



Deploying the BIG-IP LTM for Diameter Traffic Management

Important: This guide has been archived. While the content in this guide is still valid for the products and versions listed in the document, it is no longer being updated and may refer to F5 or third party products or versions that have reached end-of-life or end-of-support. For a list of current guides, see <https://f5.com/solutions/deployment-guides>.

Table of Contents

Configuring the BIG-IP LTM for Diameter traffic management	
Why Diameter support is valuable	1
Product versions and revision history	2
Configuring the BIG-IP LTM for Diameter	3
Creating the diameter monitor	3
Creating the pool	4
Creating the profiles	6
Creating the virtual server	9
Advanced configuration options	11
Preventing FIN/RST/Expire propagation from server to client	11
Creating an iRule to handle Server-initiated messages	13
Appendix: Additional information	16
Traffic flow	16
Watchdog handling	17
Host-IP-Address AVP	17
Destination-Host/Realm AVP	18
Origin-Host/Realm AVP	18
Capability-Exchange handling	18
1 by N message-based load balancing	19
Idle Connections	19

Configuring the BIG-IP LTM for Diameter traffic management

Welcome to the F5 deployment guide for Diameter traffic management. This guide provides step-by-step procedures for configuring the BIG-IP Local Traffic Manager (LTM) for load balancing and intelligent traffic management for the Diameter protocol.

The Diameter base protocol is intended to provide an Authentication, Authorization and Accounting (AAA) framework for applications such as network access or IP mobility. Diameter is also intended to work in both local Authentication, Authorization & Accounting and roaming situations (this is an excerpt from RFC3588; for more information, see the complete RFC: <http://www.ietf.org/rfc/rfc3588.txt>).

For more information on the BIG-IP LTM, see www.f5.com/products/big-ip/product-modules/local-traffic-manager.html

Why Diameter support is valuable

The ability to support diameter traffic management is extremely valuable. Consider the following example. In a typical load balancing situation, there are X number of clients and Y number of servers. If all clients generate one connection, there are X connections total. The BIG-IP LTM may balance X/Y connections to each server (which may be called connection-based load balancing). However, in a Diameter environment, the number of clients is likely to be small (X may even lower than Y) and that implies low number of connections (X). Moreover, each connection is long-lived, which provides few opportunities to load balance Diameter traffic on a per-connection basis.

Multiple sessions may be established within the one transport connection. Diameter keeps transport connections (TCP/SCTP) alive and reuses them for many Diameter sessions. Each Diameter session may contain multiple messages. Diameter protocol is asynchronous, or in other words, a client can send a new request without waiting for response for the previous request. The Server can send a response in any order, and can also send request.

In a high load environment, there is a need for per-message load balancing or message-based load balancing instead of connection-based load balancing. Imagine there is one transport connection between each client (NAS) and server (Diameter host server). The work required for a Diameter server to generate a response is significantly higher than the work required for a client to generate a request. Because of this, the Diameter server becomes a performance bottleneck for AAA requests from a single client. All requests from a particular client which using the same transport connection are served by only one server. By supporting message-based load balancing, the BIG-IP LTM may act as a proxy that will de-multiplex each request from the client to multiple servers and improve overall performance and scalability.

See *Traffic flow*, on page 16 for an example of the LTM/Diameter traffic flow.

Product versions and revision history

Product and versions tested for this deployment guide:

Product Tested	Version Tested
BIG-IP system	v10.2

Document Version	Description
1.0	New deployment guide
1.1	Changed title and some headings from “Diameter load balancing” to “Diameter Traffic Management” to more accurately reflect the role of the BIG-IP LTM.
1.2	Corrected the iRule on page 13 to remove an unnecessary bracket.

◆ Note

*This document is written with the assumption that you are familiar with the BIG-IP LTM system. For more information, see the **Configuration Guide for BIG-IP Local Traffic Manager** and the **Implementations Guide** available on Ask F5 (<http://support.f5.com/kb/en-us.html>).*

Configuring the BIG-IP LTM for Diameter

Use the following procedures to configure the BIG-IP LTM for Diameter traffic management. When you are finished, be sure to see *Advanced configuration options*, on page 11 for optional configuration procedures.

