BEA WebLogic SIP Server™

Configuring Network Resources

Version 3.1
Revised: July 16, 2007

Copyright

Restricted Rights Legend

Trademarks and Service Marks

Contents

3. Example WebLogic SIP Server Network Configuration

4. Configuring Diameter Client Nodes and Relay Agents

CHAPTER 1

Managing WebLogic SIP Server Network Resources

The following sections describe how to configure network resources for use with WebLogic SIP Server:

Overview of Network Configuration

The default HTTP network configuration for each WebLogic SIP Server instance is determined from the Listen Address and Listen Port setting for each server. However, WebLogic SIP Server does not support the SIP protocol over HTTP. The SIP protocol is supported over the UDP and TCP transport protocols. SIPS is also supported using the TLS transport protocol.

To enable UDP, TCP, or TLS transports, you configure one or more *network channels* for a WebLogic SIP Server instance. A network channel is a configurable WebLogic Server resource that defines the attributes of a specific network connection to the server instance. Basic channel attributes include:

- The protocols supported by the connection
- The listen address (DNS name or IP address) of the connection
- The port number used by the connection
- (optional) The port number used by outgoing UDP packets
- The public listen address (load balancer address) to embed in SIP headers when the channel is used for an outbound connection.

You can assign multiple channels to a single WebLogic SIP Server instance to support multiple protocols or to utilize multiple interfaces available with multihomed server hardware. You cannot assign the same channel to multiple server instances.

When you configure a new network channel for the SIP protocol, both the UDP and TCP transport protocols are enabled on the specified port. You cannot create a SIP channel that supports only UDP transport or only TCP transport. When you configure a network channel for the SIPS protocol, the server uses the TLS transport protocol for the connection.

As you configure a new SIP Server domain, you will generally create multiple SIP channels for communication to each engine tier server in your system. Engine tier servers can communicate to data tier replicas using the configured Listen Address attributes for the replicas. Note, however, that replicas must use unique Listen Addressees in order to communicate with one another.

Note: If you configure multiple replicas in a data tier cluster, you must configure a unique Listen Address for each server (a unique DNS name or IP address). If you do not specify a unique Listen Address, the replica service binds to the default “localhost” address and multiple replicas cannot locate one another.

IPv4 and IPv6

If your operating system and hardware support IPv6, you can also configure WebLogic SIP Server to use IPv6 for network communication. IPv6 for SIP traffic is enabled by configuring a network channel with an IPv6 address. You must configure an IPv6 SIP channel on each engine tier server that will support IPv6 traffic.

Note: The Windows XP IPv6 implementation does not support dual mode sockets, and cannot be used with any available JVMs to support IPv6. Using IPv6 with a network channel throws the following exception with either the Sun or JRockit JVMs:

```
Address family not supported by protocol family: bind.
```

For more information see
http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6230761

Note that each SIP network channel configured on an engine supports either IPv6 or IPv4 traffic. You cannot mix IPv4 and IPv6 traffic on a single channel. A single engine can be configured with both an IPv4 and IPv6 channel to support multiple, separate networks.

It is also possible for WebLogic SIP Server engine and data tier nodes to communicate on IPv4 (or IPv6) while supporting the other protocol version for external SIP traffic. To configure engine and data tier nodes on an IPv6 network, simply specify IPv6 listen addresses for each server instance.

See "Creating a New SIP or SIPS Channel" on page 1-6 for information about configuring IPv4 and IPv6 network channels.

Configuring Load Balancer Addresses

If your system uses one or more load balancers to distribute connections to the engine tier, you must configure SIP network channels to include a load balancer address as the *external listen address*. When a SIP channel has an external listen address that differs from the channel's primary listen address, WebLogic SIP Server embeds the host and port number of the external address in SIP headers such as `Response`. In this way, subsequent communication for the call is directed to the public load balancer address, rather than the local engine tier server address (which may not be accessible to external clients).

If a network channel does not have a configured external listen address, the primary listen address is embedded into SIP headers.

Multiple Load Balancers and Multihomed Load Balancers

If your system uses two load balancers, you must define two channels on each engine tier server (one for each network connection to each load balancer) and assign the external listen address to the corresponding load balancer. When a particular network interface on the engine tier server is selected for outbound traffic, the network channel associated with that NIC's address is examined to determine the external listen address to embed in SIP headers.

If your system uses a multihomed load balancer having two public addresses, you must also define a pair of channels to configure both public addresses. If the engine tier server has only one NIC, you must define a second, logical address on the NIC to configure a dedicated channel for the second public address. In addition, you must configure your IP routing policies to define which logical address is associated with each public load balancer address.

Enabling DNS Support

WebLogic SIP Server supports DNS for resolving the transport, IP address and port number of a proxy required to send a SIP message. This matches the behavior described in RFC 3263. DNS may also used when routing responses in order to resolve the IP address and port number of a destination.

WARNING: Because DNS resolution is performed within the context of SIP message processing, any DNS performance problems result in increased latency performance. BEA recommends using a caching DNS server in a production environment to minimize potential performance problems.

To configure DNS support:

1. Log in to the Administration Console for the WebLogic SIP Server domain you want to configure.
2. Click Lock & Edit to obtain a configuration lock.
3. Select the SipServer node in the left pane of the Console.
4. Select the Configuration->General tab in the right pane.
5. Select **Enable DNS Server Lookup**.
6. Click Save to save your changes.
7. Click Activate Changes to apply the configuration.

When you enable DNS lookup, the server can use DNS to:

- Discover a proxy server's transport, IP address, and port number when a request is sent to a SIP URI.
- Resolve an IP address and/or port number during response routing, depending on the contents of the Sent-by field.

For proxy discovery, WebLogic SIP Server uses DNS resolution only once per SIP transaction to determine transport, IP, and port number information. All retransmissions, ACKs, or CANCEL requests are delivered to the same address and port using the same transport. For details about how DNS resolution takes place, see RFC 3263: Session Initiation Protocol (SIP): Locating SIP Servers.

When a proxy needs to send a response message, WebLogic SIP Server uses DNS lookup to determine the IP address and/or port number of the destination, depending on the information provided in the sent-by field and via header.

Configuring Network Channels for SIP or SIPS

When you create a new domain using the Configuration Wizard, WebLogic SIP Server instances are configured with a default network channel supporting the SIP protocol over UDP and TCP. This default channel is configured to use Listen Port 5060, but specifies no Listen Address. Follow the instructions in Reconfiguring an Existing Channel to change the default channel's listen address or listen port settings. See "Creating a New SIP or SIPS Channel" on page 1-6 for to create a new channel resource to support additional protocols or additional network interfaces.

Reconfiguring an Existing Channel

Note: You cannot change the protocol supported by an existing channel. To reconfigure an existing listen address/port combination to use a different network protocol, you must delete the existing channel and create a new channel using the instructions in "Creating a New SIP or SIPS Channel" on page 1-6.

1. Log in to the Administration Console for the WebLogic SIP Server domain you want to configure.
2. Click Lock & Edit to obtain a configuration lock.
3. In the left pane, select the Environment->Servers tab.
4. In the right pane, select the name of the server you want to configure.
5. Select the Protocols->Channels tab to display the configured channels.
6. To delete an existing channel, select it in the table and click Delete.
7. To reconfigure an existing channel:
   a. Select the channel's name from the channel list (for example, the default `sip` channel).

b. Edit the Listen Address or Listen Port fields to correspond to the address of a NIC or logical address on the associated engine tier machine.

c. Edit the External Listen Address or External Listen Port fields to match the public address of a load balancer in the system.

d. Edit the advanced channel attributes as necessary (see "Creating a New SIP or SIPS Channel" on page 1-6 for details.)

e. Click Save.

8. Click Activate Changes to apply the configuration.

Creating a New SIP or SIPS Channel

To create add a new SIP or SIPS channel to the configuration of a WebLogic SIP Server instance:

1. Log in to the Administration Console for the WebLogic SIP Server domain you want to configure.
2. Click Lock & Edit to obtain a configuration lock.
3. In the left pane, select the Environment->Servers tab.
4. In the right pane, select the name of the server you want to configure.
5. Select the Protocols->Channels tab to display the configured channels.
6. Click New to configure a new channel.
7. Fill in the new channel fields as follows:
   - **Name:** Enter an administrative name for this channel, such as "SIPS-Channel-eth0."
   - **Protocol:** Select either SIP to support UDP and TCP transport, or SIPS to support TLS transport. Note that a SIP channel cannot support only UDP or only TCP transport on the configured port.
8. Click Next
9. Fill in the new channel's addressing fields as follows:
   - **Listen Address:** Enter the IP address or DNS name for this channel. On a multi-homed machine, enter the exact IP address of the interface you want to configure, or a DNS name that maps to the exact IP address.

