offers several types of session persistence, each one designed to accommodate a specific type of storage requirement for session data. The type of persistence that you implement depends on where and how you want to store client-specific information, such as items in a shopping cart or airline ticket reservations.

For example, you might store airline ticket reservation information in a back-end database that all servers can access, or on the specific server to which the client originally connected, or in a cookie on the client's machine. When you enable persistence, returning clients can bypass load balancing and instead connect to the server to which they last connected in order to access their saved information.

Local Traffic Manager keeps session data for a period of time that you specify.

The primary tool for configuring session persistence is to configure a persistence profile and assign it to a virtual server. If you want to enable persistence for specific types of traffic only, as opposed to all traffic passing through the virtual server, you can write an iRule.

To configure and manage persistence profiles, log in to the BIG-IP Configuration utility, and on the Main tab, expand Local Traffic, and click Persistence.

What are session persistence profiles?

A persistence profile is a pre-configured object that automatically enables persistence when you assign the profile to a virtual server. By using a persistence profile, you avoid having to write a program to implement a type of persistence.

Each type of persistence that Local Traffic Manager offers includes a corresponding default persistence profile. These persistence profiles each contain settings and setting values that define the behavior of the BIG-IP system for that type of persistence. You can either use the default profile or create a custom profile based on the default.

Persistence profile types

You can configure persistence profile settings to set up session persistence on the BIG-IP system. You can configure these settings when you create a profile or after profile creation by modifying the profiles settings.

The persistence types that you can enable using a persistence profile are:

- Cookie persistence
  Cookie persistence uses an HTTP cookie stored on a client's computer to allow the client to reconnect to the same server previously visited at a web site.
- Destination address affinity persistence
  Also known as sticky persistence, destination address affinity persistence supports TCP and UDP protocols, and directs session requests to the same server based solely on the destination IP address of a packet.
- Hash persistence
  Hash persistence allows you to create a persistence hash based on an existing iRule.
- Microsoft Remote Desktop Protocol persistence
  Microsoft Remote Desktop Protocol (MSRDP) persistence tracks sessions between clients and servers running the Microsoft Remote Desktop Protocol (RDP) service.
- SIP persistence
  SIP persistence is a type of persistence used for servers that receive Session Initiation Protocol (SIP) messages sent through UDP, SCTP, or TCP.
- Source address affinity persistence
  Also known as simple persistence, source address affinity persistence supports TCP and UDP protocols, and directs session requests to the same server based solely on the source IP address of a packet.

- SSL persistence
  SSL persistence is a type of persistence that tracks non-terminated SSL sessions, using the SSL session ID. Even when the clients IP address changes, Local Traffic Manager still recognizes the connection as being persistent based on the session ID. Note that the term non-terminated SSL sessions refers to sessions in which Local Traffic Manager does not perform the tasks of SSL certificate authentication and encryption/re-encryption.

- Universal persistence
  Universal persistence allows you to write an expression that defines what to persist on in a packet. The expression, written using the same expression syntax that you use in iRules, defines some sequence of bytes to use as a session identifier.

Session persistence and iRules

Instead of configuring a persistence profile, which enables a persistence type for all sessions passing through the virtual server, you can write an iRule, which enables a persistence type for particular requests (for example, for HTTP traffic that includes a certain cookie version only).

You can also use an iRule to enable persistence for SSL-terminated requests, that is, requests that Local Traffic Manager terminates by performing decryption and re-encryption and by handling SSL certificate authentication. In this type of iRule, you can use an HTTP header insertion iRule command to insert an SSL session ID as a header into an HTTP request.

The remainder of this chapter focuses on enabling persistence using persistence profiles. For information on enabling persistence by writing an iRule, see the F5 Networks DevCentral web site http://devcentral.f5.com, and Chapter 18, iRules.

The OneConnect profile and session persistence

When you configure session persistence, Local Traffic Manager tracks and stores session data, such as the pool member that serviced a client request. Configuring a persistence profile for a virtual server ensures that client requests are directed to the same pool member throughout the life of a session or during subsequent sessions.

The Request-URI header in an HTTP request stores certain session data. Occasionally, however, for Cookie and Universal persistence types specifically, Local Traffic Manager ignores the session data in this header, and sends requests to an unexpected node. For example, this issue can occur when clients send requests to a virtual server through an internet proxy device. You can prevent this problem by creating an OneConnect™ profile, and assigning both the OneConnect profile and the persistence profile to the virtual server.

The following two sections explain the effect of an OneConnect profile on session persistence.

HTTP parsing without an OneConnect profile

If the virtual server does not reference an OneConnect profile, Local Traffic Manager performs load balancing for each TCP connection. Once the TCP connection is load balanced, the system sends all requests that are part of the connection to the same pool member.

For example, if the virtual server does not reference an OneConnect profile, and Local Traffic Manager initially sends a client request to node A in pool A, the system inserts a cookie for node A. Then, within the same TCP connection, if Local Traffic Manager receives a subsequent request that contains a cookie for node B in pool B, the system ignores the cookie information and incorrectly sends the request to node A instead.

HTTP parsing using an OneConnect profile

Using an OneConnect type of profile solves the problem. If the virtual server references an OneConnect profile, Local Traffic Manager can perform load balancing for each request within the TCP connection. That is, when an HTTP client sends multiple requests within a single connection, Local Traffic Manager is able to process each HTTP request individually. Local Traffic Manager sends the HTTP requests to different destination servers if necessary.

For example, if the virtual server references an OneConnect profile and the client request is initially sent to node A in pool A, Local Traffic Manager inserts a cookie for node A. Then, within the same TCP connection, if Local Traffic Manager receives a subsequent request that contains a cookie for node B in pool B, the system uses that cookie information and correctly sends the request to node B.

Troubleshooting tips

To mitigate issues when Local Traffic Manager ignores persistence information in the Request-URI header and therefore sends requests to an incorrect node, you can take these actions:

- Associate an OneConnect profile with the virtual server.

  Verify that the OneConnect Transformations setting in the HTTP profile is enabled.

Criteria for session persistence

Regardless of the type of persistence you are implementing, you can specify the criteria that Local Traffic Manager uses to send all requests from a given client to the same pool member. These criteria are based on the virtual server or servers that are hosting the client connection. To specify these criteria, you use the Match Across Services, Match Across Virtual Servers, and Match Across Pools profile settings. Before configuring a persistence profile, it is helpful to understand these settings.

The Match Across Services setting

When you enable the Match Across Services profile setting, Local Traffic Manager attempts to send all persistent connection requests received from the same client, within the persistence time limit, to the same node only when the virtual server hosting the connection has the same virtual address as the virtual server hosting the initial persistent connection. Connection requests from the client that go to other virtual servers with different virtual addresses, or those connection requests that do not use persistence, are load balanced according to the load balancing method defined for the pool.

For example, suppose you configure virtual server mappings where the virtual server v1:http has persistence enabled and references the http_pool (containing the nodes n1:http and n2:http), and the virtual server v1:ssl has persistence enabled and references the pool ssl_pool (containing the nodes n1:ssl and n2:ssl).

Suppose the client makes an initial connection to v1:http, and the load balancing algorithm assigned to the pool http_pool chooses n1:http as the node. If the client subsequently connects to v1:ssl, Local Traffic Manager uses the persistence session established with the first connection to determine the pool member that should receive the connection request, rather than the load balancing method. Local Traffic Manager should then send the third connection request to n1:ssl, which uses the same node as the n1:http node that currently hosts the client's first connection with which it shares a persistent session.

If the same client then connects to a virtual server with a different virtual address (for example, v2:ssl), Local Traffic Manager starts tracking a new persistence session, using the load balancing method to determine which node should receive the connection request. The system starts a new persistence session because the requested virtual server uses a different virtual address (v2) than the virtual server hosting the first persistent connection request (v1).

> **Important:** In order for the Match Across Services setting to be effective, virtual servers that use the same virtual address, as well as those that use SSL persistence, should include the same node addresses in the virtual server mappings.

> **Note:** With respect to Cookie profiles, this setting applies to the Cookie Hash method only.

The Match Across Virtual Servers setting

You can set Local Traffic Manager to maintain persistence for all sessions requested by the same client, regardless of which virtual server hosts each individual connection initiated by the client. When you enable the Match Across Virtual Servers setting, Local Traffic Manager attempts to send all persistent connection requests received from the same client, within the persistence time limit, to the same node. Connection requests from the client that do not use persistence are load balanced according to the currently selected load balancing method.

> **Note:** With respect to Cookie profiles, this setting applies to the Cookie Hash method only.

> **Warning:** In order for this setting to be effective, virtual servers that use pools with TCP or SSL persistence should include the same member addresses in the virtual server mappings.

The Match Across Pools setting

When you enable the Match Across Pools setting, Local Traffic Manager can use any pool that contains a given persistence record. The default is disabled (cleared).

> **Warning:** Enabling this setting can cause Local Traffic Manager to direct client traffic to a pool other than that specified by the virtual server.

With respect to Cookie profiles, this setting applies to the Cookie Hash method only.

Objective - 2.09 Given a direct trace and a trace through the LTM device, compare the traces to determine the root cause of an HTTP/HTTPS application problem

2.09a - Explain how to interpret response codes

See section 2.05a

This topic content is the same as topic 2.05a – simply applied to the 2.09 objective.

2.09b - Explain circumstances under which HTTP chunking is required

See section 2.07b

This topic content is the same as topic 2.07b – simply applied to the 2.09 objective.

2.09c - Explain the function of HTTP headers within different HTTP applications (Cookies, Cache Control, Vary, Content Type & Host)

See section 2.05b

This topic content is the same as topic 2.05b – simply applied to the 2.09 objective.

2.09d - Explain HTTP methods (GET, POST, etc.)

See section 2.05c

This topic content is the same as topic 2.05c – simply applied to the 2.09 objective.

2.09e - Explain the difference between HTTP versions (i.e., 1.0 and 1.1)

See section 2.05d

This topic content is the same as topic 2.05d – simply applied to the 2.09 objective.

2.09f - Explain how to unencrypt HTTPS data

See section 2.06e

This topic content is the same as topic 2.06e – simply applied to the 2.09 objective.

2.09g - Explain how to decode POST data

See section 2.06f

This topic content is the same as topic 2.06f – simply applied to the 2.09 objective.

2.09h - Given a set of circumstances determine which persistence is required

See section 2.08g

This topic content is the same as topic 2.08g – simply applied to the 2.09 objective.

Objective - 2.10 Given a scenario, determine which protocol analyzer tool and its options are required to resolve an application issue

Scenario Based Objective

To prepare for scenario based questions the candidate will need to complete hands-on configuration and testing of the configuration on the LTM. This will allow the candidate to better understand how different configurations can produce different results. All F5 exams use scenario-based questions that make the candidate apply what they know to a situation to determine the resulting outcome.

This topic is focused on using an analyzer to resolve application issues.

2.10a - Explain how to use the advanced flags in the protocol analyzers (e.g., tcpdump, -e, -s, -v, -X)

TCPDUMP

tcpdump

The tcpdump utility is a command line packet sniffer with many features and options. For a full description, refer to the tcpdump man pages by typing the following command:

man tcpdump

There are many different flags/switches that can be used with the tcpdump command. You can read the full list in the man pages for tcpdump. Some of the common flags you should be familiar with are below:

-e

Print the link-level header on each dump line. This can be used, for example, to print MAC layer addresses for protocols such as Ethernet and IEEE 802.11.

-s

`--snapshot-length=snaplen`

Snarf snaplen bytes of data from each packet rather than the default of 65535 bytes. Packets truncated because of a limited snapshot are indicated in the output with ``[|proto]'', where proto is the name of the protocol level at which the truncation has occurred. Note that taking larger snapshots both increases the amount of time it takes to process packets and, effectively, decreases the amount of packet buffering. This may cause packets to be lost. You should limit snaplen to the smallest number that will capture the

protocol information you're interested in. Setting snaplen to 0 sets it to the default of 65535, for backwards compatibility with recent older versions of tcpdump.

-v

When parsing and printing, produce (slightly more) verbose output. For example, the time to live, identification, total length and options in an IP packet are printed. Also enables additional packet integrity checks such as verifying the IP and ICMP header checksum.

When writing to a file with the -w option, report, every 10 seconds, the number of packets captured.

-x

When parsing and printing, in addition to printing the headers of each packet, print the data of each packet (minus its link level header) in hex and ASCII. This is very handy for analysing new protocols.

Parameter	Meaning
`-i [interface]`	Listen on interface or VLAN name
`-e`	Print the link-level header on each dump line
`-n`	Don't convert host addresses to names
`-X`	Print each packet (minus link level header) in hex and ASCII
`-r [file]`	Read packets from *file* (created with -w option)
`-w [file]`	Write raw packets to file (rather than printing)
`-c [count]`	Exit tcpdump after receiving count packets
`-s [snaplen]`	Snarf snaplen bytes of data from each packet
`host [ip address]`	Capture packets if either source or destination address is host
`port [service]`	Capture packets if either source or destination port is port
`and | or | not`	Used to form expressions to select which packets to dump

2.10b - Explain how to decrypt SSL traffic for protocol analysis

Overview of Packet Tracing with the ssldump Utility

ssldump

The Secure Socket Layer (SSL) protocol is used to encrypt sensitive data for transmission on the Internet. If a BIG-IP LTM system is contributing to a technical issue, it may be helpful to decrypt the application data to better understand the issue. The ssldump utility is an SSL/TLS network protocol analyzer, which identifies TCP connections from a chosen packet trace or network interface and attempts to interpret them as SSL/TLS

traffic. When the ssldump utility identifies SSL/TLS traffic, it decodes the records and displays them in text to standard output. If provided with the private key that was used to encrypt the connections, the ssldump utility may also be able to decrypt the connections and display the application data traffic.

You can use the ssldump utility to examine, decrypt, and decode SSL-encrypted packet streams managed by the BIG-IP system. The ssldump utility can act on packet streams real-time as they traverse the system, or on a packet capture file saved in the libpcap format, such as that produced by the tcpdump utility. Although it is possible for the ssldump utility to decode and display live traffic real-time as it traverses the BIG-IP system, it is rarely the most effective method to examine the voluminous and complex output of the ssldump utility. Capturing the target traffic to a file using the tcpdump utility, then decoding the file using the ssldump utility offers a better opportunity to examine the traffic in detail.

2.10c - Explain how to use DB keys to enhance the amount of data collected with protocol analyzers

Capturing Internal TMM Information with tcpdump

DB keys and tcpdump

A number of the BIG-IP BigDB keys provide the ability to enable verbose logging, which can be useful for troubleshooting specific BIG-IP issues.

The F5 implementation of the tcpdump utility can add internal TMM information to a tcpdump capture. In the course of a support case, an F5 Technical Support engineer may ask you to capture a tcpdump where this extra information is present, or you may want to collect the data yourself for analysis in a tool such as Wireshark.

The enhanced tcpdump utility can capture extra details, such as what virtual server and what TMM is handling a specific sample of traffic. When reviewing the tcpdump output file in Wireshark, this extra information appears under the Ethernet II section in the Packet Details panel. The linked site has procedures for capturing extended TMM data with tcpdump.

2.10d - Determine which protocol analyzer options are safe to use based on traffic load and hardware model

Recommended Methods and Limitations for Running tcpdump on a BIG-IP System

Running tcpdump on a busy system

> **Important:** The BIG-IP system is designed as an application delivery network platform and not as a packet capture device. If you intend to capture traffic under high load conditions, F5 recommends that you mirror traffic to a dedicated sniffing device.