Creating the diameter monitor

The first task is to create the diameter health monitor. For more information on how diameter monitor works see Solution SOL11681 - “Overview of the diameter monitor”

(<http://support.f5.com/kb/en-us/solutions/public/11000/600/sol11681.html>).

To create the diameter health monitor

1. On the Main tab, expand **Local Traffic**, and then click **Monitors**. The Monitors screen opens.
2. Click the **Create** button. The New Monitor screen opens.
3. In the **Name** box, type a name for the Monitor. In our example, we type **diameter-monitor**.
4. From the **Type** list, select **Diameter**. The Diameter Monitor configuration options appear.
5. In the Configuration section, in the **Interval** and **Timeout** boxes, type an Interval and Timeout. We recommend at least a 1:3 +1 ratio between the interval and the timeout (for example, the default setting has an interval of **10** and an timeout of **31**). In our example, we use the default settings.
6. Optional: If you are using the CARP Hash persistence method (see *Creating the persistence profile (optional)*, on page 6), we recommend you set **Manual Resume** to **Yes** (you must first select **Advanced** from the Configuration list). For more information, see *Using CARP hash persistence*, on page 7.
7. Configure the rest of the settings as applicable for your configuration. For more information on these settings, see the online help. In our example, we leave the defaults.
8. Click the **Finished** button.

Local Traffic » Monitors » **New Monitor...**

General Properties

Name	diameter-monitor
Type	Diameter
Import Settings	diameter

Configuration: Advanced

Interval	10 seconds
Up Interval	Disabled
Time Until Up	0 seconds
Timeout	31 seconds
Manual Resume	<input checked="" type="radio"/> Yes <input type="radio"/> No
Origin Host	
Origin Realm	f5.com
Host IP Address	
Vendor ID	3375
Product Name	F5 BIGIP Diameter Health Monitoring
Auth Application ID	None
Acct Application ID	None
Vendor Specific Application ID	None
Vendor Specific Vendor ID	
Vendor Specific Auth Application ID	
Vendor Specific Acct Application ID	

Cancel Repeat Finished

Figure 1 New Diameter monitor

Creating the pool

The next task is to create a load balancing pool for the Diameter devices that uses the health monitor you just created.

To create the Diameter pool

1. On the Main tab, expand **Local Traffic**, and then click **Pools**.
1. Click the **Create** button. The New Pool screen opens.

2. From the **Configuration** list, select **Advanced**.
3. In the **Name** box, enter a name for your pool.
In our example, we use **diameter-pool**.
4. In the **Health Monitors** section, select the name of the monitor you created in *Creating the diameter monitor*, on page 3, and then click the Add (<<) button. In our example, we select **diameter-monitor**.
5. From the **Load Balancing Method** list, select **Round Robin**.
6. For this pool, we leave the Priority Group Activation **Disabled**.
7. In the New Members section, in the **Address** box, add the first server to the pool. In our example, we type **192.0.2.123**
8. In the **Service Port** box, type the service number you want to use for this device, or specify a service by choosing a service name from the list. In our example, we type **3868**, the default port for Diameter.
9. Click the **Add** button to add the member to the list.
10. Repeat steps 7-9 for each server you want to add to the pool.
11. Click the **Finished** button.

Figure 2 New Diameter pool

Creating the profiles

The next task is to create the BIG-IP profiles. A *profile* is an object that contains user-configurable settings, with default values, for controlling the behavior of a particular type of network traffic. Using profiles enhances your control over managing network traffic, and makes traffic-management tasks easier and more efficient.

Although it is possible to use the default profiles, we strongly recommend you create new profiles based on the default parent profiles. Creating new profiles allows you to easily modify the profile settings specific to this deployment, and ensures you do not accidentally overwrite the default profile.

Creating the persistence profile (optional)

The next task is to create a persistence profile. Creating a persistence profile is optional and only necessary if persistence is required in your implementation.