- **Listen Port:** Enter the port number used to communication via this channel. The combination of Listen Address and Listen Port must be unique across all channels configured for the server. SIP channels support both UDP and TCP transport on the configured port.
- **External Listen Address** and **External Listen Port**: Edit these fields to match the public address of a load balancer associated with this channel. When the server selects an interface or logical address to use for outbound network traffic, WebLogic SIP Server examines the channel that was configured with the same primary **Listen Address**; if the **External Listen Address** of this channel differs, the external address is embedded into SIP message headers for further call traffic. See "Configuring Load Balancer Addresses" on page 1-3.

10. Click Next
11. Optionally click Show to display and edit advanced channel properties, such as connection timeout values. Keep in mind the following restrictions and suggestions for advanced channel properties:
    - **Outbound Enabled**—This attribute cannot be unchecked, because all SIP and SIPS channels can originate network connections.
    - **HTTP Enabled for This Protocol**—This attribute cannot be selected for SIP and SIPS channels, because WebLogic SIP Server does not support HTTP transport SIP protocols.
    - **Maximum Message Size**—This attribute specifies the maximum TCP message size that the server allows on a connection from this channel. WebLogic SIP Server shuts off any connection where the messages size exceeds the configured value. The default size of 10,000,000 bytes is large. If you are concerned about preventing Denial Of Service (DOS) attacks against the server, reduce this attribute to a value that is compatible with your deployed services.
12. Click Next.
13. Click Activate Changes to apply the configuration.

Configuring Custom Timeout, MTU, and Other Properties

SIP channels can be further configured using one or more custom channel properties. The custom properties cannot be set using the Administration Console. Instead, you must use a text editor to add the properties to a single, `custom-property` stanza in the channel configuration portion of the `config.xml` file for the domain.

WebLogic SIP Server provides the following custom properties that affect the transport protocol of SIP channels:

- `TcpConnectTimeoutMillis`—Specifies the amount of time WebLogic SIP Server waits before it declares a destination address (for an outbound TCP connection) as unreachable. The property is applicable only to SIP channels; WebLogic SIP Server ignores this attribute value for SIPS channels. A value of 0 disables the timeout completely. A default value of 3000 milliseconds is used if you do not specify the custom property.
- `SctpConnectTimeoutMillis`—Specifies the amount of time WebLogic SIP Server waits before it declares a destination address (for an outbound STCP connection) as unreachable. The property is applicable only to SCTP channels (for Diameter traffic). A value of 0 disables the timeout completely. A default value of 3000 milliseconds is used if you do not specify the custom property. See "Static Port Configuration for Outbound UDP Packets" on page 2-4 for information about creating SCTP channels for Diameter.
- `SourcePorts`—Configures one or more static port numbers that a server uses for originating UDP packets.

  WARNING: BEA does not recommend using the `SourcePorts` custom property in most configurations because it degrades performance. Configure the property only in cases where you must specify the exact ports that WebLogic SIP Server uses to originate UDP packets. See for information about using this property.
- `Mtu`—Specifies the Maximum Transmission Unit (MTU) value for this channel. A value of -1 uses the default MTU size for the transport.
- `EnabledProtocolVersions`—Specifies the version of the SSL protocol to use with this channel when WebLogic SIP Server acts as an SSL client. When acting as an SSL client, by default the channel requires TLS V1.0 as the supported protocol. The server can be configured to use SSL V3.0 as well, if that is the highest version that the SSL peer servers support. You can set one of the following values for this property:
  - `TLS1`, the default, configures the channel to send and accept only TLS V1.0 messages. Peers must respond with a TLS V1.0 message, or the SSL connection is dropped.
  - `SSL3` configures the channel to send and accept only SSL V3.0 messages. Peers must respond with an SSL V3.0 message, or the SSL connection is dropped.
  - `ALL` supports either TLS V1.0 or SSL V3.0 messages. Peers must respond with a TLS V1.0 or SSL V3.0 message, or the SSL connection is dropped.

To configure a custom property, use a text editor to modify the `config.xml` file directly, or use a JMX client such as WLST to add the custom property. When editing `config.xml` directly, ensure that you add only one `custom-properties` element to the end of a channel's

configuration stanza. Separate multiple custom properties within the same element using semicolons (;) as shown in Listing 1-1.

Listing 1-1 Setting Custom Properties

```
<network-access-point>
  <name>sip</name>
  <protocol>sip</protocol>
  <listen-port>5060</listen-port>
  <public-port>5060</public-port>
  <http-enabled-for-this-protocol>false</http-enabled-for-this-protocol>
  <tunneling-enabled>false</tunneling-enabled>
  <outbound-enabled>true</outbound-enabled>
  <enabled>true</enabled>
  <two-way-ssl-enabled>false</two-way-ssl-enabled>
  <client-certificate-enforced>false</client-certificate-enforced>
  <custom-properties>EnabledProtocolVersions=ALL;Mtu=1000;SourcePorts=50
60</custom-properties>
</network-access-point>
```

Configuring SIP Channels for Multi-Homed Machines

If you are configuring a server that has multiple network interfaces (a “multihomed” server), you must configure a separate network channel for each IP address used by WebLogic SIP Server. WebLogic SIP Server uses the listen address and listen port values for each channel when embedding routing information into SIP message system headers.

Note: If you do not configure a channel for a particular IP address on a multihomed machine, that IP address cannot be used when populating Via, Contact, and Record-Route headers.

Configuring TCP and TLS Channels for Diameter Support

WebLogic SIP Server's Diameter implementation supports the Diameter protocol over the TCP or TLS transport protocols. To enable incoming Diameter connections on a server, you configure a dedicated network channel using the protocol type "diameter" for TCP transport, or "diameters" for both TCP and TLS transport. The Diameter implementation application may automatically upgrade Diameter connections to use TLS as described in the Diameter specification (RFC 3558).

See "Configuring Diameter Client Nodes and Relay Agents" on page 4-1 for more information about configuring network channels for Diameter protocol support.

Configuring Engine Servers to Listen on Any IP Interface (0.0.0.0 or ::)

To configure WebLogic SIP Server to listen for UDP traffic on any available IP interface, create a new SIP channel and specify **`0.0.0.0`** (or **`::`** for IPv6 networks) as the listen address. Note that you must still configure at least one additional channel with an explicit IP address to use for outgoing SIP messages. (For multi-homed machines, each interface used for outgoing messages must have a configured channel.)

Note: If you configure a SIP channel without specifying the channel listen address, but you do configure a listen address for the server itself, then the SIP channel inherits the server listen address. In this case the SIP channel *does not* listen on IP_ANY.

Note: Using the `0.0.0.0` configuration affects only UDP traffic on Linux platforms. WebLogic SIP Server only creates TCP and HTTP listen threads corresponding to the configured hostname of the server, and localhost. If multiple addresses are mapped to the hostname, WebLogic SIP Server displays warning messages upon startup. To avoid this problem and listen on all addresses, specify the `::` address, which encompasses all available addresses for both IPv6 and IPv4 for HTTP and TCP traffic as well.

Configuring Unique Listen Address Attributes for Data Tier Replicas

Each replica in the data tier must bind to a unique Listen Address attribute (a unique DNS name or IP address) in order to contact one another as peers. Follow these instructions for each replica to assign a unique Listen Address:

1. Access the Administration Console for the WebLogic SIP Server domain.

2. In the left pane, select the name of the server to configure.
3. Select the Configuration->General tab.
4. Enter a unique DNS name or IP address in the Listen Address field.
5. Click Apply.

CHAPTER 2

Production Network Architectures and WebLogic SIP Server Configuration

The following sections describe common network architectures used in production deployments, and explain how WebLogic SIP Server is configured to run in those architectures:

Overview

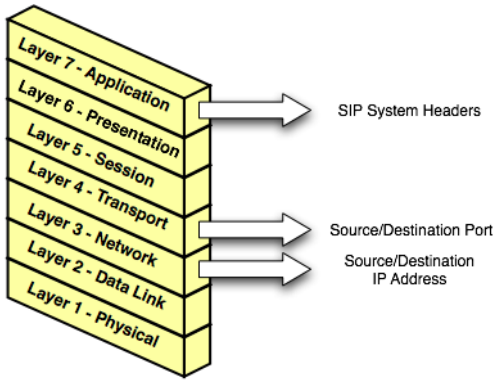
Most production installations of WebLogic SIP Server are contain one or more of the following characteristics:

- Multiple engine tier servers arranged in a cluster.
- Multiple network channels per engine tier server instance, in support of multiple SIP transport protocols or multiple Network Interface Cards (NICs) on multihomed hardware.
- One or more load balancers, or a multihomed load balancer, performing server failover and possibly Network Address Translation (NAT) for source or destination network packets.