Running tcpdump on a BIG-IP system is considered best effort, as it will place more load on the CPU and may result in inaccuracies in the tcpdump output, such as missed packets or packet timestamp irregularities.

If you run tcpdump on a heavily loaded BIG-IP system, the packet capture process may not capture all matching traffic, and the statistical values reported by tcpdump may be inaccurate. For example, the packets received by filter statistic may not provide an accurate reporting of all matching packets in the network. And the packets dropped by kernel statistic may not provide an accurate reporting of the real number of matching packets that have been dropped.

If you run tcpdump on a heavily loaded system, F5 recommends that you use tcpdump filter expressions to mitigate the potential for missed packets.

Running tcpdump on a VLAN

Recommendations

- F5 recommends that you run tcpdump on a VLAN when you intend to capture traffic for in-depth troubleshooting on the BIG-IP system. When the VLAN is specified in the tcpdump syntax, tcpdump can read packets processed by TMM.
- Starting in BIG-IP 11.x, if you run tcpdump on a VLAN that resides in a non-default partition, you must specify the path to the VLAN object in the tcpdump syntax.

  For example, the tcpdump syntax would appear similar to the following example:
  `tcpdump -ni /<partition_name>/<vlan_name>`

Limitations

- For systems containing a PVA ASIC, the tcpdump utility does not capture virtual server traffic that is fully accelerated by the PVA chip. The PVA resides on the switchboard, between the BIG-IP system's switch subsystem and the host motherboard. The chip processes accelerated traffic by accepting packets from the switch subsystem, transforming the packets to redirect them to the appropriate pool member, and then sending the packets back to the switch subsystem. Fully accelerated traffic never reaches the internal trunk and is not processed by TMM.

  For example, the following command does not capture PVA accelerated traffic:
  `tcpdump -ni <vlan_name>`

  Note: To determine whether your platform contains a PVA chip, use the tmsh show /sys hardware |grep -i pva command for BIG-IP 11.x, or the bigpipe platform |grep -i pva command for BIG-IP 9.x through 10.x.

  You can work around this limitation by temporarily disabling PVA acceleration for the FastL4 profile, capturing the traffic in a VLAN tcpdump, and then re-enabling PVA acceleration for the FastL4 profile.

Important: Temporarily disabling PVA acceleration for a FastL4 profile may significantly increase CPU utilization for the BIG-IP LTM system. Use this option only if the BIG-IP LTM is capable of handling your site's level of traffic without using the PVA.

Running tcpdump on an interface

Recommendations

- F5 recommends that you run only a tcpdump on an interface in the BIG-IP system when performing basic connectivity troubleshooting. When you run tcpdump on an interface, the packet is copied on a switch ingress to the SCCP or AOM, which then sends the packet to the host to be captured by tcpdump.
- Running tcpdump on interface 0.0 is not rate-limited like running tcpdump on a hardware switch interface. Capturing traffic on interface 0.0 has the potential to create very large files. Running tcpdump on interface 0.0 is recommended only for troubleshooting when filtering to limit the size of the capture.

Limitations

- Running tcpdump on a hardware switch interface is rate-limited to 200 packets per second. Therefore, if you run tcpdump on an interface that is processing more than 200 packets per second, the captured tcpdump file does not include all of the packets.
- For example, the following command captures PVA-accelerated traffic, but the syntax results in a rate limit of 200 packets per second:
  `tcpdump -ni <interface_number>`
- If the specified interface is a member of a trunk, tcpdump captures all traffic flowing through the trunk, not just the traffic traversing the specified interface.
- When tcpdump is run on an interface on a VIPRION system, the tcpdump command must be run on the same blade in which the interface belongs. For example, if you are logged into blade 1 and you want to run tcpdump on interface 2/2.1, you must log into blade 2 before running the tcpdump command.

Running tcpdump in a route domain

Recommendations

- When using tcpdump to capture traffic in a non-default route domain, F5 recommends that you run the tcpdump command from the default route domain (route domain 0), and specify interface 0.0.

  Note: Specifying interface 0.0 when running tcpdump captures traffic traversing all configured VLANs on the BIG-IP system.

For example, the following command captures traffic from all VLANs in all route domains when invoked from the default route domain:

```
tcpdump -ni 0.0
```

Limitations

- The tcpdump utility does not capture traffic when run from a non-default route domain. For example, if you use the rdsh utility to change the shell to a non-default route domain and run the tcpdump command, no traffic will be captured. To capture traffic, use the following command to change back to the default route domain:

  ```
  rdsh 0
  ```

  You can then run the tcpdump -ni 0.0 command to capture all route domain traffic.

Objective - 2.11 Given a trace, determine the root cause of an application problem

2.11a - Identify application issues based on a protocol analyzer trace

Packet Trace Analysis

Packet trace analysis

Note: This article provides a checklist that you can use when you analyze packet traces. In this article, F5 assumes that you have a working knowledge of tcpdump. For more information about tcpdump, refer to the tcpdump man page and SOL411: Overview of packet tracing with the tcpdump utility. If you are working with SSL encrypted packets, you can also refer to SOL10209: Overview of packet tracing with the ssldump utility.

Note: As an alternative to analyzing the trace by manually using tcpdump on the BIG-IP or FirePass system, it may be helpful to download the packet trace to a workstation that runs the Wireshark packet analyzer with the F5 Wireshark plug-in. To download Wireshark, refer to the Download Wireshark page; this link takes you to a resource outside of AskF5, and it is possible that the information may be removed without our knowledge.

To analyze a packet trace you have already captured and saved for analysis, you can perform the following tasks:

Reviewing and isolating bad connections

Review the packet trace and isolate the bad sessions/connections, which typically contain one or more of the following symptoms:

- A large number of packets sent in the same direction, with no response
- Long delays (view the time stamps)
- Packets with the reset (RST) flag set

  Note: For more information, refer to SOL9812: Overview of BIG-IP TCP RST behavior.

Troubleshooting bad connections

Review the last page of each connection that you have isolated, and perform the following checks:

- Verify sequence numbers by looking for the following occurrences:
  - Packets out of sequence
  - Dropped packets (look for a missing sequence space)
  - Duplicate packets
- Verify that all data (not packets) has been acknowledged (ACK)
- Determine reasons for reset (RST) connections:
  - If the connection has an acknowledgment (ACK) number, use the tcpdump - S command to see which packet was RST
  - Determine where the RST is originating from (client/server/BIG-IP)
  - Determine the reason for the RST by asking the following questions:
    - Was the connection reaped by the client/server/BIG-IP?
    - Did a user cancel the connection by pressing stop in the browser?
    - Was it an out-of-sequence packet?

Identifying the source of the issue

Identify the specific client, server, or device that is most likely causing the issue by performing the following tasks and answering the related questions:

- Review the checksum/sequence numbers (using the tcpdump -S -s1600 -vvv command), and answer the following questions:
  - Is the server sending an RST/out of sequence packet?
  - Is the client sending an RST/out of sequence packet?
  - Are the packets making it through the BIG-IP system? (If they are, it is probably not an issue with the BIG-IP system.)

  Important: If at this point you have proved that it is not an issue with the BIG-IP system, stop the troubleshooting process.
- If possible, run a packet trace on the next hop and verify that the packets are received. This helps you determine if the next hop is involved in the error.

Looking for Patterns

Look for a pattern by determining if one or more of the following factors are always involved when the issue occurs:

- Are there application-level factors triggering this issue? Look for the following patterns when the error occurs:
  - Repeated client requests in the same format
  - Repeated server responses in the same format
  - Common sequence of requests/responses
  - Similarities in the HTTP session
  - Similarities in the SSL session
- Is there something common in the TCP behavior in each of the problem sessions? Look for the following symptoms:
  - Out of sequence packets
  - Dropped packets
  - Repeated requests
- Is this error caused by another event? Look for some similar events happening in the same session with a short period of time, such as:

  - ARP
  - ICMP messages
  - Other (anything else that seems to be reoccurring when the error happens)
- Look for differences between the failed session and working session by reviewing differences in the following areas:

General

- Are the requests or responses different sizes?
- Are some response times faster or slower?
- Are application versions (such as browser versions) different?

Application Level

- Are there any differences in SSL cipher or SSL factors?
- Are the HTTP responses different? (200, 302, 500, 1.1, 1.1)
- Are there any differences in the keep-alives?
- Is the duration of sessions different?
- Is there any difference in the port reuse?
- If the connection is SSL, are there any differences in the loopback?

BIG-IP features used

- Are different features used on the connections?
- Does a change in features resolve the problems?

Using the data to confirm the issue

To verify your understanding of the issues involved, use the information you collected to reproduce the issue. After you reproduce the issue, investigate the relevant behaviors by reviewing the various standards (RFCs, drafts, etc.) to confirm that the behavior itself is violating an established standard.

2.11b - Explain how to follow a conversation from client side and server side traces

Follow a conversation in a full proxy environment

In a full proxy environment it can be difficult to follow a conversation from Client to Server and back again. This is because of the behavior of a full proxy handling separate TCP flows on each side of the unit. You will need to capture both sides of the conversation to see everything.

You can capture both sides of the conversation by running more than one trace at a time; one on the external VLAN and one on the internal VLAN. Another way to capture the traffic on both sides at once is to choose the 0.0 interface in the capture. This is known as the loopback interface on the BIG-IP platform. Essentially it will capture all interfaces at the same time. Once you have captured the traffic you can look for the conversation on each side either in separate traces or in the one trace.

> **Note:** you should always try to filter the amount of traffic you capture when capturing on the 0.0 interface so you do not impact the system.

With basic network load balancing in the BIG-IP platform the original client IP address will be seen on both sides of the conversation so this is a good way to find a conversation of a client heading through a BIG-IP to a server resource. Once you have found the client conversation you can start to try to see if there were any issues with the conversation. However when there are setting turned on like SNAT on the virtual server the behavior of the client IP address being seen on both sides of the conversation will not be the case. In this case you will see either a BIG-IP self-IP address or an address from a SNAT pool used as the client address on the server side of the flow. You will have to match time stamps of the packets on each side of the flow to try to find the flow and with the new –p switch in tcpdump to help with peer flows it can make finding the flow on each side of the conversation easier.

2.11c - Explain how SNAT and OneConnect effect protocol analyzer traces

Tcpdump with Snat
Overview of the OneConnect Profile

Traces affected by SNAT and OneConnect

SNAT

If you perform a tcpdump on VS that has SNAT configured you are not going to see the clients IP address on the server-side of the conversation. All traffic will use the same IP (SNAT IP) or range of IP addresses if you are

using a SNAT pool. Starting in v11.2, there is an undocumented feature that can help. It's a new "-p" flag to dump on "peer" flows in tcpdump.

Using –p in a capture

1. Log on to CLI via SSH and start packet capture:
   `tcpdump -ni 0.0:nnnp -s 0 host client-ip -w /var/tmp/traffic_from_client.pcap`
   Replacing client-ip with IP of the client.

   example from bash:
   `tcpdump -ni 0.0:nnnp -s 0 host 10.100.100.102 -w /var/tmp/traffic_from_client.pcap`

   Note, with the "-p" flag, you can narrow down by all traffic to that VIP as well if you put
   `tcpdump -ni 0.0:nnnp -s 0 host <vip-ip> and port <vip-port> -w /var/tmp/traffic_to_vip.pcap`

   Note: Above capture takes advantage of new tcpdump flag "-p" that captures peer sides of the connection which is useful when traffic is snatted on the serverside. It requires a little workaround to reset/clear the filter internally (running a different capture without the -p flag that won't match original filter).

No more capturing an insane amount of traffic for that needle in a haystack on the serverside!

OneConnect

The OneConnect feature works with HTTP Keep-Alives to allow the BIG-IP system to minimize the number of server-side TCP connections by making existing connections available for reuse by other clients.

For example, when a client makes a new connection to a BIG-IP virtual server configured with an OneConnect profile, the BIG-IP system parses the HTTP request, selects a server using the load-balancing method defined in the pool, and creates a connection to that server. When the client's initial HTTP request is complete, the BIG-IP system temporarily holds the connection open and makes the idle TCP connection to the pool member available for reuse.

When a new connection is initiated to the virtual server, if an existing server-side flow to the pool member is idle, the BIG-IP system applies the OneConnect source mask to the IP address in the request to determine whether it is eligible to reuse the existing idle connection. If it is eligible, the BIG-IP system marks the connection as non-idle and sends a client request over it. If the request is not eligible for reuse, or an idle server-side flow is not found, the BIG-IP system creates a new server-side TCP connection and sends client requests over it.

> **Note:** Using a broad source mask may skew server-side statistical data. For example, when you use the mask 0.0.0.0, requests from multiple clients may appear as though they are originating from a single IP address.

Because the OneConnect profile reuses idle connections, in a packet capture of traffic on the Server-side of the conversation you may not see a TCP 3-way handshake for the new conversation and will look like a continuation of the existing conversation. This can make it difficult to trace the conversation.

2.11d - Explain how to decrypt SSL traffic for protocol analysis

See section 2.10b

This topic content is the same as topic 2.10b – simply applied to the 2.11 objective.

2.11e - Explain how time stamps are used to identify slow traffic

Time Stamps

When packets are captured, each packet is time stamped as it comes in. These time stamps will be saved into the capture file with the corresponding packet. Time stamps come from the libpcap (WinPcap) library, which in turn gets them from the operating system kernel.

Time stamps in a capture provide us with the ability to see round trip times and application response times, which are critical to the troubleshooting process.

2.11f - Explain how to recognize the different causes of slow traffic (e.g., drops, RSTs, retransmits, ICMP errors, demotion from CMP)

Overview of BIG-IP TCP RST Behavior
Overview of Clustered Multiprocessing (11.0.0 - 11.2.x)

Causes of slow traffic

There can be many different reasons for slow traffic or poor network performance and to end users of a network based application it all seems the same “The network is slow”. Any of these topics or combination of topics can cause the network to seem slow.

Resets

There are many resets but some common resets are as follows:

TCP Reset due to no response

The next reset is a TCP reset that happens when a network frame is sent six times (this would be the original frame plus five retransmits of the frame) without a response. As a result, the sending node resets the connection. This is assuming that we have an established connection after the three way handshake. The number of retransmits before the reset is configurable, but the default is five.

> **Note:** The max number of retransmits for the Syn frame when establishing the connection is 2 by default but can be controlled by the TCPMaxConnectRetransmission registry key.

There are some very important factors to remember here and it is easy for a novice to miss this or get confused and assume TCP has reset the connection when it has not. One thing to look at is the number of retransmits. In this situation, the sender will send a frame and not receive an ACK for that frame, so TCP will go into the normal retransmit behavior, each time not receiving an ACK for that frame. After the packet is retransmitted for the fifth time, the sender waits for a set amount of time for the ACK frame. When it still does not receive the ACK, it then sends a reset for the connection. The assumption is that there is a problem either with the network between the nodes or the node the ACK is being sent to, which means the connection is no longer valid. Some things to watch out for:

- This has to be the same packet, not just five retransmitted packets. It is the same packet retransmitted five times.
- It doesn't matter if other frames have been sent from the sending node and Acknowledged beyond the frame that is being retransmitted.
- A late Acknowledgement won't cause this. A late Acknowledgement occurs when the Retransmit Time Out (RTO) has expired before the Acknowledgement of a frame is sent so that the frame is retransmitted and then Acknowledgement for that frame is received.