There are two options for creating a persistence profile for diameter:

- *Creating a custom universal persistence profile*
- *Using CARP hash persistence*

Creating a custom universal persistence profile

You can optionally create a custom persistence profile. By default, the Diameter profile you create in an upcoming section uses the base Universal persistence profile. If you do not need to modify any of the default settings in the Universal persistence profile, such as a specific timeout, or matching across pools or virtual servers, you do not need to create this profile (or apply a persistence profile to the virtual server). Continue with *Creating the Diameter profile*, on page 8.

If you want to modify the default settings, the following procedure shows how to create a new persistence profile based on the Universal parent profile.

To create a custom universal persistence profile

1. On the Main tab, expand **Local Traffic**, and then click **Profiles**.
2. On the Menu bar, click **Persistence**.
3. Click the **Create** button. The New Persistence Profile screen opens.
4. In the **Name** box, type a name for this profile. In our example, we type **diameter-universal**.
5. From the **Persistence Type** list, select **Universal**.
The configuration options for Universal persistence appear.
6. Modify any of the settings as applicable for your network. See the online help for information on specific fields.
7. Click the **Finished** button.

Using CARP hash persistence

You can choose to configure CARP hash persistence for increased performance and scalability. This method is suitable for organizations who want the diameter messages to be persisted for a very long time periods.

The CARP hash persistence method does not store the persistence records, so it reduces delay and the amount of processing power used by persistence look ups. This is also reduces processing power and network traffic in case the high availability deployment and persistence records have to be mirrored. Because CARP hash does not store the persistent records, so there is no need to mirror any persistent information across HA units.

The disadvantage of using CARP hash persistence compared to using universal persistence is when a new server is added (or comes back online after the health monitor detects it was down). The hash result changes and some requests from clients which belong to active sessions may be forwarded to the wrong server. However, in theory, if a single server is added, only around 1/N (where N is total number of servers in the pool) of requests may have their hash result change.

If CARP hash persistence is used, we recommend that the new server should be added to the pool during a maintenance period or a time with the least amount of traffic. We also recommend you enable the **manual resume** option for health monitor.

For more information on the CARP hash algorithm, see <https://support.f5.com/kb/en-us/solutions/public/11000/300/sol11362.html> or the BIG-IP documentation.

To create the persistence profile

1. On the Main tab, expand **Local Traffic**, and then click **Profiles**.
2. On the Menu bar, click **Persistence**.
3. Click the **Create** button.
4. In the **Name** box, type a name for this profile. In our example, we type **CARP-persistence**.
5. From the **Persistence Type** list, select **Hash**.
6. In the **Hash Algorithm** row, click the **Custom** box, and then select **CARP** from the list.
7. Configure any other settings as applicable for your configuration. In our example, we leave the settings at the defaults.
8. Click **Finished**.

General Properties	
Name	CARP-persistence
Persistence Type	Hash
Parent Profile	hash
Configuration Custom <input type="checkbox"/>	
Match Across Services	<input type="checkbox"/>
Match Across Virtual Servers	<input type="checkbox"/>
Match Across Pools	<input type="checkbox"/>
Hash Algorithm	CARP <input checked="" type="checkbox"/>
Hash Offset	0 <input type="checkbox"/>
Hash Length	0 <input type="checkbox"/>
Hash Start Pattern	<input type="checkbox"/>
Hash End Pattern	<input type="checkbox"/>
Hash Buffer Limit	0 <input type="checkbox"/>

Figure 3 New CARP hash persistence profile

Creating the Diameter profile

The next task is to create the Diameter profile. By default the Diameter profile uses the session-id as a persistent key. It also uses universal persistence as the underlying persistence technology.

If persistence is not required in your implementation, select **None** in step 5.

For more information on creating the Diameter profile, see the Online help or the product documentation.