A combination of these network elements can make it difficult to understand how elements interact with one another, and how a particular combination of elements or configuration options affects the contents of a SIP message or transport protocol datagram.

The sections that follow attempt to describe common WebLogic SIP Server network architectures and explain how servers are configured in each architecture. The sections also explain how information in SIP messages and transport datagrams is affected by each configuration. Figure 2-1 shows the typical Open Systems Interconnect (OSI) model layers that can be affected by different network configurations.

Figure 2-1 OSI Layers Affected by WebLogic SIP Server Network Configuration



Layer 3 (Network) and Layer 4 (Transport) contain the source or destination IP address and port numbers for both outgoing and incoming transport datagrams. Layer 7 (Application) may also be affected because the SIP protocol specifies that certain SIP headers include addressing information for contacting the sender of a SIP message.

Single-NIC Configurations with TCP and UDP Channels

In a simple network configuration for a server having a single NIC, one or more network channels may be created to support SIP messages over UDP and TCP, or SIPS over TLS. It is helpful to understand how this simple configuration affects information in the OSI model, so that you can understand how more complex configurations involving multihomed hardware and load balancers affect the same information.

Figure 2-2 Single-NIC Network Channel Configuration

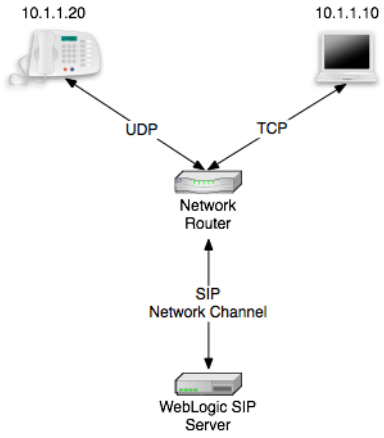


Figure 2-2 shows a single engine tier server instance with a single NIC. The server is configured with one network channel supporting SIP over UDP and TCP. (SIP channels always support both UDP and TCP transports; you cannot support only one of the two.) Figure 2-2 also shows two clients communicating with the server, one over UDP and one over TCP.

For the TCP transport, the outgoing datagram (delivered from WebLogic SIP Server to the UA) contains the following information:

- Layer 3 includes the source IP address specified by the network channel (10.1.1.10 in the example above).
- Layer 4 includes the source port number allocated by the underlying operating system.

Incoming TCP datagrams (delivered from the UA to WebLogic SIP Server) contain the following information:

- Layer 3 includes the destination IP address specified by the network channel (10.1.1.10).
- Layer 4 contains the destination port number specified by the network channel (5060).

For outgoing UDP datagrams, the OSI layer information contains the same information as with TCP transport. For incoming UDP datagrams, the OSI layer information is the same as TCP except in the case of incoming datagram Layer 4 information. For incoming UDP datagrams, Layer 4 contains either:

- The destination port number specified by the network channel (5060), or
- The ephemeral port number previously allocated by WebLogic SIP Server.

By default WebLogic SIP Server allocates ports from the ephemeral port number range of the underlying operating system for outgoing UDP datagrams. WebLogic SIP Server allows external connections to use an ephemeral point as the destination port number, in addition to the port number configured in the network channel. In other words, WebLogic SIP Server automatically listens on all ephemeral ports that the server allocates. You can optionally disable WebLogic SIP Server's use of ephemeral port numbers by specifying the following option when starting the server:

```
-Dwlss.udp.listen.on.ephemeral=false
```

You can determine WebLogic SIP Server's use of a particular ephemeral port by examining the server log file:

```
<Nov 30, 2005 12:00:00 AM PDT> <Notice> <WebLogicServer> <BEA-000202>
<Thread "SIP Message Processor (Transport UDP)" listening on port 35993.>
```

Static Port Configuration for Outbound UDP Packets

WebLogic SIP Server network channels provide a `SourcePorts` attribute that you can use to configure one or more static ports that a server uses for originating UDP packets.

WARNING: BEA does not recommend using the `SourcePorts` custom property in most configurations because it degrades performance. Configure the property only in cases where you must specify the exact ports that WebLogic SIP Server uses to originate UDP packets.

To configure the `SourcePorts` property, use a JMX client such as WLST or directly modify a network channel configuration in `config.xml` to include the custom property. `SourcePorts` defines an array of port numbers or port number ranges. Do not include spaces in the `SourcePorts` definition - use only port numbers, hyphens ("-") to designate ranges of ports, and commas (",") to separate ranges or individual ports. See Listing 2-1 for an example configuration.

Listing 2-1 Static Port Configuration for Outgoing UDP Packets

```
<network-access-point>
  <name>sip</name>
  <protocol>sip</protocol>
  <listen-port>5060</listen-port>
  <public-port>5060</public-port>
  <http-enabled-for-this-protocol>false</http-enabled-for-this-protocol>
  <tunneling-enabled>false</tunneling-enabled>
  <outbound-enabled>true</outbound-enabled>
  <enabled>true</enabled>
  <two-way-ssl-enabled>false</two-way-ssl-enabled>
  <client-certificate-enforced>false</client-certificate-enforced>
  <custom-properties>SourcePorts=5060</custom-properties>
</network-access-point>
```

Multihomed Server Configurations Overview

Engine tier servers in a production deployment frequently utilize multihomed server hardware, having two or more NICs. Multihomed hardware is typically used for one of the following purposes:

- To provide redundant network connections within the same subnet. Having multiple NICs ensures that one or more network connections are available to communicate with data tier servers or the Administration Server, even if a single NIC fails.
- To support SIP communication across two or more different subnets. For example WebLogic SIP Server may be configured to proxy SIP requests from UAs in one subnet to UAs in a second subnet, when the UAs cannot directly communicate across subnets.

The configuration requirements and OSI layer information differ depending on the use of multihomed hardware in your system. When multiple NICs are used to provide redundant connections within a subnet, servers are generally configured to listen on all available addresses (IP_ANY) as described in "Multihomed Servers Listening On All Addresses (IP_ANY)" on page 2-6.

When using multiple NICs to support different subnets, you must configure multiple network on the server for each different NIC as described in "Multihomed Servers Listening on Multiple Subnets" on page 2-6.

Multihomed Servers Listening On All Addresses (IP_ANY)

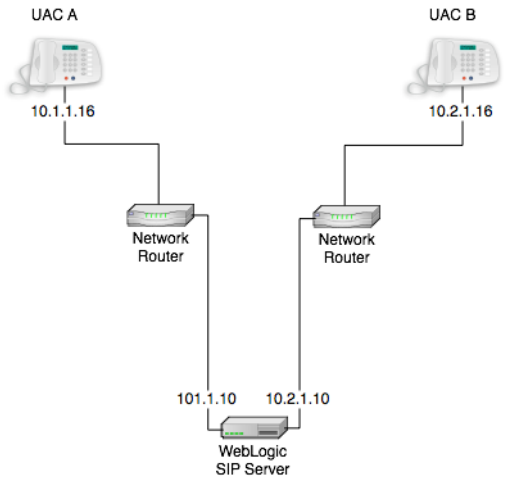
The simplest multihome configuration enables a WebLogic SIP Server instance to listen on all available NICs (physical NICs as well as logical NICs), sometimes described as IP_ANY. To accomplish this, you simply configure a single network channel and specify a channel listen address of 0.0.0.0.

Note that you must configure the 0.0.0.0 address directly on the server's network channel. If you specify no IP address in the channel, the channel inherits the listen address configured for the server instance itself. See "Configuring Engine Servers to Listen on Any IP Interface (0.0.0.0 or ::)" on page 1-10.

Multihomed Servers Listening on Multiple Subnets

Multiple NICs can also be used in engine tier servers to listen on multiple subnets. The most common configuration uses WebLogic SIP Server to proxy SIP traffic from one subnet to another where no direct access between subnets is permitted. Figure 2-3 shows this configuration.