Application Reset

This situation also generates a lot of calls and, unfortunately, is determined typically by process of elimination. In other words, there is no other reason for the reset so it must have come from the application. I hate saying that, but that really is the answer. If we look at the network traffic and see no reason for TCP itself to have sent the reset, such as the other examples above, then it must have been sent from the application. As I mentioned in the first paragraph, this is perfectly legitimate and may even be desirable. This is a common practice in an application that is making a large number of short-lived TCP connections. Such an application can cause port exhaustion on the server due to so many ports being in a Time Wait state. However, application developers need to understand why the Time Wait state exists before just resetting all connections.

> **Note:** It is possible to look at the code for an application and see if it is performing a Winsock function close (socket). If this is done on a connection where data has been set, then this will generate the Reset. You can also see this in Winsock logging. If this function is called on a TCP connection where

only the Three Way Handshake has been completed, but no data has been sent, it will result in the graceful close of the connection using Fin frames.

Dropped packets

A dropped packet on a network can cause a overhead on your network. It is likely that if a packet is dropped more than one packet will have to be resent. This is caused when network fragmentation occurs. As a network interface is placing packets on the wire it is following an MTU size limit for each packet. If the operating system is sending data to the network that is in a way that is larger than the MTU size the interface will break it up into packets that will fit the MTU. So when one packet is lost it must resend all the packets for that data not just the one lost packet.

Retransmits

A retransmit at a TCP level will likely mean that the TCP session data has to be resent and if there is also fragmentation occurring at the network layer it will compound the issue as mentioned in the dropped packet section above.

ICMP Errors

ICMP Errors can add to an already taxed network. If you are under an ICMP DOS attack the ICMP errors will add to the overhead that has to be processed.

ICMP messages are typically used for diagnostic or control purposes or generated in response to errors in IP operations (as specified in RFC 1122). ICMP errors are directed to the source IP address of the originating packet.

ICMP error messages contain a data section that includes the entire IPv4 header, plus the first eight bytes of data from the IPv4 packet that caused the error message. The ICMP packet is then encapsulated in a new IPv4 packet.

CMP Demotion

CMP should not be confused with Symmetric Multi-Processing (SMP). SMP architecture is used in multiple operating systems. SMP operates by allowing operating systems and software applications that are optimized for SMP to use the multiple processors that are available to the operating system. SMP performs this operation by spreading multiple threads across multiple processors, which allows for faster processing and more efficient use of system resources, as multiple threads can be processed simultaneously instead of waiting in a queue to be processed. CMP uses a similar approach to leverage multiple processing units by spawning a separate instance of the TMM process on each processing unit that is available to the system. While SMP may be used for any process, CMP processing is available only to the BIG-IP TMM process for the sole purpose of providing more dedicated resources to manage load balanced traffic. With multiple TMM instances simultaneously processing traffic, system performance is enhanced, and traffic management capacity is expanded.

Even if CMP is enabled on a virtual server, the BIG-IP system demotes a virtual server with incompatible features from CMP processing. This means it will run slower due to the CMP feature being turned off.

If the virtual server has been demoted, the CMP Mode line of the TMSH show ltm virtual <virtual_server_name> command reports none, disable, or single to indicate that CMP has been demoted for the virtual server.

Objective - 2.12 Given a trace, determine a solution to an application problem

2.12a - Identify application issues based on a protocol analyzer trace

See section 2.11a

This topic content is the same as topic 2.11a – simply applied to the 2.12 objective.

2.12b - Explain how to follow a conversation from client side and server side traces

See section 2.11b

This topic content is the same as topic 2.11b – simply applied to the 2.12 objective.

2.12c - Explain how SNAT and OneConnect effect protocol analyzer traces

See section 2.11c

This topic content is the same as topic 2.11c – simply applied to the 2.12 objective.

2.12d - Explain how to decrypt SSL traffic for protocol analysis

See section 2.10b

This topic content is the same as topic 2.10b – simply applied to the 2.12 objective.

2.12e - Explain how time stamps are used to identify slow traffic

See section 2.11e

This topic content is the same as topic 2.11e – simply applied to the 2.12 objective.

2.12f - Explain how to recognize the different causes of slow traffic (e.g., drops, RSTs, retransmits, ICMP errors, demotion from CMP)

See section 2.11f

This topic content is the same as topic 2.11f – simply applied to the 2.12 objective.

Objective - 2.13 Given a scenario, determine from where the protocol analyzer data should be collected

Scenario Based Objective

To prepare for scenario based questions the candidate will need to complete hands-on configuration and testing of the configuration on the LTM. This will allow the candidate to better understand how different configurations can produce different results. All F5 exams use scenario-based questions that make the candidate apply what they know to a situation to determine the resulting outcome.

This topic is focused on using a protocol analyzer to collect traffic for analysis.

2.13a - Explain how to decrypt SSL traffic for protocol analysis

See section 2.10b

This topic content is the same as topic 2.10b – simply applied to the 2.13 objective.

2.13b - Explain how to recognize the different causes of slow traffic (e.g., drops, RSTs, retransmits, ICMP errors, demotion from CMP)

See section 2.11f

This topic content is the same as topic 2.11f – simply applied to the 2.13 objective.

Objective - 2.14 Given a trace, identify monitor issues

2.14a - Describe the appropriate output for an EAV monitor

A Brief Introduction To External Application Verification Monitors

Extended Application Verification Monitor

EAVs (External Application Verification) monitors are one of most useful and extensible features of the BIG-IP product line. They give the end user the ability to utilize the underlying Linux operating system to perform complex and thorough service checks. Given a service that does not have a monitor provided, a lot of users will assign the closest related monitor and consider the solution complete. There are more than a few cases where a TCP or UDP monitor will mark a service “up” even while the service is unresponsive. EAVs give us the ability to dive much deeper than merely performing a 3-way handshake and neglecting the other layers of the application or service.

How EAVs Work

An EAV monitor is an executable script located on the BIG-IP’s file system (usually under /usr/bin/monitors) that is executed at regular intervals by the bigd daemon and reports its status. One of the most common misconceptions (especially amongst those with *nix backgrounds) is that the exit status of the script dictates the fate of the pool member. The exit status has nothing to do with how bigd interprets the pool member’s health. Any output to stdout (standard output) from the script will mark the pool member “up”. This is a nuance that should receive special attention when architecting your next EAV. Analyze each line of your script and make sure nothing will inadvertently get directed to stdout during monitor execution. The most common example is when someone writes a script that echoes “up” when the checks execute correctly and “down” when they fail. The pool member will be enabled by the BIG-IP under both circumstances rendering a useless monitor.

2.14b - Identify the different types of monitors

Monitors Settings Reference

Health Monitor Functional Categories

The following is a list of the functional types of health monitors.

- Address-check monitors
  An address-check monitor is a simple monitor that pings an IP address to verify that the address can be reached on a network.

- Service-check monitors
  A service-check monitor determines whether a service is available by opening a connection to an IP address and port.
- Content-check monitors
  A content-check monitor sends a command to a server and examines that server's response to ensure that it is serving appropriate content.
- Path-check monitors
  A path-check monitor determines whether traffic can flow through a given device to an arbitrary endpoint. The monitor sends a packet through the network device, or to a remote server, to verify that the traffic can actually pass through the network device, and not just to the device.
- Application-check monitors
  An application-check monitor is typically a custom monitor or external monitor that tests a specific application. For example, an FTP monitor connects, logs in by using a user ID and password, changes to a specified directory, and requests a specific file. This monitor succeeds when the file is received.
- Performance monitors
  A performance monitor interacts with the server (as opposed to virtual server) to examine the server load and to acquire information about the condition of virtual servers.

2.14c - Describe the characteristics of the different types of monitors

Monitors Settings Reference

Health Check Type Characteristics

The following tables describe the characteristics of the different health monitors types.

Address-check monitors

Address-check Monitor	Description
Gateway ICMP	Uses Internet Control Message Protocol (ICMP) to make a simple resource check. The check is successful if the monitor receives a response to an ICMP_ ECHO datagram.
ICMP	Makes a simple node check. The check is successful if the monitor receives a response to an ICMP_ECHO datagram.
TCP Echo	Verifies Transmission Control Protocol (TCP) connections. The check is successful if the BIG-IP system receives a response to a TCP Echo message.

Service-check monitors

Service-check Monitor	Description
Diameter	Monitors servers running the Diameter authentication service. After configuring a Diameter monitor, associate the monitor with a load balancing pool. The BIG-IP system then attempts to establish a TCP connection with a server in the pool. After successfully establishing a connection, the Diameter monitor sends a Capabilities-Exchanging-Request (CER) message to the server. The monitor then waits to receive a Capabilities-Exchanging-Answer (CEA) message, as well as a result code of DIAMETER_SUCCESS (2001).
FirePass	Checks the health of FirePass systems.
Inband	Performs passive monitoring as part of client requests. This monitor, when acting as a client, attempts to connect to a pool member. If the pool member does not respond to a connection request after a user-specified number of tries within a user-specified period, the monitor marks the pool member as down. After the monitor has marked the pool member as down, and after a user-specified period has passed, the monitor again tries to connect to the pool member (if so configured).
NNTP	Checks the status of Usenet News traffic. The check is successful if the monitor retrieves a newsgroup identification line from the server. An NNTP monitor requires a newsgroup name (for example, alt.cars.mercedes) and, if necessary, a user name and password.
MSSQL	Performs service checks on Microsoft SQL Server-based services such as Microsoft SQL Server versions 6.5 and 7.0.
MySQL	Checks the status of a MySQL database server. The check is successful if the monitor is able to connect to the server, log in as the indicated user, and log out.
Oracle	Checks the status of an Oracle database server. The check is successful if the monitor is able to connect to the server, log in as the indicated user, and log out.
POP3	Checks the status of Post Office Protocol (POP) traffic. The check is successful if the monitor is able to connect to the server, log in as the indicated user, and log out. A POP3 monitor requires a user name and password.
PostgreSQL	Checks the status of a PostgreSQL database server. The check is successful if the monitor is able to connect to the server, log in as the indicated user, and log out.
RADIUS	Checks the status of Remote Access Dial-in User Service (RADIUS) servers. The check is successful if the server authenticates the requesting user. A RADIUS monitor requires a user name, a password, and a shared secret string for the code number.
RADIUS Accounting	Checks the status of Remote Access Dial-in User Service (RADIUS) accounting servers. A RADIUS Accounting monitor requires a user name and a shared secret string for the code number.

Service-check Monitor	Description
RPC	Checks the availability of specific programs that reside on a remote procedure call (RPC) server. This monitor uses the rpcinfo command to query the RPC server and verify the availability of a given program.
SASP	Verifies the availability of an IBM Group Workload Manager. This monitor uses the Server/Application State Protocol (SASP) to communicate with the Group Workload Manager. The monitor queries the Group Workload Manager for information on the current weights of each managed resource. These weights determine which resource currently provides the best response time. When the monitor receives this information from the Group Workload Manager (GWM), it configures the dynamic ratio option for the resources, allowing the BIG-IP system to select the most appropriate resource to respond to a connection request. **Note:** When you assign an SASP monitor, the monitor initially marks the resources as down. This change in status occurs because the GWM might not yet have information pertaining to its resources. As soon as the monitor receives the results of its query, it changes the status as needed. In most configurations, the monitor receives these results within a few seconds.
SIP	Checks the status of SIP Call-ID services. By default, this monitor type issues an SIP OPTIONS request to a server device. However, you can use alternative protocols instead: TCP, UDP, TLS, and SIPS (that is, Secure SIP).
SMB	Verifies the availability of a Server Message Block/Common Internet File System (SMB/CIFS) server. Use this monitor to check the availability of the server as a whole, the availability of a specific service on the server, or the availability of a specific file used by a service.
SOAP	Tests a web service based on the Simple Object Access Protocol (SOAP). The monitor submits a request to a SOAP-based web service, and optionally, verifies a return value or fault.
TCP Half Open	Monitors the associated service by sending a TCP SYN packet to the service. As soon as the monitor receives the SYN-ACK packet, the monitor marks the service as up.
UDP	Verifies the User Datagram Protocol (UDP) service by attempting to send UDP packets to a pool, pool member, or virtual server and receiving a reply.

Content-check monitors

Content-check Monitor	Description
DNS	Checks the status of Domain Name Server (DNS) servers, by sending a specific string, and verifying receipt of that string. The check is successful if the DNS server responds with a specified string within a specified period.
HTTP	Checks the status of Hypertext Transfer Protocol (HTTP) traffic. Like a TCP monitor, an HTTP monitor attempts to receive specific content from a web page, and unlike a TCP monitor, may send a user name and password. Note: An HTTP monitor can monitor Outlook Web Access (OWA) in Microsoft Exchange Server 2007 and Microsoft SharePoint 2007 web sites that require NT LAN Manager (NTLM) authentication. NTLM authentication requires a send string that complies with HTTP/1.1, a user name, and a password.
HTTPS	Checks the status of Hypertext Transfer Protocol Secure (HTTPS) traffic. An HTTPS monitor attempts to receive specific content from a web page protected by SSL security. The check is successful when the content matches the Receive String value. **Note:** An HTTP monitor can monitor Outlook Web Access (OWA) in Microsoft Exchange Server 2007 and Microsoft SharePoint 2007 web sites that require NT LAN Manager (NTLM) authentication. NTLM authentication requires a send string that complies with HTTP/1.1, a user name, and a password.
https_443	Checks the status of Hypertext Transfer Protocol Secure (HTTPS) traffic, by using port 443.
LDAP	Checks the status of Lightweight Directory Access Protocol (LDAP) servers. A check is successful if entries are returned for the base and filter specified. An LDAP monitor requires a user name, a password, and base and filter strings.
Scripted	Generates a simple script that reads a file that you create. The file contains send and expect strings to specify lines that you want to send or that you expect to receive.
SMTP	Checks the status of Simple Mail Transport Protocol (SMTP) servers. This monitor type checks only that the server is up and responding to commands. The check is successful if the mail server responds to the standard SMTP HELO and QUIT commands.
TCP	Verifies the Transmission Control Protocol (TCP) service by attempting to receive specific content from a resource. The check is successful when the content matches the Receive String value.
WAP	Monitors Wireless Application Protocol (WAP) servers. The common usage for the WAP monitor is to specify the Send String and Receive String settings only. The WAP monitor functions by requesting a URL and finding the string in the Receive String setting in the data returned by the URL response.

Path-check monitors

Path-check Monitor	Description
Gateway ICMP	Uses Internet Control Message Protocol (ICMP) to make a simple resource check. The check is successful if the monitor receives a response to an ICMP_ ECHO datagram.
ICMP	Makes a simple node check. The check is successful if the monitor receives a response to an ICMP_ECHO datagram.
TCP Echo	Verifies Transmission Control Protocol (TCP) connections. The check is successful if the BIG-IP system receives a response to a TCP Echo message.

Application-check monitors

Application-check Monitor	Description
BIG-IP	Gathers metrics and statistics information that the Local Traffic Manager acquires through the monitoring of its own resources. Typically, it is sufficient to assign only the BIG-IP monitor to a Local Traffic Manager. When you want to verify the availability of a specific resource managed by the Local Traffic Manager, F5 Networks recommends that you first assign the appropriate monitor to the resource through the Local Traffic Manager, and then assign a BIG-IP monitor to the Local Traffic Manager through the Global Traffic Manager. This configuration provides the most efficient means of tracking resources managed by a BIG-IP system.
BIG-IP Link	Gathers metrics and statistics information that the Link Controller acquires through the monitoring of its own resources. When you use the Global Traffic Manager in a network that contains a Link Controller, you must assign a BIG-IP Link monitor to the Link Controller. This monitor is automatically assigned to the Link Controller if you do not manually assign it.
External	Enables you to create your own monitor type.
FTP	Attempts to download a specified file to the /var/tmp directory, and if the file is retrieved, the check is successful. Note that once the file has been successfully downloaded, the BIG-IP system does not save it.
IMAP	Checks the status of Internet Message Access Protocol (IMAP) traffic. An IMAP monitor is essentially a POP3 type of monitor with the addition of the Folder setting. The check is successful if the monitor is able to log into a server and open the specified mail folder.