To create the diameter profile

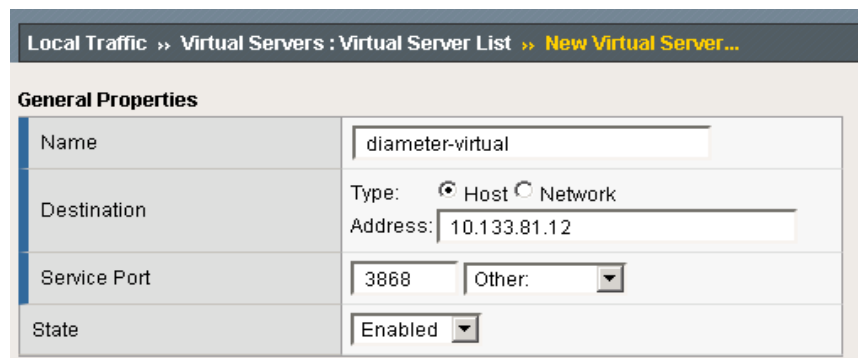
1. On the Main tab, expand **Local Traffic**, and then click **Profiles**.
2. On the Menu bar, from the **Services** menu, click **Diameter**.
3. Click the **Create** button.
4. In the Name box, type a name for this profile. In our example, we type **diameter-profile**.
5. Only if persistence is *not* required in your implementation, from the **Persist Attribute** row, click the **Custom** box, and then select **None** from the list.
6. Configure any of the other settings as applicable for your configuration.
7. Click **Finished**.

Creating the virtual server

Next, we configure a virtual server that references the profiles and pool you created in the preceding procedures.

To create the virtual server

1. On the Main tab, expand **Local Traffic**, and then click **Virtual Servers**.
2. Click the **Create** button.
3. In the **Name** box, type a name for this virtual server. In our example, we type **diameter-virtual**.
4. In the **Destination** section, select the **Host** option button.
5. In the **Address** box, type the IP address of this virtual server. In our example, we use **10.133.81.12**.
6. In the **Service Port** box, type **3868**.



General Properties	
Name	diameter-virtual
Destination	Type: <input checked="" type="radio"/> Host <input type="radio"/> Network
	Address: 10.133.81.12
Service Port	3868 Other: <input type="text"/>
State	Enabled <input type="text"/>

Figure 4 General Properties of the virtual server

7. From the Configuration list, select **Advanced**.
The Advanced configuration options appear.
8. Leave the **Type** list at the default setting: **Standard**.
9. From the **Diameter Profile** list select the name of the profile you created *Creating the Diameter profile*, on page 8. In our example, we select **diameter-profile**.
10. From the **SNAT Pool** list, select **Automap**.
Note: If your servers are configured with the BIG-IP device as their default gateway, this step is not necessary.
11. In the Resources section, from the **Default Pool** list, select the pool you created in *Creating the pool*, on page 4. In our example, we select **diameter-pool**.
12. From the **Default Persistence Profile** list, select the Universal or CARP hash persistence profile you created in *Creating the persistence profile (optional)*, on page 6.

If you did not create a custom Universal persistence profile, and want the Diameter profile to use the default Universal persistence profile, do not select a persistence profile in this step. In our example, we select **CARP-persistence**.

- Click the **Finished** button.

SMTP Profile	None
Diameter Profile	diameter-profile
SIP Profile	None
VLAN and Tunnel Traffic	All VLANs and Tunnels
SNAT Pool	Auto Map

Resources

iRules	Enabled	Available
	<div style="display: flex; justify-content: space-between;"> <div style="border: 1px solid gray; width: 40%; height: 40px;"></div> <div style="border: 1px solid gray; width: 40%; height: 40px;"></div> </div> <div style="display: flex; justify-content: center; gap: 10px; margin-top: 5px;"> << >> </div> <div style="display: flex; justify-content: center; gap: 10px; margin-top: 5px;"> Up Down </div>	
Default Pool	+	diameter-pool
Default Persistence Profile		CARP-persistence
Fallback Persistence Profile		None

Cancel Repeat Finished

Figure 5 Virtual server configuration and resources (truncated to show relevant settings)

Advanced configuration options

In this section, we provide some advanced configuration procedures that are applicable to some implementations. These procedures are optional.