Figure 2-3 Multihomed Configuration for Proxying between Subnets



To configure the WebLogic SIP Server instance in Figure 2-3 you must define a separate network channel for each NIC used on the server machine. Listing 2-2 shows the `config.xml` entries that define channels for the sample configuration.

Listing 2-2 Sample Network Channel Configuration for NICs on Multiple Subnets

```
<NetworkAccessPoint ListenAddress=”10.1.1.10” ListenPort="5060"
Name="sipchannelA" Protocol="sip"/>

<NetworkAccessPoint ListenAddress=”10.2.1.10” ListenPort="5060"
Name="sipchannelB" Protocol="sip"/>
```

Understanding the Route Resolver

When WebLogic SIP Server is configured to listen on multiple subsets, a feature called the *route resolver* is responsible for the following activities:

- Populating OSI Layer 7 information (SIP system headers such as Via, Contact, and so forth) with the correct address.

- Populating OSI Layer 3 information with the correct source IP address.

For example, in the configuration shown in Figure 2-3, WebLogic SIP Server must add the correct subnet address to SIP system headers and transport datagrams in order for each UA to continue processing SIP transactions. If the wrong subnet is used, replies cannot be delivered because neither UA can directly access the other UA's subnet.

The route resolver works by determining which NIC the operating system will use to send a datagram to a given destination, and then examining the network channel that is associated with that NIC. It them uses the address configured in the selected network channel to populate SIP headers and Layer 3 address information.

For example, in the configuration shown in Figure 2-3, an INVITE message sent from WebLogic SIP Server to UAC B would have a destination address of 10.2.1.16. The operating system would transmit this message using NIC B, which is configured for the corresponding subnet. The route resolver associates NIC B with the configured `sipchannelB` and embeds the channel's IP address (10.2.1.10) in the VIA header of the SIP message. UAC B then uses the VIA header to direct subsequent messages to the server using the correct IP address. A similar process is used for UAC A, to ensure that messages are delivered only on the correct subnet.

IP Aliasing with Multihomed Hardware

IP aliasing assigns multiple logical IP addresses to a single NIC, and is configured in the underlying server operating system. If you configure IP aliasing and all logical IP addresses are within the same subnet, you can simply configure WebLogic SIP Server to listen on all addresses as described in "Multihomed Servers Listening On All Addresses (IP_ANY)" on page 2-6.

If you configure IP aliasing to create multiple logical IP addresses on different subnets, you must configure a separate network channel for each logical IP address. In this configuration, WebLogic SIP Server treats all logical addresses as separate physical interfaces (NICs) and uses the route resolver to populate OSI Layer 4 and Layer 7 information based on the configured channel.

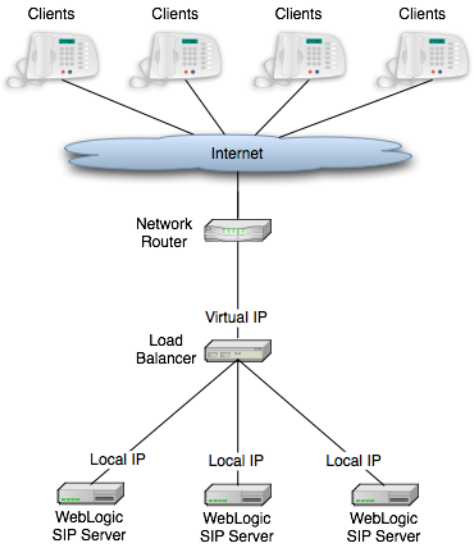
Load Balancer Configurations

In addition to providing failover capabilities and distributing the client load across multiple servers, a load balancer is also an important tool for configuring the network information transmitted between clients and servers. The sections that follow describe common load balancer configurations used with WebLogic SIP Server.

Single Load Balancer Configuration

The most common load balancer configuration utilizes a single load balancer that gates access to a cluster of engine tier servers, as shown in Figure 2-4.

Figure 2-4 SIngle Load Balancer Configuration



To configure WebLogic SIP Server for use with a single load balancer as in Figure 2-4, configure one or more network channels for each server, and configure the public address of each channel with the Virtual IP address of the load balancer. In this configuration, WebLogic SIP Server embeds the load balancer IP address in SIP message system headers to ensure that clients can reach the cluster for subsequent replies. "Managing WebLogic SIP Server Network Resources" on page 1-1 presents detailed steps for configuring network channels with load balancer addresses.

Note: Although some load balancing switches can automatically re-route all SIP messages in a given call to the same engine tier server, this functionality is not required with WebLogic SIP Server installations. See Alternate Configurations in *Configuring WebLogic SIP Server* for more information.

Multiple Load Balancers and Multihomed Load Balancers

Multiple load balancers (or a multihomed load balancer) can be configured to provide several virtual IP addresses for a single WebLogic SIP Server cluster. To configure WebLogic SIP Server for use with a multihomed load balancer, you create a dedicated network channel for each load balancer or local server NIC, and set the channel's public address to the virtual IP address of the appropriate load balancer. In this configuration, the route resolver associates a configured channel with the NIC used for originating SIP messages. The public address of the selected channel is then used for populating SIP system messages. See "Understanding the Route Resolver" on page 2-7.

Network Address Translation Options

In the most common case, a load balancer is configured using destination NAT to provide a public IP address that clients use for communicating with one or more internal (private) WebLogic SIP Server addresses. Load balancers may also be configured using source NAT, which modifies the Layer 3 address information originating from a private address to match the virtual IP address of the load balancer itself.

With the default route resolver behavior, a WebLogic SIP Server engine originates UDP packets having a source IP address that matches the address of a local NIC (the private address). This can be problematic for applications that try to respond directly to the Layer 3 address embedded in the transport packet, because the local server address may not be publicly accessible. If your applications exhibit this problem, BEA recommends that you configure the load balancer to perform source NAT to change the transport Layer 3 address to a publicly-accessible virtual IP address.

IP Masquerading Alternative to Source NAT

WARNING: Using the WebLogic SIP Server IP masquerading functionality can lead to network instability, because it requires duplicate IP addresses on multiple servers. Production deployments must use a load balancer configured for source NAT, rather than IP masquerading, to ensure reliable network performance.

If you choose not to enable source NAT on your load balancer, WebLogic SIP Server provides limited IP masquerading functionality. To use this functionality, configure a logical address on each engine tier server using the public IP address of the load balancer for the cluster. (Duplicate the same logical IP address on each engine tier server machine). When a local server interface matches the IP address of a configured load balancer (defined in the public address of a network

channel), WebLogic SIP Server uses that interface to originate SIP UDP messages, and the Layer 3 address contains a public address.

You can disable WebLogic SIP Server's IP masquerading functionality by using the startup option:

```
-Dwlss.udp.lb.masquerade=false
```

CHAPTER 3

Example WebLogic SIP Server Network Configuration

The following sections describe a sample network configuration for WebLogic SIP Server using a non-SIP-aware load balancer:

Overview

WebLogic SIP Server is compatible with load balancers that are not SIP-aware, meaning that they do not consider existing SIP dialogues when routing requests to servers. This document demonstrates load balancer and WebLogic SIP Server configuration, as well as SIP and Network Address Translation (NAT) interactions in various configurations.

For more information about implementation-dependent issues surrounding NAT see the IETF document, *NAT Behavioral Requirements for Unicast UDP*.

Example Network Topology

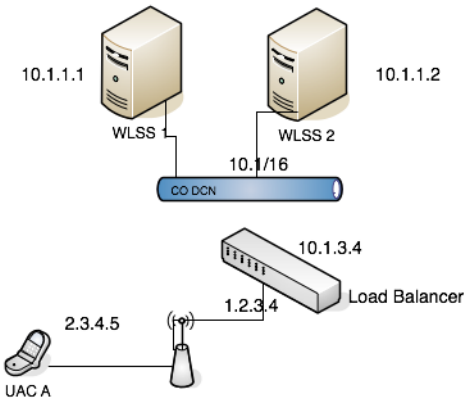
Figure 3-1 shows the sample network topology described in this section. A WebLogic SIP Server cluster, consisting of engines WLSS 1 and WLSS 2, is configured on private IP network 10.1/16

(an internal 16-bit subnet). The cluster's public IP address is 1.2.3.4, which is the virtual IP address configured on the load balancer.

The User Agent, UAC A, with IP address 2.3.4.5 never sees the internal IP addresses configured for the WebLogic SIP Server cluster. Instead, it sends requests to, and receives responses from 1.2.3.4.