Application-check Monitor	Description
Module Score	Enables global and local traffic management systems to load balance in a proportional manner to local traffic management virtual servers associated with the WebAccelerator system and Application Security Manager. When you configure a Module Score monitor, the local traffic management system uses SNMP to pull the gtm_score values from the downstream virtual servers and set the dynamic ratios on the associated upstream local traffic management pool members or nodes. The Module Score monitor retrieves the gtm_score values from the virtual server and the gtm_vs_score values associated with the virtual server. Then, if a pool name is not specified, this monitor sets the dynamic ratio on the node that is associated with the virtual server. The BIG-IP system uses the lowest non-zero value of the gtm_vs_score values to set the dynamic ratio. If all gtm_vs_score values are zero, then the gtm_score value is used to set the dynamic ratios. If you specify a pool name in the monitor definition, then the dynamic ratio is set on the pool member.
Virtual Location	Optimizes end-user response time in environments with dynamic distribution of application resources across multiple data centers. When using the Virtual Location monitor, the BIG-IP sets the Priority Group value of all local pool members to 2 (a higher priority). When a member of a load balancing pool migrates to a remote data center the Virtual Location monitor lowers the members Priority Group value to 1 (a lower priority). This value adjustment results in subsequent connections being sent to local pool members only if available. If no local pool members are available, connections are sent to the remote pool member.

Performance monitors

Performance Monitor	Description
BIG-IP	Collects data from Global Traffic Manager and Local Traffic Manager. Typically, the Local Traffic Manager probes local pool members and provides the results to Global Traffic Manager. Note: When the BIG-IP monitor fails, all virtual servers for that Local Traffic Manager system are marked unavailable, regardless of the results of individual virtual server probes.
BIG-IP Link	Gathers metrics and statistics information acquired through the monitoring of Global Traffic Manager or Link Controller resources.
SNMP	Checks the performance of a server that runs an SNMP agent to load balance to that server. A custom snmp gtm import setting is assigned to servers that are not developed by F5 Networks.

Performance Monitor	Description
SNMP DCA	Checks the performance of a server running an SNMP agent such as UC Davis, for the purpose of load balancing traffic to that server. With this monitor you can define ratio weights for CPU, memory, and disk use.
SNMP DCA Base	Checks the performance of servers that are running an SNMP agent, such as UC Davis. However, you should use this monitor only when you want the load balancing destination to be based solely on user data, and not CPU, memory, or disk use.
Real Server	Checks the performance of a node that is running the RealSystem Server data collection agent. The monitor then dynamically load balances traffic accordingly.
WMI	Checks the performance of a node that is running the Windows Management Infrastructure (WMI) data collection agent, and then dynamically load balances traffic accordingly. Generally, you would use a WMI monitor with dynamic ratio load balancing. Note: When using the GetWinMediaInfo command with a WMI monitor, Microsoft Windows Server 2003 and Microsoft Windows Server 2008 require the applicable version of Windows Media Services to be installed on each server.

Objective - 2.15 Given a monitor issue, determine an appropriate solution

2.15a - Describe the appropriate output for an EAV monitor

See section 2.14a

This topic content is the same as topic 2.14a – simply applied to the 2.15 objective.

2.15b - Describe the steps necessary to generate advanced monitors

Creating a Custom Monitor

Advanced Monitors

Advanced monitors can be simply adding a good GET and Receive string to an HTTP monitor, or it could mean to create a scripted monitor. Either way you will need to create a custom monitor in the configuration and define the parameters of the monitor.

Creating a custom monitor

Before creating a custom monitor, you must decide on a monitor type.

You can create a custom monitor when the values defined in a pre-configured monitor do not meet your needs, or no pre-configured monitor exists for the type of monitor you are creating.

> **Important:** When defining values for custom monitors, make sure you avoid using any values that are on the list of reserved keywords. For more information, see solution number 3653 (for version 9.0 systems and later) on the AskF5 technical support web site.

1. On the Main tab, click Local Traffic > Monitors. The Monitor List screen opens.
2. Click Create. The New Monitor screen opens.
3. Type a name for the monitor in the Name field.
4. From the Type list, select the type of monitor. The screen refreshes, and displays the configuration options for the monitor type.
5. From the Import Settings list, select an existing monitor. The new monitor inherits initial configuration values from the existing monitor.
6. From the Configuration list, select Advanced. This selection makes it possible for you to modify additional default settings.
7. Configure all settings shown.
8. Click Finished.

2.15c - Identify the different types of monitors

See section 2.14b

This topic content is the same as topic 2.14b – simply applied to the 2.15 objective.

2.15d - Describe the characteristics of the different types of monitors

See section 2.14c

This topic content is the same as topic 2.14c – simply applied to the 2.15 objective.

2.15e - Describe the input parameters passed to EAV monitors

A Brief Introduction To External Application Verification Monitors

EAV input Parameters

Bigd automatically provides two arguments to the EAV's script upon execution: node IP address and node port number. The node IP address is provided with an IPv6 prefix that may need to be removed in order for the script to function correctly. You'll notice we remove the "::ffff:///" prefix with a sed substitution in the example below. Other arguments can be provided to the script when configured in the UI (or command line). The user-provided arguments will have offsets of $3, $4, etc.

```
#!/bin/bash

# $1 = node IP
# $2 = node port
# $3 = hostname to resolve

[[ $# != 3 ]] && logger -p local0.error -t ${0##*/} -- "usage: ${0##*/} <node
IP> <node port> <hostname to resolve>" && exit 1

node_ip=$(echo $1 | sed 's/::fff://')

dig +short @$node_ip $3 IN A &> /dev/null

[[ $? == 0 ]] && echo "UP"
```

2.14f - Describe the construct of an EAV monitor

A Brief Introduction To External Application Verification Monitors

UNIX Shell Programming QuickStart

EAV Construct

An EAV script is a bash script.

The shbang line	The "shbang" line is the very first line of the script and lets the kernel know what shell will be interpreting the lines in the script. The shbang line consists of a #! followed by the full pathname to the shell, and can be followed by options to control the behavior of the shell. EXAMPLE `#!/bin/bash`
Comments	Comments are descriptive material preceded by a # sign. They are in effect until the end of a line and can be started anywhere on the line. EXAMPLE `# This is a comment`
Arguments	Functions to check to see if the application is working
Output	To print output to the screen, the echo command is used. Wildcards must be escaped with either a backslash or matching quotes. EXAMPLE `echo “How are you?”`

In the example below we are using the dig (Domain Information Groper) command to query our DNS server for an A record. We use the exit status from dig to determine if the monitor will pass. Notice how the script will never output anything to stdout other than "UP" in the case of success. If there aren't enough arguments for the script to proceed, we output the usage to /var/log/ltm and exit. This is a very simple 13 line script, but effective example.

```
#!/bin/bash

# $1 = node IP
# $2 = node port
# $3 = hostname to resolve

[[ $# != 3 ]] && logger -p local0.error -t ${0##*/} -- "usage: ${0##*/} <node
IP> <node port> <hostname to resolve>" && exit 1

node_ip=$(echo $1 | sed 's/::fff://')

dig +short @$node_ip $3 IN A &> /dev/null

[[ $? == 0 ]] && echo "UP"
```

SECTION 3 – TROUBLESHOOT BASIC PERFORMANCE ISSUES

Objective - 3.01 Interpret log file messages and or command line output to identify LTM device issues

3.01a - Identify LTM device logs

Event Logging

LTM Logs

Viewing and managing log messages are an important part of maintaining a BIG-IP system. Log messages inform you on a regular basis of the events that are happening on the system. Some of these events pertain to general events happening within the operating system, while other events are specific to the BIG-IP system, such as the stopping and starting of BIG-IP system services.

The mechanism that the BIG-IP system uses to log events is the Linux utility syslog-ng. The syslog-ng utility is an enhanced version of the standard UNIX and Linux logging utility syslog.

The types of events that the BIG-IP system logs are:

- System events - System event messages are based on Linux events, and are not specific to the BIG-IP system.
- Packet filter events - Packet filter messages are those that result from the implementation of packet filters and packet-filter rules.
- Local traffic events - Local-traffic event messages pertain specifically to the local traffic management system.
- Audit events - Audit event messages are those that the BIG-IP system logs as a result of changes to the BIG-IP system configuration. Logging audit events is optional.

To configure and manage event logging, log in to the BIG-IP Configuration utility, and on the Main tab, expand System, and click Logs.

The BIG-IP system automatically logs these four main event types: system, packet filter, local traffic, and configuration changes (audit). Each type of event is stored in a separate log file, and the information stored in each log file varies depending on the event type. All log files for these event types are in the directory /var/log.

Logging system events

Many events that occur on the BIG-IP system are Linux-related events, and do not specifically apply to the BIG-IP system.

Using the Configuration utility, you can display these system messages.

Logging packet filter events

Some of the events that the BIG-IP system logs are related to packet filtering. The system logs the messages for these events in the file /var/log/pktfilter.

Using the Configuration utility, you can display these packet filter messages.

Logging local traffic events

Many of the events that the BIG-IP system logs are related to local area traffic passing through the BIG-IP system. The BIG-IP system logs the messages for these events in the file /var/log/ltm.

Using the Configuration utility, you can display these local-traffic messages.

Some of the specific types of events that the BIG-IP system displays on the Local Traffic logging screen are:

- Address Resolution Protocol (ARP) packet and ARP cache events
- bigdb database events (such as populating and persisting bigdb variables)
- HTTP protocol events
- HTTP compression events
- IP packet discard events due to exceptional circumstances or invalid parameters (such as a bad checksum)
- Layer 4 events (events related to TCP, UDP, and Fast L4 processing)
- MCP/TMM configuration events
- Monitor configuration events
- Network events (Layers 1 and 2)
- Packet Velocity ASIC (PVA) configuration events
- iRule events related to run-time iRule processing
- SSL traffic processing events
- General TMM events such as TMM startup and shutdown

3.01b - Identify the LTM device statistics in the GUI and command line

Viewing Memory Utilization for Multiple TMM Instances on CMP Systems

LTM device statistics

Clustered Multi-Processing (CMP) is a feature that allows BIG-IP platforms with multiple processors to use multiple TMM instances to increase traffic handling.

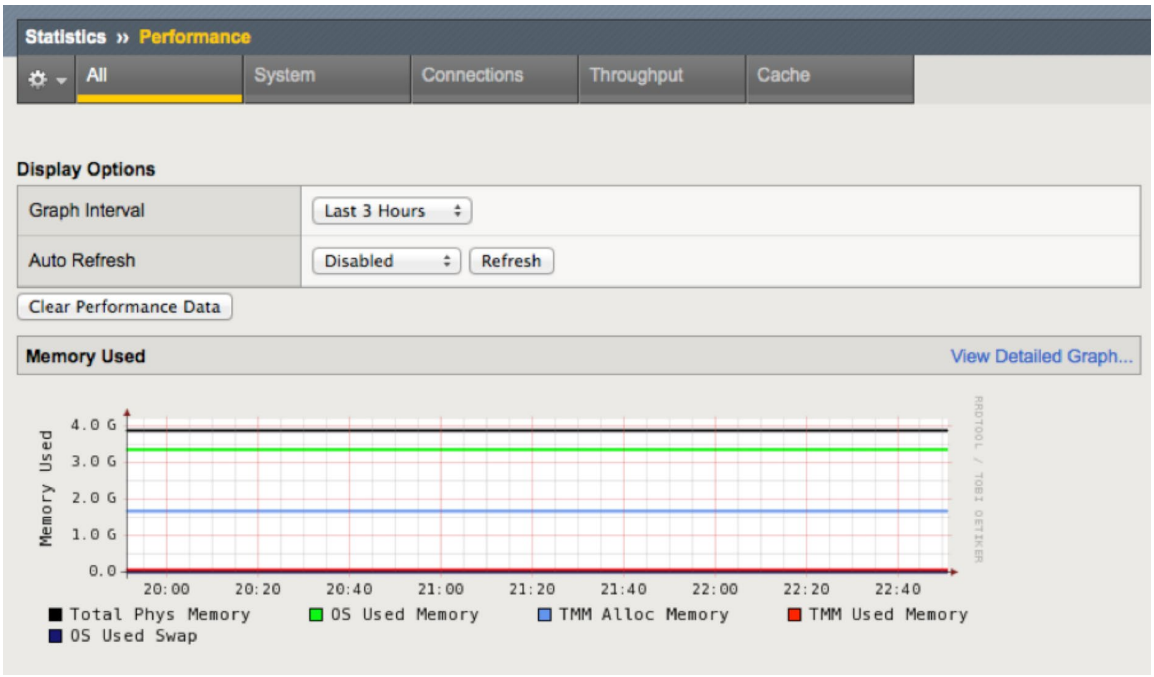
You can view memory utilization for the multiple TMM instances running on a CMP system using one of the following methods:

- Viewing memory utilization for multiple TMM instances using the Configuration utility
- Viewing memory utilization for multiple TMM instances using SNMP
- Viewing memory utilization for multiple TMM instances using the TMSH utility

Viewing memory utilization for multiple TMM instances using the Configuration utility

1. Log in to the Configuration utility.
2. Click Statistics. (In versions prior to 11.1.0, click Overview.)
3. Click Performance.
4. Click the System menu or the All menu.

The Memory Used graph displays the following:



Viewing memory utilization for each TMM instance using SNMP

1. Log in to the command line.
2. Use an SNMP application, such as snmpwalk, to poll the F5-BIGIP-SYSTEM-MIB::sysTmmStatMemoryUsed OID:

```
snmpwalk -c public localhost .1.3.6.1.4.1.3375.2.1.8.2.3.1.32
```

For example, the following snmpwalk command output was taken from a BIG-IP 8800 platform running four TMM instances:

```
F5-BIGIP-SYSTEM-MIB::sysTmmStatMemoryUsed."1.0" = Counter64: 24558472
F5-BIGIP-SYSTEM-MIB::sysTmmStatMemoryUsed."1.1" = Counter64: 25595500
F5-BIGIP-SYSTEM-MIB::sysTmmStatMemoryUsed."1.2" = Counter64: 25592356
F5-BIGIP-SYSTEM-MIB::sysTmmStatMemoryUsed."1.3" = Counter64: 25601120
```

Note: Using an SNMP application to poll the F5-BIGIP-SYSTEM-MIB::sysStatMemoryUsedOID displays the aggregate memory utilization for all TMM instances.

Viewing memory utilization for multiple TMM instances using the TMSH utility

1. Log in to the command line.
2. To display memory utilization and other statistical information for TMM instances, type the following TMSH command:

   tmsh show /sys tmm-info

```
Sys::TMM: 0.0
---------------------------
Global
 TMM Process Id    28044
Running TMM Id      0
TMM Count         1
CPU Id           0
Memory (bytes)
Available       1.6G
Used          26.8M
CPU Usage Ratio (%)
Last 5 Seconds      3
Last 1 Minute      3
Last 5 Minutes      2

Sys::TMM: 0.1
---------------------------
Global
 TMM Process Id    28056
Running TMM Id      1
TMM Count         1
CPU Id           1
```

```
Memory (bytes)
Available      1.6G
Used         26.8M
CPU Usage Ratio (%)
Last 5 Seconds      3
Last 1 Minute      3
Last 5 Minutes      2
```

3.01c - Describe the functionality and uses of the AOM

Always On Management

Always-On Management (AOM) is a separate subsystem that provides lights-out management for the BIG-IP system by using the 10/100/1000 Ethernet management port over secure shell (SSH), or by using the serial console.