This section is divided into the following parts:

- *Preventing FIN/RST/Expire propagation from server to client*
- *Creating an iRule to handle Server-initiated messages*, on page 13

Preventing FIN/RST/Expire propagation from server to client

In Message-Based Load Balancing (MBLB), some administrators may prefer to hide a problem that happens to a single server so it does not affect the rest of the connections. With the current behavior, LTM propagates the following action from server-side to client-side:

- a server sends FIN to close connection
- a server sends RST when a problem occurs
- a connection to the server has reached the idle timeout and LTM issues an RST to terminate the connection.

To prevent these actions to be propagated to the client-side, use the following procedures.

Adding the MBLB profile to the virtual server

The first task is to add the MBLB profile to the virtual server. This must be done using the **tmsh** shell.

To add the profile using the tmsh shell

1. On the BIG-IP system, start a console session.
2. Type a user name and password, and then press Enter.
3. To add the profile to the virtual server, use the following syntax:

```
tmsh modify ltm virtual <virtual server name> profiles
add { mblb }
```

In our example, we type:

```
tmsh modify ltm virtual diameter-virtual profiles add { mblb }
```

4. To save the changes, type the following command:

```
tmsh save sys config
```

Creating the iRule

The next task is to create an iRule that periodically checks if the client connection closed, and if so, it also closes associated server connections.

To create the iRule

1. On the Main tab, expand **Local Traffic**, and then click **iRules**.
2. Click the **Create** button.
3. In the **Name** box, type a name. We type **diameter-irule**.
4. In the Definition section, copy and paste the following iRule, omitting the line numbers:

```

1  when CLIENT_ACCEPTED {
2      set client_closed 0
3  }
4  when CLIENT_CLOSED {
5      set client_closed 1
6  }
7  when SERVER_CONNECTED {
8      after 1000 -periodic if { $client_closed } { TCP::close; }
9  }

```

5. Click **Finished**.
6. See *Modifying the virtual server to reference the iRule(s)*, on page 15 to associate the iRule with the Diameter virtual server you created.

Preventing client connections from being reset between monitor intervals

There is also the case the server might be down but health monitor does not detect it yet (for example, if the health monitor is configured to run every 15 seconds, there may be a gap up to 15 seconds that a server is down but it has not been detected). In this case, the BIG-IP LTM could choose the unavailable server, and so it would fail to send the Diameter message. This would cause the LTM to reset the client connection. To prevent the client connection from being reset in this manner, create the following iRule.

To create the iRule

1. On the Main tab, expand **Local Traffic**, and then click **iRules**.
2. Click the **Create** button.
3. In the **Name** box, type a name. We type **diameter-reset-irule**.

-
4. In the Definition section, copy and paste the following iRule, omitting the line numbers:

```
1  when LB_FAILED {
2      if { [active_members [LB::server pool]] > 0 } {
3          after 100
4              LB::reselect pool [LB::server pool]
5          }
6      }
```

5. Click **Finished**.
6. See *Modifying the virtual server to reference the iRule(s)*, on page 15 to associate the iRule with the Diameter virtual server you created.

Creating an iRule to handle Server-initiated messages

In most Diameter traffic management deployments, clients send requests and servers reply with responses. However, the Diameter protocol is also designed to support server-initiated messages. There are some diameter applications/deployments that not only clients that send requests, servers may send the requests as well.

In the current version of the BIG-IP LTM, to support server-initiated message an additional iRule is required. In future versions, support for server-initiated messages will be added to diameter profile (and will not require the iRule). Refer to CR140615 for more information.

The following iRule adds support for server-initiated message. The iRule looks for session-id from server-initiated message and uses it as a persistent key.

◆ Note

This iRule cannot be used with CARP (either do not apply a persistence profile, or apply the UIE persistence profile).