The sections that follow discuss configuring the WebLogic SIP Server cluster and load balancer for this example system.

Figure 3-1 Example Network Topology



WebLogic SIP Server Configuration

The WebLogic SIP Server cluster configuration specifies the public address as 1.2.3.4, and the public port as 5060 (see "Configuring Load Balancer Addresses" on page 1-3) for each engine. The default route on both WebLogic SIP Server engines points to the load balancer's 10.1/16 network interface: 10.1.3.4. The WebLogic SIP Server (servers WLSS 1 and WLSS 2) routing table is shown in Listing 3-1.

Listing 3-1 WebLogic SIP Server Routing Table

```
$ /sbin/route
```

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.0.0          *             255.255.0.0     U     0      0        0 eth0
default         10.1.3.4         0.0.0.0         UG    0      0
```

Load Balancer Configuration

The load balancer is configured with a virtual IP address of 1.2.3.4, and two real servers, WLSS 1 and WLSS 2, having addresses 10.1.1.1 and 10.1.1.2, respectively. The load balancer also has an internal IP address of 10.1.3.4 configured on the 10.1/16 network. The UAC address, 2.3.4.5, is reachable from the load balancer by static route configuration on the load balancer. The load balancer routing table is shown in Listing 3-2.

Listing 3-2 Load balancer Routing Table

```
$ /sbin/route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.0.0          *             255.255.0.0     U     0      0        0 eth1
1.2.0.0           *              255.255.0.0     U     0      0
```

Because the SIP protocol specification (RFC 3261) dictates the destination IP address and UDP port numbers that user agents must use when sending requests or responses, the NAT configuration of the load balancer must be done in a way that does not violate RFC 3261 requirements. Three setup options can be used to accomplish this configuration:

- NAT-based configuration
- maddr-Based Configuration
- rport-Based Configuration

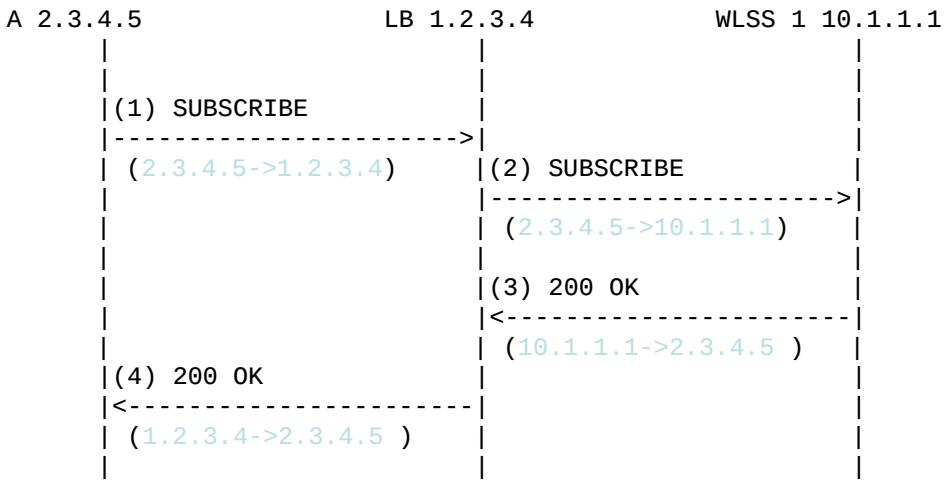
The sections that follow describe each approach.

NAT-based configuration

The default UDP NAT behavior for load balancers is to perform destination IP address translation in the public->private network direction, and source IP address translation in the private->public network direction. This means setting up destination address translation in the UAC->WebLogic SIP Server (2.3.4.5->1.2.3.4) direction without source address translation, and source address translation in the WebLogic SIP Server->UAC (10.1/16->2.3.4.5) direction without destination address translation.

Figure 3-2 illustrates the UDP packet flow for a SUBSCRIBE/200OK transaction.

Figure 3-2 SUBSCRIBE Sequence



Note that the source and destination IP addresses of the UDP packets are shown in blue. In the UAC->WebLogic SIP Server direction, the load balancer translates the destination IP address but not the source IP address. In the WebLogic SIP Server->UAC direction, the load balancer translates the source IP address but not the destination IP address.

The complete message trace (including IP and UDP headers, as well as the SIP payload) for the sequence from Figure 3-2 is shown in Listing 3-3 below.

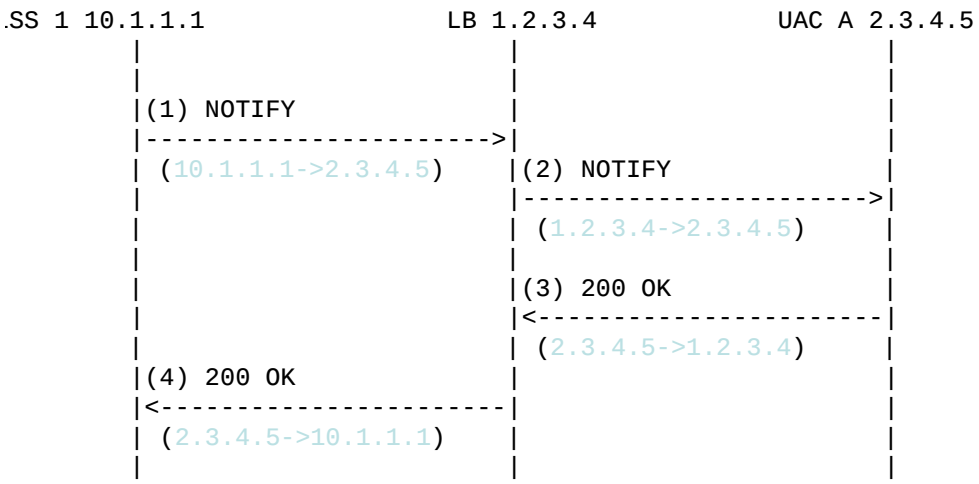
Listing 3-3 Complete SUBSCRIBE Message Trace

```
No.     Time       Source                Destination           Protocol Info
      1 1.425250    2.3.4.5           1.2.3.4          SIP      Request:
SUBSCRIBE sip:subscribe@1.2.3.4:5060

Internet Protocol, Src: 2.3.4.5 (2.3.4.5), Dst: 1.2.3.4 (1.2.3.4)
User Datagram Protocol, Src Port: 9999 (9999), Dst Port: sip (5060)
Session Initiation Protocol
    Request-Line: SUBSCRIBE sip:subscribe@1.2.3.4:5060 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 2.3.4.5:9999;branch=1
        From: sipp <sip:sipp@2.3.4.5>;tag=1
        To: sut <sip:subscribe@1.2.3.4:5060>
        Call-ID: 1-25923@2.3.4.5
        Cseq: 1 SUBSCRIBE
        Contact: sip:sipp@2.3.4.5:9999
        Max-Forwards: 70
        Event: ua-profile
        Expires: 10
        Content-Length: 0

No.     Time       Source                Destination           Protocol Info
      2 2.426250    2.3.4.5           10.1.1.1          SIP      Request:
SUBSCRIBE sip:subscribe@1.2.3.4:5060

Internet Protocol, Src: 2.3.4.5 (2.3.4.5), Dst: 10.1.1.1 (10.1.1.1)
User Datagram Protocol, Src Port: 9999 (9999), Dst Port: sip (5060)
```

```
Session Initiation Protocol
    Request-Line: SUBSCRIBE sip:subscribe@1.2.3.4:5060 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 2.3.4.5:9999;branch=1
        From: sipp <sip:sipp@2.3.4.5>;tag=1
        To: sut <sip:subscribe@1.2.3.4:5060>
        Call-ID: 1-25923@2.3.4.5
        Cseq: 1 SUBSCRIBE
        Contact: sip:sipp@2.3.4.5:9999
        Max-Forwards: 70
        Event: ua-profile
        Expires: 10
        Content-Length: 0

No.     Time        Source                Destination          Protocol Info
     3 3.430903   10.1.1.1           2.3.4.5         SIP     Status: 200 OK