AOM allows you to manage BIG-IP platforms using SSH (most platforms) or the serial console, even if the Host subsystem is turned off. The BIG-IP Host subsystem and the AOM subsystem operate independently. If AOM is reset or fails, the BIG-IP Host subsystem continues to operate and there is no interruption to load-balanced traffic. AOM is always turned on when power is supplied to the platform. If the BIG-IP Host subsystem stops responding, you can use the AOM Command Menu to reset the Big-IP.

AOM features:

- Responsible for status monitoring and maintaining system health
- Provides the ability to turn on, turn off, and restart the BIG-IP Host subsystem
- Supports a hardware watchdog with a minimum five second time-out before a hardware restart
- Provides a command menu, but no shell access
- Supports IPv4 and IPv6 access
- Uses password-based authentication for SSH (root password, by default)
- Provides power supply monitoring
- Menu options allow users to view MAC address, serial number, and some memory configuration
- Supports baud configuration

Objective - 3.02 Identify the appropriate command to use to determine the cause of an LTM device problem

3.02a - Identify LTM device bash commands and TMSH commands needed to identify issues

Traffic Management Shell (tmsh) Reference Guide

Traffic Management Shell (TMSH)

You can configure TMSH terminal access for a user account so that the user can go directly to the TMSH utility when logging in to the BIG-IP terminal.

You can configure bash terminal access for a user account, and the user can access the TMSH utility by typing the following command from the command line:

```
tmsh
```

Additionally, users with advanced shell access can run a single TMSH command from the command line by entering specific parameters after TMSH.

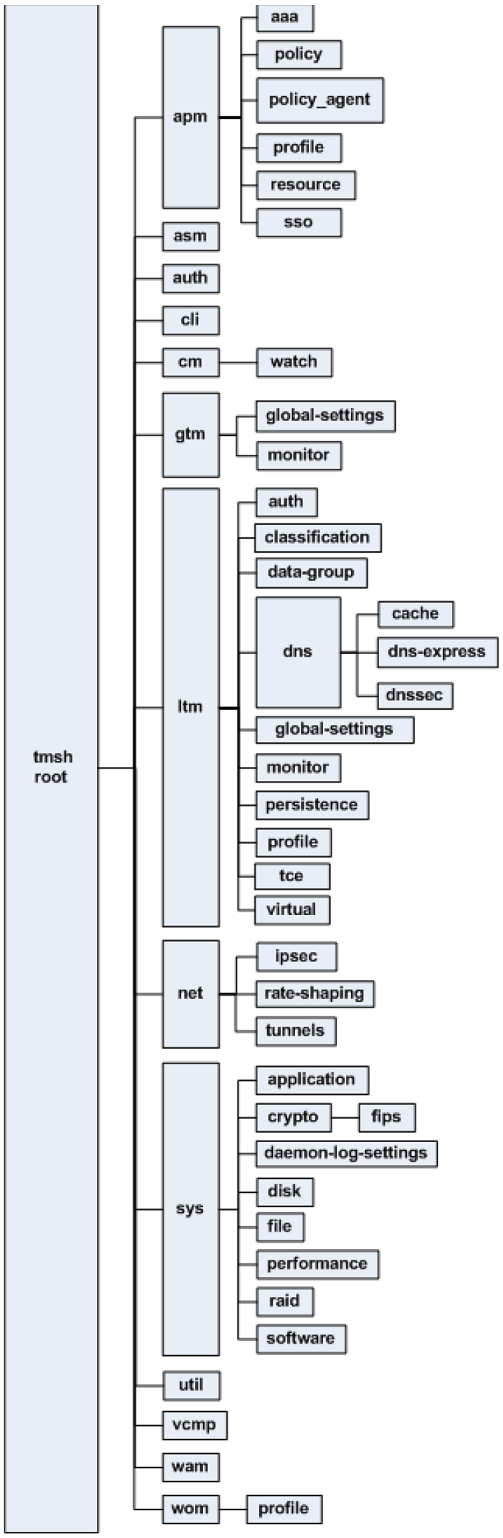
For example:

```
root@(bigipc1)(cfg-sync Disconnected)(Active)(/Common)(tmos)# tmsh show /ltm pool all
```

If you disable terminal access for a user account, the user will not have access to the TMSH utility.

TMSH is an interactive shell that you use to manage the BIG-IP system. The structure of TMSH is hierarchical and modular. The highest level is the root module, which contains twelve subordinate modules.

Several modules also contain subordinate modules. All modules and subordinate modules contain components that you configure to manage the BIG-IP system. You can configure a component that resides anywhere in the hierarchy from anywhere else in the hierarchy by using the full path to that component. Alternatively, you can configure a component by navigating to that component directly. After you create a component, you can modify that component in object mode, which is the lowest level of the hierarchy.


tmsh
root
apm
aaa
policy
policy_agent
profile
resource
sso
asm
auth
cli
cm
watch
gtm
global-settings
monitor
ltm
auth
classification
data-group
dns
cache
dns-express
dnssec
global-settings
monitor
persistence
profile
tce
virtual
net
ipsec
rate-shaping
tunnels
sys
application
crypto
fips
daemon-log-settings
disk
file
performance
raid
software
util
vcmp
wam
wom
profile

3.02b - Explain how a virtual server processes a request (most specific to least specific)

Order of Precedence for Virtual Server Matching (9.x - 11.2.1)

Virtual Server Processing Order

In version 4.x, which was just prior to version 9.x (when TMOS was created), the BIG-IP system used a virtual server precedence to define the order in which it routes a packet to a specific virtual server in the event that the packet matches multiple virtual server definitions.

The order of virtual server precedence was (from the highest precedence to the lowest precedence) as follows:

- ip:port
- ip:any
- network:port
- any:port
- network:any
- vlan:port
- vlan:any
- any:any

Many things have changed since then.

In Version 9.x through 11.2.1, (which covers this version of the exam) the BIG-IP system determines the order of precedence applied to new inbound connections using an algorithm that places a higher precedence on the address netmask and a lesser emphasis on the port. BIG-IP LTM sets virtual server precedence according to the following criteria:

- The first precedent of the algorithm chooses the virtual server that has the longest subnet match for the incoming connection.
- If the number of bits in the subnet mask match, the algorithm chooses the virtual server that has a port match.
- If no port match is found, the algorithm uses the wildcard server (if a wildcard virtual server is defined).
- A wildcard address has a netmask length of zero; thus, it has a lower precedence than any matching virtual server with a defined address.

This algorithm results in the following order of precedence:

- <address>:<port>
- <address>:*
- <network>:<port>
- <network>:*
- *:<port>
- *:*

Example of VIP precedence behavior

For example, for a BIG-IP system with the following VIPs configured on the inbound VLAN:

```
10.0.0.0/8:80

10.10.0.0/16:80

10.10.10.10/32:80

20.0.0.0/8:*

20.0.0.0/8:80

*:80 (alternatively noted as 0.0.0.0/0:80)

*:* (alternatively noted as any:any, 0.0.0.0/0:any)
```

The following table illustrates how inbound destination addresses map to the configured VIPs:

Inbound destination address	VIP
10.10.10.10:80	10.10.10.10/32:80 - address match and port match
10.10.10.11:80	10.10.0.0/16:80 - most specific address match and port match
10.1.10.10:80	10.0.0.0/8:80 - most specific address match and port match
20.0.0.0:80	20.0.0.0/8:80 - most specific address match and port match
20.0.0.0:443	20.0.0.0/8:* - most specific address match with wildcard port
1.1.1.1:443	*:* - wildcard address and wildcard port

Further changes in the order of precedence applied to new inbound connections are in Version 11.3 and later. If you care to read on this it can be found at the following location.

SOL14800: Order of Precedence for Virtual Server Matching (11.3.0 and later)

Objective - 3.03 Given a scenario, determine the cause of an LTM device failover

Scenario Based Objective

To prepare for scenario based questions the candidate will need to complete hands-on configuration and testing of the configuration on the LTM. This will allow the candidate to better understand how different configurations can produce different results. All F5 exams use scenario-based questions that make the candidate apply what they know to a situation to determine the resulting outcome.

This topic is focused on working with high availability environments and causes for failover.

3.03a - Interpret log messages to determine the cause of high availability issues

Error Message: Clock Advanced by <number> Ticks

HA Log entries

You should be familiar with log messages that are related to HA issues. The information in this section describes one of the more common ones you may see.

The BIG-IP system daemon heartbeat is a recurring signal that a service generates. The BIG-IP system continually monitors daemon heartbeat signals for certain daemons, such as the Traffic Management Microkernel (TMM), to determine whether the service is running.

The BIG-IP system logs an error message similar to the following example, to the /var/log/ltm file, when TMM fails to complete its polling loop and update its heartbeat signal within the threshold period:

01010029:5: Clock advanced by <number> ticks

Starting in BIG-IP 10.x, the threshold period is 100 milliseconds. In versions prior to BIG-IP 10.x, the threshold period is 500 milliseconds.

If the error message is logged continually, the system is likely experiencing resource issues that may be caused by one or more of the following factors:

Memory paging, I/O operations, or other events causing TMM to lose CPU time

Complex iRule configurations

Large complex configurations, such as those containing numerous Secure Socket Layer (SSL) profiles with SSL certificates or keys

Certain administrative operations, such as dumping large numbers of connections or persistence entries

Listing large ARP tables using commands such as TMSH show net arp

For example, the following log sample from a BIG-IP system indicates that TMM failed to update its heartbeat signal, causing a failover action:

```
01010029:5: Clock advanced by 518 ticks

01010029:5: Clock advanced by 505 ticks

01010029:5: Clock advanced by 590 ticks

01010029:5: Clock advanced by 568 ticks

01010029:5: Clock advanced by 1681 ticks

01010029:5: Clock advanced by 6584 ticks

01140029:5: HA daemon_heartbeat tmm fails action is failover and restart.

010c0026:5: Failover condition, active attempting to go standby.
```

These error messages indicate that, for a certain amount of time, TMM was descheduled from the CPU. During this time, all traffic intended to flow through any of the switch interfaces is most likely affected. Although this condition is impossible to prevent completely, changes were introduced in BIG-IP 10.0.0 that may reduce the likelihood of this condition occurring. For more information, refer to SOL10472: Intense disk drive I/O activity may cause essential BIG-IP processes to be swapped out or descheduled.

> **Important:** When this condition occurs on one of the members of a BIG-IP high-availability pair that uses network failover, network failover traffic will also be disrupted. If TMM is descheduled for an amount of time greater than the network failover timeout, but less than the TMM heartbeat timeout, both systems will become active, and stay active, until network failover traffic is restored or the TMM heartbeat expires.

3.03b - Explain the effect of network failover settings on the LTM device

Limiting Network Failover Disruptions

Network Failover

Failover is a process that occurs when an active system in the device group becomes unavailable, causing a peer unit to begin processing traffic originally targeted for the unavailable unit. The network failover feature uses the network to determine the availability of the peer devices. If network failover is not properly configured for your environment, failover from one device to another may be delayed and not occur seamlessly as expected.

Issues that may cause network failover problems

The issues listed, following, may prevent the BIG-IP system from transmitting or receiving network failover heartbeat packets, and result in more than one BIG-IP device attempting to service the same local traffic objects.

Busy or blocked TMM daemon

Certain conditions may temporarily prevent TMM from processing new network traffic. For example, excessive memory paging, I/O operations, or other events that cause TMM to lose CPU time may disrupt network failover traffic. While this may occur on any BIG-IP device, it is more common on systems that do not have dedicated CPU or I/O resources, such as BIG-IP VE and vCMP guest instances.

> **Note:** When TMM is busy or blocked, the daemon may fail to update its heartbeat signal within the threshold period. When this occurs, the system logs an error message to the /var/log/ltm file that appears similar to the following example:
>
> ```
> 01010029:5: Clock advanced by <number> ticks
> ```

To limit network failover disruptions related to the previous scenario, implement one or more of the following options:

- Configure redundant network failover channels (use the management network, in addition to a TMM network)
- Configure hardwired failover (appliance platforms)
- Add additional failover unicast addresses for device group members
- Increase the network failover timeout

VIPRION blade failure

When the primary blade fails on the active VIPRION unit in a redundant pair, network failover traffic can be disrupted and result in an active-active state in which both VIPRION systems attempt to service the same local traffic management objects. This may occur when the network failover timeout (the default setting is three seconds) is less than the VIPRION blade heartbeat timeout of 10 seconds. The scenario is described, following:

1. The VIPRION systems are configured in a redundant pair (VIPRION1 is active, and VIPRION2 is standby).
2. The primary blade fails on the active unit (VIPRION1).
3. After the network failover timeout of 3 seconds, the standby system (VIPRION2) becomes active.
4. The newly active system, VIPRION2, attempts to establish a network failover connection to VIPRION1, but VIPRION1 has not yet promoted the secondary blade and does not accept the network failover connection.
5. After 10 seconds, VIPRION1 promotes the secondary blade to primary and the systems resume the network failover connection.

To limit network failover disruptions related to the previous scenario, implement the following options:

- Configure network failover for VIPRION systems
- Increase the network failover timeout
- Slow network establishment

When a BIG-IP system initializes and prepares to go online, the system establishes VLAN connectivity and attempts to make a network failover connection to the peer device. In some network environments, it is possible for network failover to timeout before the system's Ethernet links, LACP heartbeats, and spanning tree convergence for VLAN interfaces have reached a steady state. This condition is more likely to occur on BIG-IP systems in which network failover redundancy is not configured (for example, network failover is only configured on a TMM network (self IP address), but not the mgmt network).

To limit failover disruptions related to the previous scenario, implement one or more of the following options:

- Configure redundant network failover channels (use the management network, in addition to a TMM network)
- Configure hardwired failover (appliance platforms)
- Configure gateway failsafe

Failover events causing VLAN disruption

When a BIG-IP system experiences a failover event or is manually forced to standby, the previously active unit can immediately signal its condition using network failover heartbeat packets. Doing so allows the standby unit to become active without waiting for the network failover timeout to expire. However, when network failover redundancy is not configured, certain events that result in failover and also a loss of VLAN connectivity (for example, a TMM core dump) may prevent the previously-active unit from immediately signaling its condition using network failover heartbeat packets; this scenario can cause unnecessary failover delay.

To limit failover disruptions related to the previous scenario, implement the following option:

- Configure redundant network failover channels (use the management network, in addition to a TMM network)

Recommendations

When configuring network failover, you should consider the following:

Configure network failover for VIPRION systems

SOL13915: Configuring Network Failover for Redundant VIPRION Systems (11.x).

Configure redundant network failover channels (use the management network, in addition to a TMM network)

F5 recommends that you configure network failover to communicate over the management network, in addition to a TMM network. Configuring network failover in this way creates a reliable failover communication channel while the system initializes, and before the TMM network connectivity channels are fully established. To do this you must configure a failover unicast address and a failover multicast address for each device group member. For more information, refer to the following resources:

BIG-IP Redundant Systems Configuration Guide (BIG-IP versions 11.0 through 11.21)

Configure hardwired failover (appliance platforms)

BIG-IP appliance platforms are equipped with a failover port to provide hardwired failover to a redundant peer device. This option is available only for sync-failover device groups with only two devices and one floating traffic group on BIG-IP appliance hardware without vCMP guests.