To create the iRule

1. On the Main tab, expand **Local Traffic**, and then click **iRules**.
2. Click the **Create** button.
3. In the **Name** box, type a name. We type **diameter-server-irule**.
4. In the Definition section, copy and paste the following iRule, omitting the line numbers:

```

1  when CLIENT_ACCEPTED {
2      set create_client_context 1
3  }
4  when SERVER_CONNECTED {
5      TCP::collect
6  }
7  when SERVER_DATA {
8      while { [TCP::payload length] > 20 } {
9          binary scan [TCP::payload] II a b
10         set comcode [ expr $b & 0xffffffff ]
11         set mlen [ expr $a & 0xffffffff ]
12         set rflag [expr ($b >> 31)&1 ]
13         if { [TCP::payload length] < $mlen } {
14             TCP::collect
15             return
16         }
17
18         if { $rflag } {
19             switch $comcode {
20                 257 -
21                 280 -
22                 282 {
23                     # do nothing
24                 }
25                 default {
26                     # find session-id
27                     set index 20
28                     while { $index < $mlen } {
29                         binary scan [TCP::payload $mlen] @${index}III avp_code avp_len vendor_id
30                         set avp_len [ expr $avp_len & 0xffffffff ]
31                         set avp_len_pad [expr (($avp_len + 3)/4)*4]
32                         if { $avp_code == 263 } {
33                             set avp_flag_v [ expr ($avp_len >> 31)&1 ]
34                             if { $avp_flag_v == 0 } {
35                                 set avp_dlen [ expr $avp_len - 8 ]
36                                 binary scan [TCP::payload $mlen] @${index}x8a${avp_dlen} sid
37                             } else {
38                                 set avp_dlen [ expr $avp_len - 12 ]
39                                 binary scan [TCP::payload $mlen] @${index}x12a${avp_dlen} sid
40                             }
41                             persist add uie $sid
42                             break
43                         }
44                         incr index $avp_len_pad
45                     }
46                 }
47             }
48         }
49         TCP::release $mlen
50     }
51     TCP::collect
52 }

```

5. Click **Finished**.

6. See *Modifying the virtual server to reference the iRule(s)*, on page 15 to associate the iRule with the Diameter virtual server you created.

Modifying the virtual server to reference the iRule(s)

The next task is to modify the virtual server you created in *Creating the virtual server*, on page 9 to use the iRule or iRules you just created.

To modify the existing virtual server

1. On the Main tab, expand **Local Traffic**, and then click **Virtual Servers**.
2. From the **Virtual Server** list, click the virtual server you created in the *Creating the virtual server*, on page 9 section. In our example, we click **diameter-virtual**.
3. On the menu bar, click **Resources**.
The Resources page for the virtual server opens.
4. In the iRules section, click the **Manage** button.
The Resource Management screen opens.
5. From the **Available** list, select the iRule you created in *Creating the iRule*, and click the Add (<<) button. In our example, we select **diameter-irule**.
6. Repeat step 5 for any additional iRules you created in this section.
7. Click the **Finished** button.

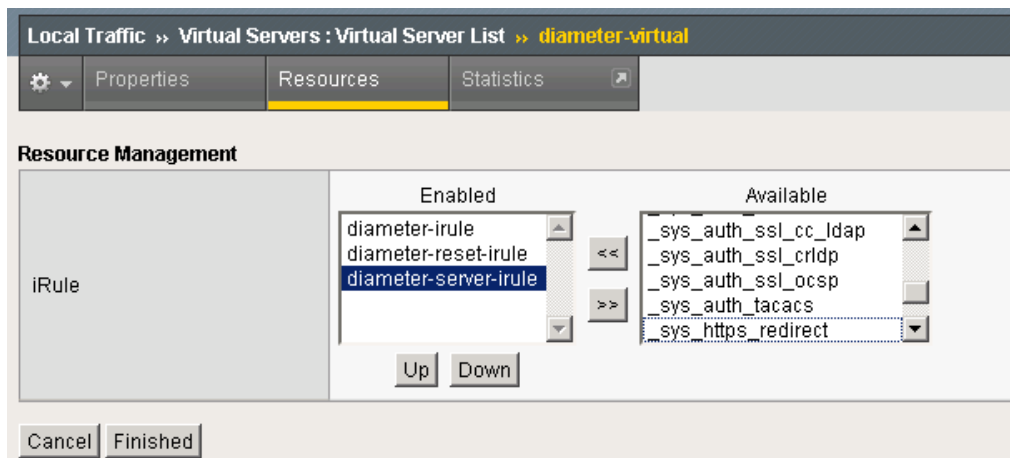


Figure 6 Virtual Server Resource Management page

Appendix: Additional information

This section contains additional information that you might find useful when configuring the BIG-IP LTM for Diameter.