Internet Protocol, Src: 10.1.1.1 (10.1.1.1), Dst: 2.3.4.5 (2.3.4.5)
User Datagram Protocol, Src Port: 42316 (42316), Dst Port: 9999 (9999)
Session Initiation Protocol
    Status-Line: SIP/2.0 200 OK
    Message Header
        To: sut <sip:subscribe@1.2.3.4:5060>;tag=82722c03
        Content-Length: 0
        Contact:
<sip:app-12eomtm5h5f77@1.2.3.4:5060;transport=udp;wlsscid=1ae4479ac6ff71>
        CSeq: 1 SUBSCRIBE
        Call-ID: 1-25923@2.3.4.5
```

If WebLogic SIP Server subsequently sends a NOTIFY request to the UAC, the sequence shown in Figure 3-3 takes place:

Figure 3-3 NOTIFY Sequence



As in the previous sequence, the IP address translation takes place in the WebLogic SIP Server->UAC direction for the source IP address, and UAC->WebLogic SIP Server direction for the destination IP address.

Note that this setup does not require the load balancer to maintain session state information or to be SIP-aware. The complete message trace from Figure 3-3 is shown in Listing 3-4 below.

Listing 3-4 Complete NOTIFY Message Trace

```
No.     Time        Source                Destination           Protocol Info
     1 5.430952    10.1.1.1              2.3.4.5          SIP     Request:
NOTIFY sip:sipp@2.3.4.5:9999


Internet Protocol, Src: 10.1.1.1 (10.1.1.1), Dst: 2.3.4.5 (2.3.4.5)
User Datagram Protocol, Src Port: 42316 (42316), Dst Port: 9999 (9999)
Session Initiation Protocol
```

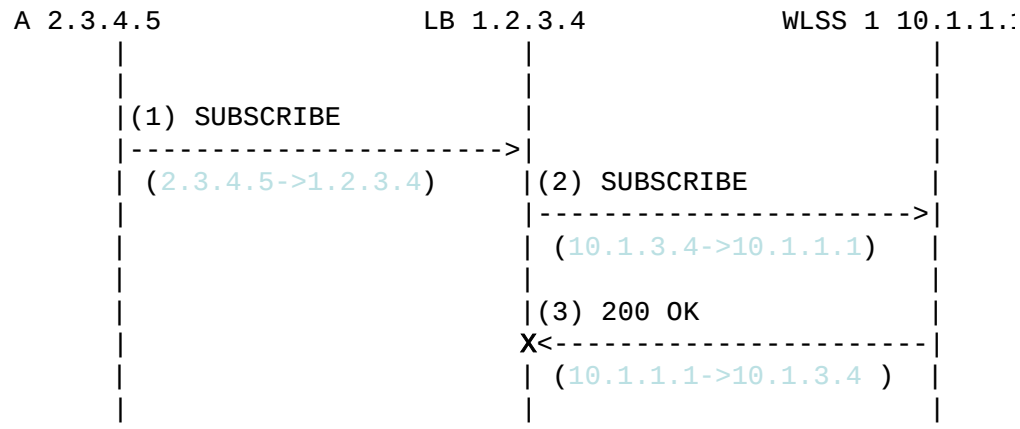
```
    Request-Line: NOTIFY sip:sipp@2.3.4.5:9999 SIP/2.0
    Message Header
        To: sipp <sip:sipp@2.3.4.5>;tag=1
        Content-Length: 0
        Contact:
<sip:app-12eomtm5h5f77@1.2.3.4:5060;transport=udp;wlsscid=1ae4479ac6ff71>
        CSeq: 1 NOTIFY
        Call-ID: 1-25923@2.3.4.5
        Via: SIP/2.0/UDP
1.2.3.4:5060;wlsscid=1ae4479ac6ff71;branch=z9hG4bKc5e4c3b4c22be517133ab749
adeece4e
        From: sut <sip:subscribe@1.2.3.4:5060>;tag=82722c03
        Max-Forwards: 70

No.     Time       Source               Destination          Protocol Info
      2 6.430952    1.2.3.4          2.3.4.5          SIP      Request:
NOTIFY sip:sipp@2.3.4.5:9999

Internet Protocol, Src: 1.2.3.4 (1.2.3.4), Dst: 2.3.4.5 (2.3.4.5)
User Datagram Protocol, Src Port: 2222 (2222), Dst Port: 9999 (9999)
Session Initiation Protocol
    Request-Line: NOTIFY sip:sipp@2.3.4.5:9999 SIP/2.0
    Message Header
        To: sipp <sip:sipp@2.3.4.5>;tag=1
        Content-Length: 0
        Contact:
<sip:app-12eomtm5h5f77@1.2.3.4:5060;transport=udp;wlsscid=1ae4479ac6ff71>
        CSeq: 1 NOTIFY
        Call-ID: 1-25923@2.3.4.5
```

```
        Via: SIP/2.0/UDP
1.2.3.4:5060;wlsscid=1ae4479ac6ff71;branch=z9hG4bKc5e4c3b4c22be517133ab749
adeece4e
        From: sut <sip:subscribe@1.2.3.4:5060>;tag=82722c03
        Max-Forwards: 70



No.     Time        Source                Destination           Protocol Info
      3 7.431367    2.3.4.5          1.2.3.4         SIP      Status: 200 OK


Internet Protocol, Src: 2.3.4.5 (2.3.4.5), Dst: 1.2.3.4 (1.2.3.4)
User Datagram Protocol, Src Port: 9999 (9999), Dst Port: sip (5060)
Session Initiation Protocol
    Status-Line: SIP/2.0 200 OK
    Message Header
        Via: SIP/2.0/UDP
1.2.3.4:5060;wlsscid=1ae4479ac6ff71;branch=z9hG4bKc5e4c3b4c22be517133ab749
adeece4e
        From: sut <sip:subscribe@1.2.3.4:5060>;tag=82722c03
        To: sipp <sip:sipp@2.3.4.5>;tag=1;tag=1
        Call-ID: 1-25923@2.3.4.5
        CSeq: 1 NOTIFY
        Contact: <sip:2.3.4.5:9999;transport=UDP>
```

WARNING: If NAT is performed on both the source (SNAT) and destination IP addresses, the configuration does not work because the load balancer usually relies on a specific destination port number value to be sent in responses to requests. That port number value is dictated by RFC 3261, and must come from the Via header, which presents a conflict with load balancer's NAT requirements. RFC 3261 requires that responses to SIP requests be sent to the IP address used to send the request (unless maddr is present in the Via, as described in “maddr-Based

Configuration" on page 3-12). Consequently, in Figure 3-4 below, Step 3, WebLogic SIP Server sends a 200 OK response to the load balancer internal IP address (10.1.3.4) and port 5060. That response is then dropped.

Figure 3-4 Source and Destination NAT



The complete message trace from Figure 3-4 is show in Listing 3-5 below.

Listing 3-5 Complete Failing SUBSCRIBE Message Trace

```
No.     Time       Source                Destination          Protocol Info
      1 1.425250    2.3.4.5           1.2.3.4          SIP      Request:
SUBSCRIBE sip:subscribe@1.2.3.4:5060


Internet Protocol, Src: 2.3.4.5 (2.3.4.5), Dst: 1.2.3.4 (1.2.3.4)
User Datagram Protocol, Src Port: 9999 (9999), Dst Port: sip (5060)
Session Initiation Protocol
    Request-Line: SUBSCRIBE sip:subscribe@1.2.3.4:5060 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 2.3.4.5:9999;branch=1
```

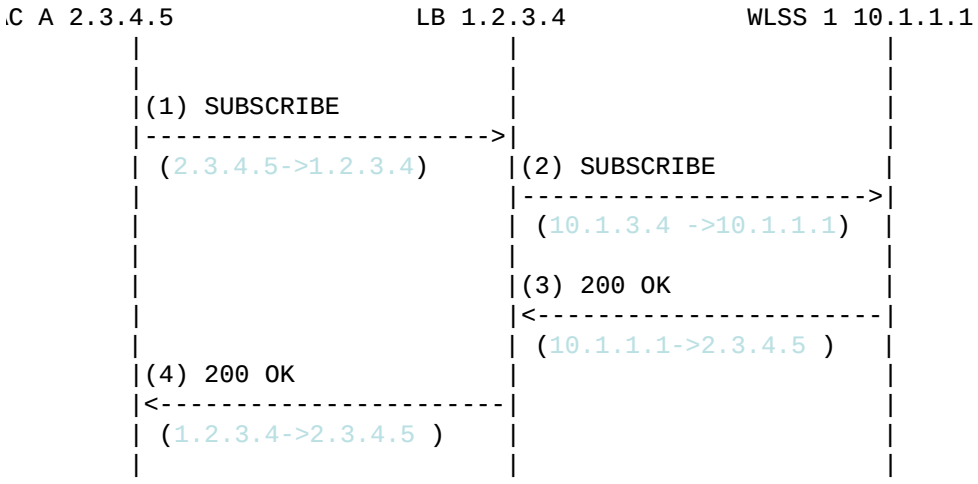
```
        From: sipp <sip:sipp@2.3.4.5>;tag=1
        To: sut <sip:subscribe@1.2.3.4:5060>
        Call-ID: 1-25923@2.3.4.5
        Cseq: 1 SUBSCRIBE
        Contact: sip:sipp@2.3.4.5:9999
        Max-Forwards: 70
        Event: ua-profile
        Expires: 10
        Content-Length: 0