For more information, refer to the following resources:

Platform and Hardware guides for your BIG-IP hardware model

Add additional failover unicast addresses for device group members

Adding additional unicast failover addresses to a device group increases network failover redundancy and allows additional TMM instances to process network failover heartbeat connections to the peer system. For example, if each BIG-IP system in the device group is running two or more TMM instances, you can ensure that each TMM instance processes a network failover connection by creating additional self IP addresses and configuring the addresses as unicast failover addresses.

For more information, refer to the following resources:

BIG-IP Redundant Systems Configuration Guide (BIG-IP versions 11.0 through 11.21)

Increase the network failover timeout

For many BIG-IP deployments, the default network failover timeout of three seconds is sufficient. However, for some deployments it may be beneficial to increase the network failover timeout to a value greater than the TMM heartbeat timeout, the VIPRION blade timeout (for VIPRION platforms), and the gateway failsafe timeout (if using the Gateway Failsafe feature). For example, increasing the network failover timeout to 12 seconds will guarantee that the time in which the system initiates network failover is greater than the TMM heartbeat timeout of 10 seconds, the VIPRION blade heartbeat timeout of 10 seconds (for VIPRION platforms), and the recommended gateway failsafe timeout of 11 seconds. Configuring the network failover timeout in this way may prevent temporary active-active scenarios in which multiple BIG-IP devices attempt to service the same local traffic objects.

> **Note:** Changing the network failover timeout to a value that is too high may delay actual failover events.

For more information, refer to the following resources:

SOL7249: Configuring the Network Failover Timer.

Configure gateway failsafe

Gateway failsafe monitors traffic between a BIG-IP system in a device group and a pool containing a gateway router. You configure the gateway failsafe feature if you want the BIG-IP system to take an action, such as fail over, when some number of gateway routers in a pool of routers becomes unreachable. The Gateway Failsafe feature can detect when the network is not available for transmitting or receiving network failover heartbeat traffic.

For more information, refer to the following resources:

- SOL15367: Configuring Gateway Fail-safe Using the TMSH Utility (11.x)

  Note: For purposes of this article, the gateway pools should be monitored with a custom Gateway ICMP monitor. The recommended settings are as follows: Interval of 1 second, a Timeout of 11

seconds, and minimum Up Interval of 3 seconds. If a monitor timeout value of less than the TMM Heartbeat (10 seconds) is used, gateway failsafe may cause a premature failover when a TMM instance is temporarily blocked from processing new network traffic but may still recover.

3.03c - Describe configuration settings (e.g., port lockdown, network timeouts, packet filters, remote switch settings)

Overview of Port Lockdown Behavior (10.x - 11.x)

Port Lockdown

The port lockdown feature allows you to secure the BIG-IP system from unwanted connection attempts by controlling the level of access to each self IP address defined on the system. Each port lockdown list setting, defined later in this document, specifies the protocols and services from which a self IP can accept connections. The system refuses traffic and connections made to a service or protocol port that is not on the list.

Port lockdown exceptions

- TCP port 1028: In BIG-IP 11.0.0 - 11.3.0 redundant pair configurations, the system allows tcp:1028 for connection and persistence mirroring, regardless of the port lockdown settings.
- TCP port 4353: When BIG-IP 11.0.0 and later devices are configured in a synchronization group, peer devices communicate using Centralized Management Infrastructure (CMI) on tcp:4353, regardless of the port lockdown settings.

  Note: CMI uses the same port as iQuery tcp:4353, but is independent of iQuery and the port configuration options available for the port.
- ICMP: ICMP traffic to the self-IP address is not affected by the port lockdown list and is implicitly allowed in all cases.

  Note: In most cases, it is not possible to ping self IP addresses across Virtual Local Area Networks (VLANs). For more information, refer to SOL3475: The BIG-IP system may not respond to ICMP ping requests for a self IP address.

Port lockdown setting definitions:

Allow Default

This option allows access for a pre-defined set of network protocols and services that are typically required in a BIG-IP deployment.

The Allow Default setting specifies that connections to the self IP address are allowed from the following protocols and services:

Allowed protocol	Service	Service definition
OSPF	N/A	N/A
TCP	4353	iQuery
UDP	4353	iQuery
TCP	443	HTTPS
TCP	161	SNMP
UDP	161	SNMP
TCP	22	SSH
TCP	53	DNS
UDP	53	DNS
UDP	520	RIP
UDP	1026	network failover

You can also determine the default supported protocols and services by using the following command:

```
tmsh list net self-allow
```

The output will appear similar to the following example:

```
net self-allow {
  defaults {
    ospf:any
    tcp:domain
    tcp:f5-iquery
    tcp:https
    tcp:snmp
    tcp:ssh
    udp:520
    udp:cap
    udp:domain
    udp:f5-iquery
    udp:snmp
  }
}
```

Allow All

This option specifies that all connections to the self IP address are allowed, regardless of protocol or service.

Allow None

This option specifies that no connections are allowed on the self IP address, regardless of protocol or service. However, ICMP traffic is always allowed, and if the BIG-IP systems are configured in a redundant pair, ports that are listed as exceptions are always allowed from the peer system.

Allow Custom

This option allows you to specify the protocols and services for which connections are allowed on the self IP address. However, ICMP traffic is always allowed, and if the BIG-IP systems are configured in a redundant pair, ports that are listed as exceptions are always allowed from the peer system.

> **Important:** A known issue prevents connections to the state mirroring address when port tcp:1028 is explicitly allowed in the custom port lockdown list. For more information, refer to SOL12932: The BIG-IP system resets statemirror connections when port 1028 is configured in the Self IP Port Lockdown list.

Default port lockdown setting

When creating self IP addresses using the Configuration utility, the default port lockdown setting in BIG-IP 10.x through 11.5.2 is Allow Default, and for BIG-IP 11.6.0 and later versions is Allow None.

When creating self IP addresses using the bigpipe or TMSH utilities, the default port lockdown setting in BIG-IP 10.x is Allow None. For BIG-IP 11.0.0-11.5.2, the default port lockdown setting is Allow Default, and for BIG-IP 11.6.0 and later versions, the default port lockdown setting is Allow None.

Using the Configuration utility to modify port lockdown settings for a specific self IP

1. Log in to the Configuration utility.
2. Navigate to Network > Self IPs.
3. Click the relevant self IP address.
4. From the Port Lockdown box, select the desired setting.
5. Click Update.

Using the TMSH utility to modify port lockdown settings

1. Log in to the Traffic Management Shell (TMSH) by typing the following command:
   `tmsh`

2. To modify the port lockdown settings for a self IP address, use the following command syntax:
   ```
   modify /net self <self_ip> allow-service <option>
   ```
   For example, to change the port lockdown setting for self IP address 10.10.10.1 to default, you would type the following command:
   ```
   modify /net self 10.10.10.1 allow-service default
   ```
3. Save the change by typing the following command:
   - BIG-IP 10.1.0 and later:
     ```
     save sys config
     ```

Recommendations

For optimal security, F5 recommends that you use the port lockdown feature to allow only the protocols or services required for a self IP address.

Overview of Port Lockdown Behavior (10.x - 11.x)

Network Timeouts

The BIG-IP network failover feature allows devices to use the network to communicate their failover status. The standby BIG-IP system uses an internal timer to track how much time has passed since the standby received the last failover status update from the active unit. The system uses the failover.nettimeoutsec database key to specify the number of seconds that the standby unit will wait after the last update from the active unit before the standby attempts to become active. The default setting is three seconds. Since the network connection between the units may traverse multiple network connections and devices, network conditions may occasionally prevent the units from communicating their status to each other for a brief period of time (for example, a small number of packets dropped or a brief network interruption). This setting helps prevent premature failover in case of a temporary traffic disruption.

> **Note:** For most environments, F5 recommends that you use the default failover.nettimeoutsec value of three seconds.

Procedures

Changing the failover.nettimeoutsec database value

You can change the failover.nettimeoutsec database value from the default of three seconds to a new value. To do so, perform the following procedure:

> **Impact of procedure:** Changing the failover.nettimeoutsec to a value that is less than the default of three seconds may unnecessarily increase failover events. Conversely, changing the variable to a value that is too high may delay actual failover events.

1. Log in to the Traffic Management Shell (tmsh) by typing the following command:
   `tmsh`
2. Modify the db key by using the following command syntax, where <new_value> is the desired value:
   `modify /sys db failover.nettimeoutsec value <new_value>`
3. Save the change by typing the following command:
   `save /sys config`

BIG-IP® TMOS®: Concepts

Packet Filters

Packet filters enhance network security by specifying whether a BIG-IP system interface should accept or reject certain packets based on criteria that you specify. Packet filters enforce an access policy on incoming traffic. They apply to incoming traffic only.

You implement packet filtering by creating packet filter rules, using the BIG-IP Configuration utility. The primary purpose of a packet filter rule is to define the criteria that you want the BIG-IP system to use when filtering packets. Examples of criteria that you can specify in a packet filter rule are:

- The source IP address of a packet
- The destination IP address of a packet
- The destination port of a packet

You specify the criteria for applying packet filter rules within an expression. When creating a packet filter rule, you can instruct the BIG-IP system to build an expression for you, in which case you need only choose the criteria from predefined lists, or you can write your own expression text, using the syntax of the tcpdump utility. For more information on the tcpdump utility, see the online man page for the tcpdump command.

Packet filter rules are unrelated to iRules.

You can also configure global packet filtering that applies to all packet filter rules that you create. The following sections describe how to use the Configuration utility to set global packet filtering options, as well as create and manage individual packet filters rules.

Overview of Port Lockdown Behavior (10.x - 11.x)

Remote Switch Settings

A failure of an upstream or downstream switch can cause an HA pair to failover. When you configure VLAN failsafe for a VLAN, the BIG-IP system monitors network traffic on the VLAN. If the BIG-IP system detects a

loss of network traffic on the VLAN, the BIG-IP system attempts to generate VLAN failsafe traffic to nodes or the default router that is accessible through the VLAN in the following manner:

VLAN failsafe timeout value expires by	Actions taken by the BIG-IP system
Half (1/2)	• Initiates an Address Resolution Protocol (ARP) request for the IP address of the oldest entry in the BIG-IP ARP table • Initiates an ICMPv6 neighbor discovery probe (only if entries exist in the BIG-IP IPv6 neighbor cache)
Three quarters (3/4)	• Initiates an ARP request for the all IP address in the BIG-IP ARP table • Initiates an ICMPv6 neighbor discovery probe (only if entries exist in the BIG-IP IPv6 neighbor cache) • Initiates a multicast ping to 224.0.0.1

The failover action is avoided if the BIG-IP system receives a response to the VLAN failsafe traffic it generated.

Recommendations

When configuring VLAN failsafe for a VLAN, you should consider the following factors:

- VLAN failsafe configuration is local to the BIG-IP system and is not a shared configuration that is synchronized between high-availability systems during ConfigSync operations. As a result, you must define VLAN failsafe on all BIG-IP units in a high-availability system.
- To avoid unnecessary failover, the VLAN failsafe timeout value should be set to a value larger than the number of seconds that the neighboring links take to initialize. An unnecessary failover may cause more disruption than a brief flap in network connectivity. Setting the timeout too low can cause system and network instability issues if both members of the redundant system experience intermittent connectivity.
- If you enable VLAN failsafe on a VLAN with nodes that do not respond consistently to the standard VLAN failsafe probes, the BIG-IP high-availability systems can experience unintended VLAN failsafe events.
- Unwanted VLAN failsafe events can occur if VLAN failsafe is enabled on a VLAN with no default gateway or pool members, and the VLAN contains only devices that do not respond to ARP requests, ICMPv6 neighbor discovery probes, or multicast pings. To help prevent this behavior, you can assign a health monitor to at least one node on that VLAN. This practice helps to consistently populate the ARP tables on the BIG-IP high-availability systems, and give a more accurate view of VLAN availability.

- If you set the VLAN failsafe action to Restart All or Reboot when a low failsafe timeout value is configured, the BIG-IP system may enter a cycle of restarting services or rebooting until VLAN failsafe is disabled. This behavior may occur when the timeout value is set too low and the interfaces are not available after a reboot or restart due to a Spanning Tree update or because LACP enabled trunks have not initialized. To prevent this behavior from occurring, do not set the VLAN failsafe timeout value below the recommended value of 90 seconds.
- Testing the VLAN failsafe feature for an HA redundant pair by removing all nodes from the VLAN may not result in a VLAN failsafe action being triggered. If the VLAN being tested is used for network failover and/or state mirroring, the traffic generated between the redundant BIG-IP systems is sufficient to prevent VLAN failsafe from being triggered.

3.03d - Explain the relationship between serial and network failover

Network failover

Network failover is based on heartbeat detection where the system sends heartbeat packets over the internal network.

The system uses the primary and secondary failover addresses to send network failover heartbeat packets. For more information about the BIG-IP mirroring and network failover transport protocols, refer to the following articles:

- SOL9057: Service port and protocol used for BIG-IP network failover
- SOL7225: Transport protocol used for BIG-IP connection and persistence mirroring

The BIG-IP system considers the peer down after the Failover.NetTimeoutSec timeout value is exceeded. The default value of Failover.NetTimeoutSec is three seconds, after which the standby unit attempts to switch to an active state. The following database entry represents the default settings for the failover time configuration:

```
Failover.NetTimeoutSec = 3
```

Device Service Clustering (DSC) was introduced in BIG-IP 11.0.0 and allows many new features such as synchronization and failover between two or more devices. Network failover provides communication between devices for synchronization, failover, and mirroring and is required for the following deployments:

- Sync-Failover device groups containing three or more devices
- Active-active configurations between two BIG-IP platforms
- BIG-IP VIPRION platforms
- BIG-IP Virtual Edition

An active-active pair must communicate over the network to indicate the objects and resources they service. Otherwise, if network communications fail, the two systems may attempt to service the same traffic management objects, which could result in duplicate IP addresses on the network.

A broken network may cause BIG-IP systems to enter into active-active mode. To avoid this issue, F5 recommends that you dedicate one interface on each system to perform only failover communications and, when possible, directly connect these two interfaces with an Ethernet cable to avoid network problems that could cause the systems to go into an active-active state.

> **Important:** When you directly connect two BIG-IP systems with an Ethernet cable, do not change the speed and duplex settings of the interfaces involved in the connection. If you do, depending on the BIG-IP software version, you may be required to use a crossover cable. For more information, refer to SOL9787: Auto MDI/MDIX behavior for BIG-IP platforms.

If you configure a BIG-IP high-availability pair to use network failover, and the hardwired failover cable also connects the two units, hardwired failover always has precedence; if network failover traffic is compromised, the two units do not fail over because the hardwired failover cable still connects them.

Hardwired failover

Hardwired failover is also based on heartbeat detection, where one BIG-IP system continuously sends voltage to another. If a response does not initiate from one BIG-IP system, failover to the peer occurs in less than one second. When BIG-IP redundant devices connect using a hardwired failover cable, the system automatically enables hardwired failover.

The maximum hardwired cable length is 50 feet. Network failover is an option if the distance between two BIG-IP systems exceeds the acceptable length for a hardwired failover cable.

> **Note:** For information about the failover cable wiring pinouts, refer to SOL1426: Pinouts for the failover cable used with BIG-IP platforms.

Hardwired failover can only successfully be deployed between two physical devices. In this deployment, hardwired failover can provide faster failover response times than network failover. However, peer state may be reported incorrectly when using hardwired failover alone. For more information about peer status when using hardwired failover, refer to SOL13888: The active BIG-IP system incorrectly reports the status of its peer device when using only hardwired failover.