Traffic flow

The following diagram contains an example of the traffic flow for a load balanced Diameter implementation.

Note

This traffic flow diagram is written with the following assumptions: this is the first time the LTM has delivered traffic to the server and there are no persistent records; the round robin load balancing algorithm is used, and the persistent key is session-id AVP.

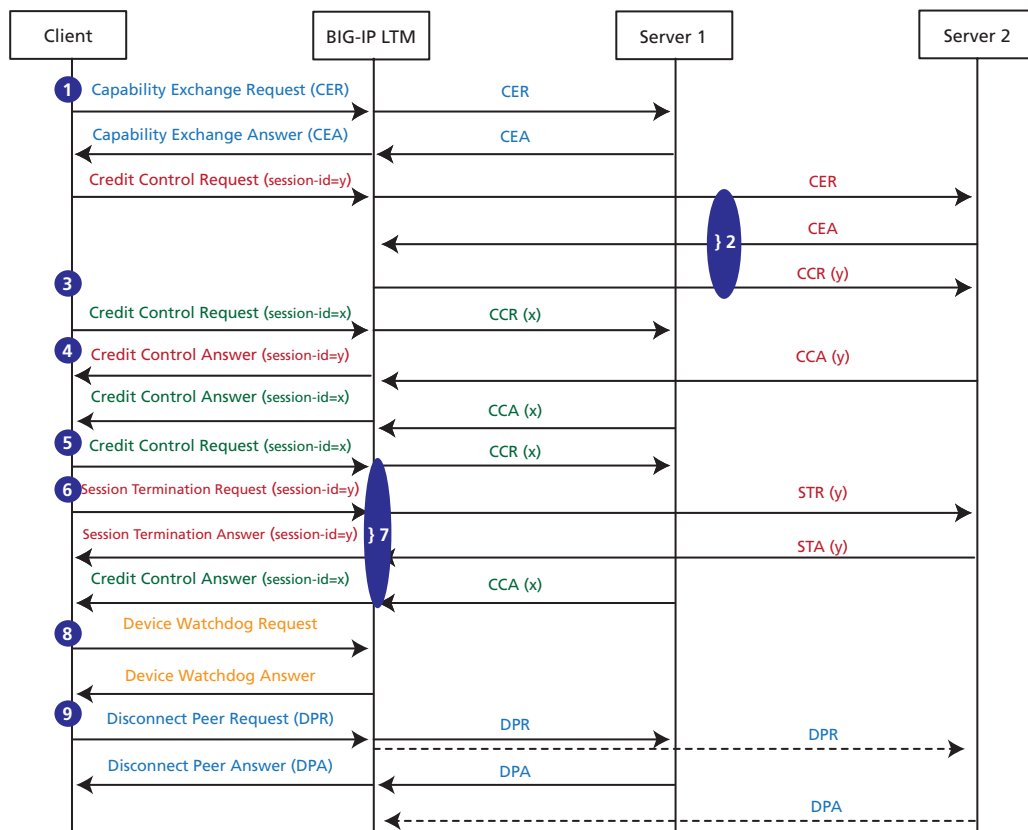


Figure 7 Load balanced Diameter traffic flow example

1. The Diameter Handshake or Capability Exchange Request is initiated and load balanced by the BIG-IP LTM to server 1.

-
2. The next Diameter request has session-id = y. It has never been seen before, so the BIG-IP LTM sends the request to server 2. However, prior to sending the request, the BIG-IP LTM performs the Diameter Handshake with the server. The BIG-IP LTM uses the session-id AVP as a persistent key. This request has session-id = y.
 3. This request has session-id = x. It has never been seen before, so the BIG-IP LTM sends the request to server 1.
 4. This request has session-id = y. The BIG-IP LTM persists this request to server 2.
 5. This request has session-id = x. The BIG-IP LTM persists this request to server 1.
 6. This request has session-id = y. The BIG-IP LTM persists this request to server 2.
 7. The Diameter protocol is asynchronous. The response from the server can come any time and in any order. Note STA(y) is sent back to the client prior to CCA(x).
 8. When the client or server sends a Device Watchdog Request, the BIG-IP LTM responds with a Device Watchdog Answer (the BIG-IP LTM does not forward DWR).
 9. When the client sends the Disconnect Peer Request, the LTM broadcasts the request to all servers.
If the server sends the Disconnect Peer Request, the LTM responds with a Disconnect Peer Answer to that particular server only. The connections to the client and other servers are not affected.