No.     Time       Source               Destination          Protocol Info
      2 2.426250    10.1.3.4           10.1.1.1          SIP     Request:
SUBSCRIBE sip:subscribe@1.2.3.4:5060

Internet Protocol, Src: 10.1.3.4 (10.1.3.4), Dst: 10.1.1.1 (10.1.1.1)
User Datagram Protocol, Src Port: 2222 (2222), Dst Port: sip (5060)
Session Initiation Protocol
    Request-Line: SUBSCRIBE sip:subscribe@1.2.3.4:5060 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 2.3.4.5:9999;branch=1
        From: sipp <sip:sipp@2.3.4.5>;tag=1
        To: sut <sip:subscribe@1.2.3.4:5060>
        Call-ID: 1-25923@2.3.4.5
        Cseq: 1 SUBSCRIBE
        Contact: sip:sipp@2.3.4.5:9999
        Max-Forwards: 70
        Event: ua-profile
        Expires: 10
```

```
        Content-Length: 0

No.     Time       Source                Destination          Protocol Info
     3 3.430903   10.1.1.1           10.1.3.4        SIP     Status: 200 OK

Internet Protocol, Src: 10.1.1.1 (10.1.1.1), Dst: 10.1.3.4 (10.1.3.4)
User Datagram Protocol, Src Port: 42316 (42316), Dst Port: 9999 (9999)
Session Initiation Protocol
```

maddr-Based Configuration

When the maddr parameter is present in the Via header, the response is sent to the IP address specified in the maddr rather than to the received IP address (even when SNAT is enabled). In the example below, the UAC specifies a maddr set to 2.3.4.5 in the Via header. Consequently the response from the SIP server makes it to the UAC.

Figure 3-5 maddr Sequence



The complete message trace from Figure 3-5 is shown in Listing 3-6 below.

Listing 3-6 Complete maddr Message Trace

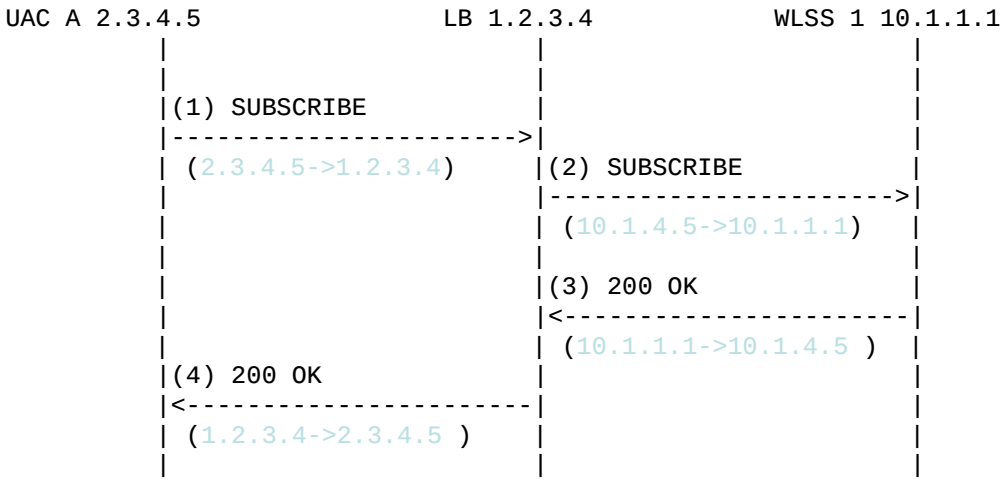
```
No.     Time       Source                Destination           Protocol Info
      1 1.425250    2.3.4.5           1.2.3.4          SIP      Request:
SUBSCRIBE sip:subscribe@1.2.3.4:5060

Internet Protocol, Src: 2.3.4.5 (2.3.4.5), Dst: 1.2.3.4 (1.2.3.4)
User Datagram Protocol, Src Port: 9999 (9999), Dst Port: sip (5060)
Session Initiation Protocol
    Request-Line: SUBSCRIBE sip:subscribe@1.2.3.4:5060 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 2.3.4.5:9999;maddr=2.3.4.5;branch=1
        From: sipp <sip:sipp@2.3.4.5>;tag=1
        To: sut <sip:subscribe@1.2.3.4:5060>
        Call-ID: 1-25923@2.3.4.5
        Cseq: 1 SUBSCRIBE
        Contact: sip:sipp@2.3.4.5:9999
        Max-Forwards: 70
        Event: ua-profile
        Expires: 10
        Content-Length: 0

No.     Time       Source                Destination           Protocol Info
      2 2.426250    10.1.3.4          10.1.1.1          SIP     Request:
SUBSCRIBE sip:subscribe@1.2.3.4:5060

Internet Protocol, Src: 10.1.3.4 (10.1.3.4), Dst: 10.1.1.1 (10.1.1.1)
User Datagram Protocol, Src Port: 2222 (2222), Dst Port: sip (5060)
```

```
Session Initiation Protocol
    Request-Line: SUBSCRIBE sip:subscribe@1.2.3.4:5060 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 2.3.4.5:9999;maddr=2.3.4.5;branch=1
        From: sipp <sip:sipp@2.3.4.5>;tag=1
        To: sut <sip:subscribe@1.2.3.4:5060>
        Call-ID: 1-25923@2.3.4.5
        Cseq: 1 SUBSCRIBE
        Contact: sip:sipp@2.3.4.5:9999
        Max-Forwards: 70
        Event: ua-profile
        Expires: 10
        Content-Length: 0

No.     Time        Source                Destination          Protocol Info
     3 3.430903   10.1.1.1           2.3.4.5         SIP     Status: 200 OK

Internet Protocol, Src: 10.1.1.1 (10.1.1.1), Dst: 2.3.4.5 (2.3.4.5)
User Datagram Protocol, Src Port: 42316 (42316), Dst Port: 9999 (9999)
Session Initiation Protocol
    Status-Line: SIP/2.0 200 OK
    Message Header
        To: sut <sip:subscribe@1.2.3.4:5060>;tag=82722c03
        Content-Length: 0
        Contact:
<sip:app-12eomtm5h5f77@1.2.3.4:5060;transport=udp;wlsscid=1ae4479ac6ff71>
```

rport-Based Configuration

RFC 3581 improves SIP and NAT interactions by allowing the client to request that the server send responses to a UDP port number from the request rather than from the Via. In order for both SUBSCRIBE and NOTIFY to work correctly, both the UAC as well as WebLogic SIP Server must support RFC 3581. Figure 3-6 illustrates the SUBSCRIBE flow.

Figure 3-6 rport SUBSCRIBE Sequence



The complete message trace from Figure 3-6 is shown in Listing 3-7 below.

Listing 3-7 Complete Message Trace for rport SUBSCRIBE

```
No.     Time       Source               Destination          Protocol Info
      1 1.425250    2.3.4.5           1.2.3.4         SIP      Request:
SUBSCRIBE sip:subscribe@1.2.3.4:5060