Hardwired failover is only a heartbeat and carries no status information. Communication over the network is necessary for certain features to function properly. For example, Traffic Management Microkernel (TMM) uses the network to synchronize packets and flow state updates to peers for connection mirroring. To enable proper state reporting and mirroring, F5 recommends that you configure network failover in addition to hardwired failover.

3.03e - Differentiate between unicast and multicast network failover modes

Failover Modes

The unicast failover configuration uses a self IP address and TMM switch port to communicate failover packets between the HA pair.

When configuring redundant VIPRION systems, F5 recommends that you provide additional redundancy by defining both unicast and multicast failover addresses for each VIPRION system in the device group. The unicast failover configuration uses a self IP address and TMM switch port to communicate failover packets between each VIPRION system, and the multicast failover entry uses the management port to communicate failover packets between each VIPRION system. As an alternative to configuring the multicast failover option, you can define a unicast mesh using the management port for each VIPRION system. This is listed as an alternative option element, following.

3.03f - Identify the cause of failover using logs and statistics

Interpret log messages

You will need to be familiar with reviewing logs and statistics to recognizing the cause of a fail over in an HA pair.

Certain statistical output may reflect that the active unit was not receiving any traffic on a VLAN and when you notice in the logs that a fail over occurred you may attribute it to the VLAN failsafe setting in your configuration. The following is an example of errors you may see in the log if VLAN failsafe triggered:

```
01140029:5: HA vlan_fs vlan1 fails action is reboot.
01140029:5: HA vlan_fs vlan2 fails action is reboot.
```

Objective - 3.04 Given a scenario, determine the cause of loss of high availability and/or sync failure

Scenario Based Objective

To prepare for scenario based questions the candidate will need to complete hands-on configuration and testing of the configuration on the LTM. This will allow the candidate to better understand how different configurations can produce different results. All F5 exams use scenario-based questions that make the candidate apply what they know to a situation to determine the resulting outcome.

This topic is focused on working with high availability environments and causes for failover.

3.04a - Explain how the high availability concepts relate to one another

BIG-IP® TMOS®: ConceptsTroubleshooting ConfigSync and Device Service Clustering Issues (11.x)

Ref: 1, Chapter 8-9.

DSC components

DSC provides the foundation for centralized management and high-availability features in BIG-IP 11.x, including the following components:

- **Device trust and trust domains**
  Device trust establishes trust relationships between BIG-IP devices through certificate-based authentication. Each device generates a device ID key and Secure Socket Layer (SSL) certificate upon upgrade or installation. A trust domain is a collection of BIG-IP devices that trust each other, and can synchronize and fail over their BIG-IP configuration data, as well as regularly exchange status and failover messages.

  When the local BIG-IP device attempts to join a device trust with a remote BIG-IP device, the following applies:

  - If the local BIG-IP device is added as a peer authority device, the remote BIG-IP device presents a certificate signing request (CSR) to the local device, which then signs the CSR and returns the certificate along with its CA certificate and key.
  - If the local BIG-IP device is added as a subordinate (non-authority) device, the remote BIG-IP device presents a CSR to the local device, which then signs the CSR and returns the certificate. The CA certificate and key are not presented to the remote BIG-IP device. The subordinate device is unable to request other devices to join the device trust.

- **Device groups**
  A device group is a collection of BIG-IP devices that reside in the same trust domain and are configured to securely synchronize their BIG-IP configuration and failover when needed. Device groups can initiate a ConfigSync operation from the device group member with the desired configuration change. You can create two types of device groups:

  - A Sync-Failover device group contains devices that synchronize configuration data and support traffic groups for failover purposes.
  - A Sync-Only device group contains devices that synchronize configuration data, but do not synchronize failover objects and do not fail over to other members of the device group.

- **Traffic groups**
  A traffic group represents a collection of related configuration objects that are configured on a BIG-IP device. When a BIG-IP device becomes unavailable, a traffic group can float to another device in a device group.

- **Folders**
  A folder is a container for BIG-IP configuration objects. You can use folders to set up synchronization and failover of configuration data in a device group. You can sync all configuration data on a BIG-IP device, or you can sync and fail over objects within a specific folder only.

3.04b - Explain the relationship between device trust and device groups

Troubleshooting ConfigSync and Device Service Clustering Issues (11.x)

Device trust and trust domains

Device trust establishes trust relationships between BIG-IP devices through certificate-based authentication. Each device generates a device ID key and Secure Socket Layer (SSL) certificate upon upgrade or installation. A trust domain is a collection of BIG-IP devices that trust each other, and can synchronize and fail over their BIG-IP configuration data, as well as regularly exchange status and failover messages.

When the local BIG-IP device attempts to join a device trust with a remote BIG-IP device, the following applies:

- If the local BIG-IP device is added as a peer authority device, the remote BIG-IP device presents a certificate signing request (CSR) to the local device, which then signs the CSR and returns the certificate along with its CA certificate and key.

- If the local BIG-IP device is added as a subordinate (non-authority) device, the remote BIG-IP device presents a CSR to the local device, which then signs the CSR and returns the certificate. The CA certificate and key are not presented to the remote BIG-IP device. The subordinate device is unable to request other devices to join the device trust.

Device groups

A device group is a collection of BIG-IP devices that reside in the same trust domain and are configured to securely synchronize their BIG-IP configuration and failover when needed. Device groups can initiate a ConfigSync operation from the device group member with the desired configuration change. You can create two types of device groups:

- A Sync-Failover device group contains devices that synchronize configuration data and support traffic groups for failover purposes.

- A Sync-Only device group contains devices that synchronize configuration data, but do not synchronize failover objects and do not fail over to other members of the device group.

3.04c - Identify the cause of ConfigSync failures

Troubleshooting ConfigSync and Device Service Clustering Issues (11.x)

ConfigSync Failures

F5 introduced the Device Service Cluster (DSC) architecture in BIG-IP 11.x. DSC provides the framework for ConfigSync, and other high-availability features, such as failover for BIG-IP device groups.

> **Note:** The DSC technology is also referred to as centralized management infrastructure (CMI).

This article provides steps to troubleshoot ConfigSync and the underlying DSC components. DSC and ConfigSync include the following elements (each described in the previous sections):

- Device trust and trust domains
- Device groups
- Traffic groups
- Folders

CMI communication channel

The BIG-IP system uses SSL certificates to establish a trust relationship between devices. In a device trust, BIG-IP devices can act as certificate signing authorities, peer authorities, or subordinate non-authorities. When acting as a certificate signing authority, the BIG-IP device signs x509 certificates for another BIG-IP device that is in the local trust domain. The BIG-IP device for which a certificate signing authority device signs its certificate is known as a subordinate non-authority device. The BIG-IP system uses the following certificates to establish a secure communication channel:

File name	GUI location	Description
/config/ssl/ssl.crt/dtdi.crt	Device Management > Device Trust > Identity	The dtdi.crt is the identity certificate that is used by a device to validate its identity with another device.
/config/ssl/ssl.key/dtdi.key	N/A	The dtdi.key is the corresponding key file used by a device to validate its identity with another device.
/config/ssl/ssl.crt/dtca.crt	Device Management > Device Trust > Local Domain	The dtca.crt is the CA root certificate for the trust network.
/config/ssl/ssl.key/dtca.key	N/A	The dtca.crt is the CA root key for the trust network.

When the DSC components are properly defined, the device group members establish a communication channel to accommodate device group communication and synchronization. The CMI communication channel allows the mcpd process that runs on the device group member to exchange Master Configuration Process (MCP) messages and commit ID updates to determine which device has the latest configuration and is eligible to synchronize its configuration to the group. After the ConfigSync IP addresses are defined on each device, and the device group is created, the devices establish the communication channel, as follows:

1. The local mcpd process connects to the local Traffic Management Microkernel (TMM) process over port 6699.
2. The local TMM uses an SSL certificate (/config/ssl/ssl.crt/dtca.crt) to create a secure connection to the peer TMM using the ConfigSync IP address and TCP port 4353.
3. The peer TMM translates the port to 6699 and passes the connection to the peer mcpd.
4. Once the connections are established, a full mesh exists between mcpd processes for devices in the trust domain.
5. If a device fails to establish the connection for any reason, the local mcpd process attempts to re-establish the connection every 5 seconds.

ConfigSync operation

ConfigSync is a high-availability feature that synchronizes configuration changes from one BIG-IP device to other devices in a device group. This feature ensures that the BIG-IP device group members maintain the same configuration data and work in tandem to more efficiently process application traffic. The ConfigSync operation is dependent on the DSC architecture and the resulting communication channel.

The BIG-IP system uses commit ID updates to determine which device group member has the latest configuration and is eligible to initiate a ConfigSync operation. The configuration transfers between devices as an MCP transaction. The process works as follows:

1. A user updates the configuration of a BIG-IP device group member using the Configuration utility or the TMSH utility.
2. The configuration change is communicated to the local mcpd process.
3. The mcpd process communicates the new configuration and commit ID to the local TMM process.
4. The local TMM process sends the configuration and commit ID update to remote TMM processes over the communication channel.
5. The remote TMM process translates the port to 6699 and connects to its mcpd process.
6. The remote mcpd process loads the new configuration into memory, and writes the configuration changes to the appropriate configuration files.

Automatic Sync

If you enable the Automatic Sync feature for a device group, the BIG-IP system automatically synchronizes changes to a remote peer system's running configuration, but does not save the changes to the configuration files on the peer device. This behavior is by design and recommended for larger configurations to avoid a long ConfigSync duration due to large configurations.

In some cases, you may want to configure Automatic Sync to update the running configuration and save the configuration to the configuration files on the remote peer devices. For information, refer to SOL14624: Configuring the Automatic Sync feature to save the configuration on the remote devices.

Symptoms

DSC and ConfigSync issues may result in the following symptoms:

1. Device group members have configuration discrepancies.
2. The system displays status messages that indicate a synchronization or device trust issue.
3. The BIG-IP system logs error messages related to device trust or the ConfigSync process.

Procedures

When you investigate a possible device service clustering/ConfigSync issue, you should first verify that the required configuration elements are set for all device group members. If the required elements are set, then

attempt a ConfigSync operation. If ConfigSync fails, the BIG-IP system generates Sync status messages that you can use to diagnose the issue.

Troubleshooting a ConfigSync operation

Attempt a ConfigSync operation to gather diagnostic information to help you troubleshoot ConfigSync/DSC issues. To troubleshoot the ConfigSync operation, perform the following procedures:

Verifying the required elements for ConfigSync/DSC

For DSC and ConfigSync to function properly, you must verify that required configuration elements are set. To do so, review the following requirement information:

File name	GUI location	Description
/config/ssl/ssl.crt/dtdi.crt	Device Management > Device Trust > Identity	The dtdi.crt is the identity certificate that is used by a device to validate its identity with another device.
/config/ssl/ssl.key/dtdi.key	N/A	The dtdi.key is the corresponding key file used by a device to validate its identity with another device.
/config/ssl/ssl.crt/dtca.crt	Device Management > Device Trust > Local Domain	The dtca.crt is the CA root certificate for the trust network.
/config/ssl/ssl.key/dtca.key	N/A	The dtca.crt is the CA root key for the trust network.

Viewing the commit ID updates

When you troubleshoot a ConfigSync issue, it is helpful to determine which device group member has the latest commit ID update and contains the most recent configuration. You can then decide whether to replicate the newer configuration to the group, or perform a ConfigSync operation that replicates an older configuration to the group, thus overwriting a newer configuration.

To display the commit ID and the commit ID time stamps for the device group, perform the following procedures:

> **Impact of procedure:** Performing the following procedure should not have a negative impact on your system.

1. Log in to the BIG-IP command line.

2. To display the commit IDs for the device group, type one of the following commands:
   BIG-IP 11.0.0 through 11.2.0:
   watch_devicegroup_device

3. Locate the relevant device group and review the cid.id and cid.time columns.

   For example, the following output shows that the sync_test device group has three members, and device bigip_a has the latest configuration as indicated by the cid.id (commit ID number) and cid.time (commit ID timestamp) columns:

```
devices <devgroup      [device  cid.id  cid.orig        cid.time last_sync
20 21  sync_test      bigip_a  32731  bigip_a.pslab.local   14:27:00   : :
20 21  sync_test      bigip_b  1745  bigip_a.pslab.local   13:39:24  13:42:04
20 21  sync_test      bigip_c  1745  bigip_a.pslab.local   13:39:24  13:42:04
```

   Note: Multiple devices with identical information are collapsed into a single row that displays in green.

4. Perform steps 1 through 3 on all devices in the device group.

5. Compare the commit ID updates for each device with each device group member. If the commit ID updates are different between devices, or a device is missing from the list, proceed to the Troubleshooting DSC section.

Verifying a ConfigSync operation

When troubleshooting a ConfigSync issue, attempt a ConfigSync operation and verify the sync status message. If the ConfigSync operation fails, the BIG-IP system generates a sync status message that you can use to diagnose the issue. To attempt a ConfigSync operation, perform one of the following three procedures:

Configuration utility (11.0.0 through 11.2.0)

Impact of procedure: Performing the following procedure should not have a negative impact on your system.

1. Log in to the Configuration utility.
2. Navigate to Device Management > Device Groups.
3. Click the name of the device group to synchronize.
4. Click the ConfigSync tab.
5. Click Sync Device to Group.

Traffic Management Shell (TMSH)

Impact of procedure: Performing the following procedure should not have a negative impact on your system.

1. Log in to the Traffic Management Shell (TMSH) by typing the following command:
   `tmsh`
2. To synchronize the configuration to the device group, use the following command syntax:
   `run /cm config-sync <option> <device_group>`

   For example, to synchronize the local device configuration to the device group, use the following syntax:
   `run /cm config-sync to-group <device_group>`

Verifying the sync status

After you attempt the ConfigSync operation, you can verify the synchronization status messages and begin to troubleshoot the issue. To verify the synchronization status, refer to the following utilities:

Configuration utility

Version	Configuration utility page	Description
11.0.0 - 11.2.0	Device Management > Device Groups > Group_Name > ConfigSync	The ConfigSync Properties section displays the ConfigSync status for the device.

TMSH utility

Version	tmsh command	Description
11.x	tmsh show /cm sync-status	Displays the ConfigSync status of the local device.

Understanding sync status messages

The BIG-IP system displays ConfigSync status messages for device groups and specific devices. Common synchronization status messages are displayed in the following tables.

Synchronization status messages for device groups

The BIG-IP system displays a number of specific synchronization status messages for each device group. Use the following table to help you troubleshoot messages that you might encounter:

Sync status	Summary	Details	Recommendation
Awaiting Initial Sync	The device group is awaiting the initial ConfigSync	The device group was recently created and has either not yet made an initial sync, or the device has no configuration changes to be synced.	Sync one of the devices to the device group.
Awaiting Initial Sync	hostname-1, hostname-2, etc. awaiting the initial config sync	One or more device group members have not yet synced their data to the other device group members, or a device group member has not yet received a synchronization from another member.	Sync the device that has the most current configuration to the device group.
Changes Pending	Changes Pending	One or more device group members have recent configuration changes that have not been synchronized to the other device group members.	Sync the device that has the most current configuration to the device group.
Changes Pending	There is a possible change conflict between hostname-1, hostname-2, etc.	There is a possible conflict among two or more devices because more than one device contains changes that have not been synchronized to the device group.	View the individual synchronization status of each device group member, and then sync the device that has the most current configuration to the device group.
Not All Devices Synced	hostname-1, hostname-2, etc. did not receive last sync successfully	One or more of the devices in the device group does not contain the most current configuration.	View the individual synchronization status of each device group member, and then sync the device that has the most current configuration to the device group.
Sync Failure	A validation error occurred while syncing to a remote device	The remote device was unable to sync due to a validation error.	Review the /var/log/ltm log file on the affected device.
Unknown	The local device is not a member of the selected device group	The device that you are logged in to is not a member of the selected device group.	Add the local device to the device group.
Unknown	Not logged in to the primary cluster member	The system cannot determine the synchronization status of the device group because you are logged in to a secondary cluster member instead of the primary cluster member. This status pertains to VIPRION systems only.	Use the primary cluster IP address to log in to the primary cluster member.