Watchdog handling

In current version of the BIG-IP LTM, when the client or server sends a watchdog request to the BIG-IP LTM, the LTM responds to the watchdog request. In version 10.1, there is a known issue that the watchdog response from the BIG-IP LTM may contain an empty Origin-Host and Origin-Realm AVP. This issue is fixed in version 10.2.

Refer to CR137617 for more information.

In a future release, the BIG-IP LTM will be able to initiate watchdog requests if the connection is idle.

Host-IP-Address AVP

Starting in BIG-IP LTM version 10.2, the Host-IP-Address will be rewritten to appropriate value. For example, if the BIG-IP LTM performs SNAT, for all messages from the BIG-IP LTM to the servers, the Host-IP-Address will be rewritten to the SNAT address. Refer to CR137700 for more information.

Destination-Host/Realm AVP

In some implementations, the Diameter servers may discard the Diameter request if the Destination-Host and/or Destination-Realm AVP does not match their identity. The current version contains an option to customize both AVPs.

For the Destination-Realm AVP, you can specify a string value to which the BIG-IP LTM will rewrite all Diameter requests.

For the Destination-Host AVP, it is just an option to enable or disable. If it is enabled, the BIG-IP LTM will rewrite the Destination-Host AVP to Diameter server's IP address.

In future releases, the BIG-IP LTM will be able to rewrite the Destination-Host AVP to the value learned during the Diameter handshake (the value from Capability Exchange Answer - CEA).

Origin-Host/Realm AVP

In some implementations, Diameter clients or servers may require the BIG-IP LTM to rewrite Origin-Host/Realm AVP to LTM's Identity. This will be supported in a future release of the BIG-IP LTM.

Capability-Exchange handling

In the current version of the BIG-LTM, when the client makes a TCP connection and sends a Capability-Exchange-Request (CER), the BIG-IP LTM forwards the CER to one of servers in the pool, and then waits for server response. Once the server replies with a Capability-Exchange-Answer (CEA), the LTM forwards the CEA message back to the client. Typically, if the CEA message is a positive response, the client starts sending other Diameter requests, which will then be load balanced per each message by LTM. (If the CEA message is negative, the client and/or the server may terminate the TCP connection).

When the client sends Diameter requests (after the CER/CEA handshake is complete), if the BIG-IP LTM picks a server which does not yet have connection open, the BIG-IP makes a new TCP connection to the server, sends a CER, and waits for CEA. If the server replies with a positive response, the LTM forwards the client's Diameter request to the server. Otherwise, the LTM will drop the connection to the server and log the following message: **diameter hud_dime_handle error Prerequisite operation not in progress**. (This log message will be changed to be more informative in future release).

I by N message-based load balancing

For each client connection, the LTM makes one connection to each server. We refer to this as 1 by N message-based load balancing. Diameter requests from multiple client connections will not be multiplexed to same server connection.

Idle Connections

If the client or server stops sending any Diameter message, the LTM will terminate the connection after the idle timeout has been reached. The LTM will also terminate the associated connections. For example, if there are 3 servers (A, B, C) in pool, a client makes a connection to the LTM and the LTM establishes connections to all 3 servers. If the connection to server-A is idle, the LTM will terminate the TCP connection to server-A, and then the connection to server-B, server-C and the client.

To prevent this connection termination from being propagated, follow the procedure *Preventing FIN/RST/Expire propagation from server to client*, on page 11.

If the client or server stops sending Diameter messages, but is still sending Device-Watchdog-Requests (DWR) periodically, the LTM will respond to DWR and maintain the connection.