Sync status	Summary	Details	Recommendation
Unknown	Error in trust domain	The trust relationships among devices in the device group are not properly established.	On the local device, reset device trust and then re-add all relevant devices to the local trust domain.
None	X devices with Y different configurations	The configuration time for two or more devices in the device group differs from the configuration time of the other device group members. This condition causes one of the following synchronization status messages to appear for each relevant device: Device_name awaiting initial config sync Device_name made last configuration change on date_time	Sync the device that has the most current configuration to the device group.

Synchronization status messages for devices

The BIG-IP system displays a number of specific synchronization status messages for individual devices.

Use the following table to help you troubleshoot messages that you might encounter:

Sync status	Summary	Recommendation
Awaiting Initial Sync	The local device is waiting for the initial ConfigSync. The device has not received a sync from another device and has no configuration changes to be synced to other members of the device group.	Determine what device has the latest/desired configuration and perform a ConfigSync from the device.
Changes Pending	The device has recent configuration changes that have not been synced to other device group members.	Sync the device to the device group.
Awaiting Initial Sync with Changes Pending	The configuration on the device has changed since joining the device group, or the device has not received a sync from another device but has configuration changes to be synced to other device group members.	Determine the device with the latest/ desired configuration and perform a ConfigSync operation from the device.

Sync status	Summary	Recommendation
Does not have the last synced configuration, and has changes pending	The device received at least one synchronization previously, but did not receive the last synchronized configuration, and the configuration on the device has changed since the last sync.	Determine the device with the latest/ desired configuration and perform a ConfigSync operation from the device.
Disconnected	The iQuery communication channel between the devices was terminated or disrupted. This may be a result of one of the following: *The disconnected device is not a member of the local trust domain. *The disconnected device does not have network access to one or more device group members.	*Join the disconnected device to the local trust domain. *Verify that the devices have network access using the ConfigSync IP addresses.
Device does not recognize membership in this group	The local device does not recognize that it is a member of the device group.	Add the device to the device group.
No config sync address has been specified for this device	The device does not have a ConfigSync IP address.	Configure a ConfigSync IP address for the device.
Does not have the last synced configuration	The device previously received the configuration from other device group members, but did not receive the last synced configuration.	Perform a ConfigSync operation to sync the device group to the local device.

Reviewing the log files for ConfigSync error messages

The BIG-IP system logs messages related to ConfigSync and DSC to the /var/log/ltm file. To review log files related to ConfigSync and DSC issues, refer to the following commands:

- To display the /var/log/ltm file, use a Linux command similar to the following example:
  ```
  cat /var/log/ltm
  ```
- To display log messages related to DSC or CMI, use a command similar to the following example:
  ```
  grep –i cmi /var/log/ltm
  ```
- To display log messages related to ConfigSync, use a command similar to the following example:
  ```
  grep –i configsync /var/log/ltm
  ```

Troubleshooting DSC

ConfigSync failure is often a result of DSC issues. Perform the following steps to troubleshoot DSC:

Verifying the device trust status

BIG-IP devices must be members of the same local trust domain before you can add them to a device group. When a BIG-IP device joins the local trust domain, it establishes a trust relationship with peer BIG-IP devices that are members of the same trust domain. If the trust relationship among all device group members is not properly established, the synchronization functionality does not work as expected. To verify the device trust status, perform the following procedure:

> **Impact of procedure:** Performing the following procedure should not have a negative impact on your system.

1. Log in to the Traffic Management Shell (TMSH) by typing the following command:
   `tmsh`
2. List the devices in the local trust domain by typing the following command:
   `show /cm device-group device_trust_group`
3. Verify that all device group members are joined to the local trust domain.

   For example, the following output indicates that three devices are joined to the trust domain of the local BIG-IP system:

```
-------------------------------------------------------------------------
CM::Device-Group
Group Name     Member Name     Time since Last Sync (HH:MM:SS)
device_trust_group bigip_a        -
device_trust_group bigip_b       00:15:12
device_trust_group bigip_c       18:29:59
```

4. Perform steps 1 through 3 on all devices in the device group.
5. Compare the list for each device with each device group member.

Verifying time synchronization for device group members

If you experience device trust issues, ensure that all device group members are configured with an NTP server, and that the time across all devices is in synchronization. To verify time synchronization and NTP configuration, perform the following procedure:

Impact of procedure: Performing the following procedure should not have a negative impact on your system.

1. Log in to the command line.
2. Run the following command:
   ```
   date; tmsh list /sys ntp
   ```
3. Perform steps 1 and 2 on all devices in the device group.
4. Verify that each device group member is configured to use an NTP server, and that the times are in synchronization.

   To configure NTP on a BIG-IP system, refer to SOL3122: Using the BIG-IP Configuration utility to add an NTP server.

Troubleshooting the CMI communication channel

When the device clustering components are properly defined, the device group members establish a communication channel to accommodate device group communication and synchronization. To troubleshoot the CMI communication channel, perform the following procedures:

Verifying network access to the ConfigSync IP address

The self IP addresses used for ConfigSync must be defined and routable between device group members. To test network access to another device's ConfigSync IP address, perform the following procedure:

Impact of procedure: Performing the following procedure should not have a negative impact on your system.

1. Obtain the ConfigSync IP address of each device group member.

   For example:
   ```
   tmsh list /cm device configsync-ip
   ```
2. Log in to the command line of one of the device group members.
3. To test network access to each of the other device's ConfigSync IP addresses, use the following command syntax:
   ```
   ping <remote_configsync-ip>
   ```
4. Perform steps 1 through 3 on all devices in the device group.

Verifying the communication channel using netstat

The communication channel allows device group members to exchange MCP messages and commit ID timestamps. When the device trust is working properly, the devices establish a full mesh channel. You can verify this behavior by checking the connection table on each device and ensuring that each device has a connection entry to all other devices in device_trust_group. To do so, perform the following procedure on each device:

Impact of procedure: Performing the following procedure should not have a negative impact on your system.

1. Log in to the command line.
2. Type the following command:
   ```
   netstat -pan | grep -E 6699
   ```
3. Perform steps 1 and 2 on all devices in the device group, and then confirm that the list looks the same on each device.

Monitoring the CMI communications channel for commit ID updates

The sniff-updates program monitors the internal CMI communications channel for commit ID updates. Each commit ID update is linked to an MCP transaction. The system displays each commit ID update as it arrives, one per line. To monitor the CMI communication channel for commit ID updates, perform the following procedure:

Impact of procedure: Performing the following procedure should not have a negative impact on your system.

1. Log in to the Traffic Management Shell (TMSH) by typing the following command:
   ```
   tmsh
   ```
2. To run sniff-updates (add -v for verbose output), type the following command:
   ```
   run /cm sniff-updates
   ```

   For example, the following command output shows a single commit ID update:
   ```
   # run /cm sniff-updates
   [12:48:01] bigip_b (v11.2.1) -> sync_test: CID 32 (bigip_b) at 12:48:01 FORCE_SYNC
   ```

The fields are defined as follows:

Example output	Description
[12:48:01]	Time that the update arrived from the network
bigip_b	Source device
(v11.2.1)	Version of source device
sync_test	Destination device group
CID 32	Commit ID
(bigip_b)	Commit ID originator
12:48:01	Commit ID timestamp
FORCE_SYNC	FORCE_SYNC if set (nothing if not)

Verifying required processes

DSC requires that the following processes are running on all device group members:

- devmgmtd: Establishes and maintains device group functionality.
- mcpd: Allows userland processes to communicate with TMM.
- sod: Provides failover and restart capability.
- tmm: Performs traffic management for the system.

To verify that the processes are running, perform the following procedure:

1. Log in to the command line.
2. Run the following command:
   bigstart status devmgmtd mcpd sod tmm
3. Verify that the processes are all running and that the output appears similar to the following example:
```
devmgmtd   run (pid 7277) 330 hours, 2 starts
mcpd     run (pid 7309) 330 hours, 1 start
sod     run (pid 7339) 330 hours, 1 start
tmm     run (pid 7352) 330 hours, 1 start
```
4. If one or more processes are not running, use the following command syntax to start the service:
```
bigstart start <daemon>
```

Resetting the device trust and re-adding a device to the trust domain

If you verify the elements in the previous section and still have issues, you can reset the device trust and re-add a device to the trust domain by performing the following procedure:

> **Impact of procedure:** F5 recommends that you perform this procedure during a maintenance window. This procedure causes the current device to lose connectivity with all other BIG-IP devices. Depending on the device group and traffic group configuration, the connectivity loss may result in an unintentional Active-Active condition that causes a traffic disruption. For appliance platforms only (non-VIPRION platforms), you can prevent a standby device from going active by setting the standby device in the device group to Force Offline before performing the procedure. Standby devices that were set to Force Offline should be set to Release Offline after performing the procedure.

1. Log in to the Configuration utility of the affected BIG-IP device.
2. Navigate to Device Management > Device Trust > Local Domain.
3. Click Reset Device Trust.
4. Select the Generate new self-signed authority option.
5. Log in to the Configuration utility of the authority BIG-IP device.
6. Navigate to Device Management > Device Trust > Local Domain.
7. Click the Peer List or Subordinate List menu (in BIG-IP 11.0.0 through 11.1.0, scroll to the Peer Authority Devices area or the Subordinate Non-Authority Devices area of the screen).
8. Click Add.
9. Type the management IP address, administrator user name, and administrator password for the affected BIG-IP device, and then click Retrieve Device Information (in BIG-IP 11.0.0 through 11.1.0, click Next).
10. Verify that the certificate and the remote device name are correct, and then click Finished (in BIG-IP 11.0.0 through 11.1.0, click Next and advance through the remaining screens).

Resetting device trust across all devices

> **Warning:** Attempt this procedure only if none of the devices are communicating.

If procedures in the previous sections do not alleviate the issue, then completely reset device trust across all devices. To reset the device trust across all devices, perform the following procedure on all devices in the device group:

Impact of procedure: F5 recommends that you perform this procedure during a maintenance window. This procedure causes the current device to lose connectivity with all other BIG-IP devices. Depending on the device group and traffic group configuration, the connectivity loss may result in an unintentional Active-Active condition that causes a traffic disruption. For appliance platforms only (non-VIPRION platforms), you can prevent a standby device from going active by setting the standby device in the device group to Force Offline before performing the procedure. Standby devices that were set to Force Offline should be set to Release Offline after performing the procedure.

1. Log in to the Configuration utility.
2. Navigate to Device Management > Device Trust > Local Domain.
3. Click Reset Device Trust.
4. Select the Generate new self-signed authority option.
5. Click Update (or Next).
6. Click Finished.
7. Repeat this procedure for each device in the device group.

After you complete the device trust reset on all devices you must set up the device trust.

3.04d - Explain the relationship between traffic groups and LTM objects

Troubleshooting ConfigSync and Device Service Clustering Issues (11.x)

Traffic groups and Configuration Objects

A traffic group represents a collection of related configuration objects that are configured on a BIG-IP device. When a BIG-IP device becomes unavailable, a traffic group can float to another device in a device group.

3.04e - Interpret log messages to determine the cause of high availability issues

ConfigSync May Fail when Using the Management IP Address

Interpret log messages

You will need to be familiar with reviewing logs and recognizing HA errors. The following is an example of errors you may see:

Configuration synchronization (ConfigSync) may fail when using the management IP address.

This issue occurs when all of the following conditions are met:

- BIG-IP Sync-Only or Sync-Failover device group members are configured to use the management IP addresses for ConfigSync.
- Imported external file types similar to the following need to be synchronized to the other device group members:
  - External data group files
  - External monitor program files
  - SSL certificate files
- The ConfigSync operation attempts to sync the imported file object.

When performing a ConfigSync by way of the management IP address, the rsync process is unable to connect and transfer files.

> **Note:** By default, the management address is not available for ConfigSync communication. For the BIG-IP system to use the management IP address for synchronization, you must have manually enabled the configsync.allowmanagement db variable.

Messages that appear similar to the following examples are logged to the **/var/log/ltm** file on the system sending the ConfigSync:

```
err mcpd[<pid>]: 01071392:3: Background command '/usr/bin/rsync --rsync-path=/usr/
bin/rsync -at –blocking io /var/named/config/ rsync://<mgmt_IP_address>/var_name'
failed. The command exited with status 10.
```

```
err mcpd[<pid>]: 01071392:3: Background command '/usr/bin/rsync --rsync-path=/usr/
bin/rsync -at --blocking-io /config/big3d/ rsync://<mgmt_IP_address>/big3d' failed.
The command exited with status 10.
```

Messages that appear similar to the following are logged to the **/var/log/ltm** file on the system receiving the ConfigSync:

```
err mcpd[<pid>]: 0107134b:3: (rsync: failed to connect to <mgmt_IP_address>:
Connection refused (111)) errno(0) errstr().
```

```
err mcpd[<pid>]: 0107134b:3: ( ) errno(0) errstr().
```

```
err mcpd[<pid>]: 0107134b:3: (rsync error: error in socket IO (code 10) at
clientserver.c(110) [receiver=2.6.8]) errno(0) errstr().
```

```
err mcpd[<pid>]: 0107134b:3: ( ) errno(0) errstr().

err mcpd[<pid>]: 0107134b:3: (syncer /usr/bin/rsync failed! (2560) () ) errno(0)
errstr().

info mcpd[<pid>]: 0107134b:6: Per-invocation log rate exceeded; throttling.

err mcpd[<pid>]: 0107134b:3: (rsync process failed.) errno(255) errstr().

notice mcpd[<pid>]: 01071038:5: Loading keys from the file.

err mcpd[<pid>]: 01070712:3: Caught configuration exception (0), Failed to sync
files. - sys/validation/FileObject.cpp, line 5821.

err mcpd[<pid>]: 01071488:3: Remote transaction for device group /Common/<device_
group> to commit id 814 5868240562026205561 /Common/LTM1.example.com failed with
error 01070712:3: Caught configuration exception (0), Failed to sync files. - sys/
validation/FileObject.cpp, line 5821..
```

Conclusion

This document is intended as a study guide for the F5 301b – BIG-IP LTM Specialist: Maintain and Troubleshoot exam. This study guide is not an all-inclusive document that will guarantee a passing grade on the exam. It is intended to be a living doc and any feedback or material that you feel should be included, to help exam takers better prepare, can be sent to channeleng@f5.com.

Thank you for using this study guide to prepare the 301b – BIG-IP LTM Specialist exam and good luck with your certification goals.

Thanks
Eric Mitchell
Channel FSE, East US and Federal

F5 Networks, Inc. 401 Elliott Avenue West, Seattle, WA 98119 888-882-4447 f5.com

Americas info@f5.com
Asia-Pacific apacinfo@f5.com
Europe/Middle-East/Africa emeainfo@f5.com
Japan f5j-info@f5.com