Internet Protocol, Src: 2.3.4.5 (2.3.4.5), Dst: 1.2.3.4 (1.2.3.4)
User Datagram Protocol, Src Port: 9999 (9999), Dst Port: sip (5060)
Session Initiation Protocol
```

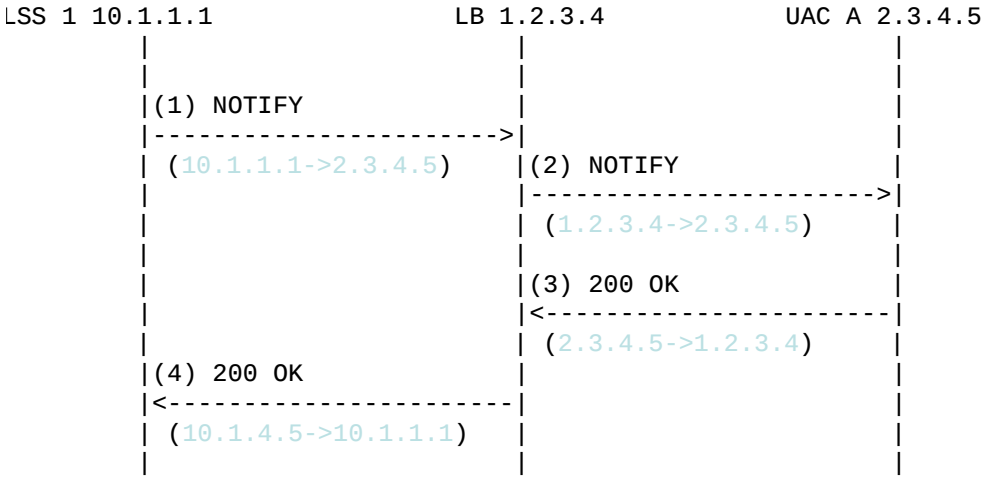
```
    Request-Line: SUBSCRIBE sip:subscribe@1.2.3.4:5060 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 2.3.4.5:9999;rport;branch=1
        From: sipp <sip:sipp@2.3.4.5>;tag=1
        To: sut <sip:subscribe@1.2.3.4:5060>
        Call-ID: 1-25923@2.3.4.5
        Cseq: 1 SUBSCRIBE
        Contact: sip:sipp@2.3.4.5:9999
        Max-Forwards: 70
        Event: ua-profile
        Expires: 10
        Content-Length: 0

No.     Time        Source                Destination           Protocol Info
      2 2.426250    10.1.3.4           10.1.1.1          SIP     Request:
SUBSCRIBE sip:subscribe@1.2.3.4:5060

Internet Protocol, Src: 10.1.3.4 (10.1.3.4), Dst: 10.1.1.1 (10.1.1.1)
User Datagram Protocol, Src Port: 2222 (2222), Dst Port: sip (5060)
Session Initiation Protocol
    Request-Line: SUBSCRIBE sip:subscribe@1.2.3.4:5060 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 2.3.4.5:9999;rport;branch=1
        From: sipp <sip:sipp@2.3.4.5>;tag=1
        To: sut <sip:subscribe@1.2.3.4:5060>
        Call-ID: 1-25923@2.3.4.5
        Cseq: 1 SUBSCRIBE
        Contact: sip:sipp@2.3.4.5:9999
```

```
        Max-Forwards: 70
        Event: ua-profile
        Expires: 10
        Content-Length: 0

No.     Time        Source                Destination           Protocol Info
     3 3.430903   10.1.1.1            10.1.3.4         SIP     Status: 200 OK

Internet Protocol, Src: 10.1.1.1 (10.1.1.1), Dst: 10.1.3.4 (10.1.3.4)
User Datagram Protocol, Src Port: 42316 (42316), Dst Port: 2222 (2222)
Session Initiation Protocol
    Status-Line: SIP/2.0 200 OK
    Message Header
        To: sut <sip:subscribe@1.2.3.4:5060>;tag=82722c03
        Content-Length: 0
        Contact:
<sip:app-12eomtm5h5f77@1.2.3.4:5060;transport=udp;wlsscid=1ae4479ac6ff71>
        CSeq: 1 SUBSCRIBE
        Call-ID: 1-25923@2.3.4.5
```

Figure 3-7 illustrates the NOTIFY flow.

Note that while source address NAT is enabled for both directions (UAS->WebLogic SIP Server and WebLogic SIP Server->UA), the load balancer can correctly identify the destination address in Step 3 by relying on receiving responses on the same port number as the one used to send requests. This implies that the load balancer maintains state.

Figure 3-7 rport NOTIFY Sequence



The complete message trace from Figure 3-7 is shown in Listing 3-8 below.

Listing 3-8 Complete Message Trace for rport NOTIFY

```
No.     Time        Source                Destination          Protocol Info
     1 5.430952    10.1.1.1              2.3.4.5         SIP     Request:
NOTIFY sip:sipp@2.3.4.5:9999


Internet Protocol, Src: 10.1.1.1 (10.1.1.1), Dst: 2.3.4.5 (2.3.4.5)
User Datagram Protocol, Src Port: 42316 (42316), Dst Port: 9999 (9999)
Session Initiation Protocol
    Request-Line: NOTIFY sip:sipp@2.3.4.5:9999 SIP/2.0
    Message Header
        To: sipp <sip:sipp@2.3.4.5>;tag=1
        Content-Length: 0
        Contact:
<sip:app-12eomtm5h5f77@1.2.3.4:5060;transport=udp;wlsscid=1ae4479ac6ff71>
```

```
        CSeq: 1 NOTIFY

        Call-ID: 1-25923@2.3.4.5

        Via: SIP/2.0/UDP
1.2.3.4:5060;wlsscid=1ae4479ac6ff71;branch=z9hG4bKc5e4c3b4c22be517133ab749
adeece4e;rport

        From: sut <sip:subscribe@1.2.3.4:5060>;tag=82722c03

        Max-Forwards: 70


No.     Time       Source               Destination          Protocol Info

      2 6.430952    1.2.3.4          2.3.4.5           SIP      Request:
NOTIFY sip:sipp@2.3.4.5:9999


Internet Protocol, Src: 1.2.3.4 (1.2.3.4), Dst: 2.3.4.5 (2.3.4.5)

User Datagram Protocol, Src Port: 2222 (2222), Dst Port: 9999 (9999)

Session Initiation Protocol

    Request-Line: NOTIFY sip:sipp@2.3.4.5:9999 SIP/2.0

    Message Header

        To: sipp <sip:sipp@2.3.4.5>;tag=1

        Content-Length: 0

        Contact:
<sip:app-12eomtm5h5f77@1.2.3.4:5060;transport=udp;wlsscid=1ae4479ac6ff71>

        CSeq: 1 NOTIFY

        Call-ID: 1-25923@2.3.4.5

        Via: SIP/2.0/UDP
1.2.3.4:5060;wlsscid=1ae4479ac6ff71;branch=z9hG4bKc5e4c3b4c22be517133ab749
adeece4e;rport

        From: sut <sip:subscribe@1.2.3.4:5060>;tag=82722c03

        Max-Forwards: 70
```

```
No.     Time        Source                Destination           Protocol Info
      3 7.431367    2.3.4.5          1.2.3.4         SIP      Status: 200 OK

Internet Protocol, Src: 2.3.4.5 (2.3.4.5), Dst: 1.2.3.4 (1.2.3.4)
User Datagram Protocol, Src Port: 9999 (9999), Dst Port: (2222)
Session Initiation Protocol
    Status-Line: SIP/2.0 200 OK
    Message Header
        Via: SIP/2.0/UDP
1.2.3.4:5060;wlsscid=1ae4479ac6ff71;branch=z9hG4bKc5e4c3b4c22be517133ab749
adeece4e;rport
        From: sut <sip:subscribe@1.2.3.4:5060>;tag=82722c03
        To: sipp <sip:sipp@2.3.4.5>;tag=1;tag=1
        Call-ID: 1-25923@2.3.4.5
        CSeq: 1 NOTIFY
        Contact: <sip:2.3.4.5:9999;transport=UDP
```

CHAPTER 4

Configuring Diameter Client Nodes and Relay Agents

The following sections describe how to configure individual servers to act as Diameter nodes or relays in a WebLogic SIP Server domain:

Overview of Diameter Protocol Configuration

A typical WebLogic SIP Server domain includes support for the Diameter base protocol and one or more IMS Diameter interface applications (Sh, Ro, Rf) deployed to engine tier servers that act as Diameter client nodes. SIP Servlets deployed on the engines can use the available Diameter applications to initiate requests for user profile data, accounting, and credit control, or to subscribe to and receive notification of profile data changes.

One or more server instances may be also be configured as Diameter relay agents, which route Diameter messages from the client nodes to a configured Home Subscriber Server (HSS) or other nodes in the network, but do not modify the messages. BEA recommends configuring one or more servers to act as relay agents in a domain. The relays simplify the configuration of Diameter client nodes, and reduce the number of network connections to the HSS. Using at least two relays ensures that a route can be established to an HSS even if one relay agent fails.

Note: In order to support multiple HSSs, the 3GPP defines the Dh interface to look up the correct HSS. WebLogic SIP Server 3.1 does not provide a Dh interface application, and can be configured only with a single HSS.

Note that relay agent servers do not function as either engine or data tier instances—they should not host applications, store call state data, maintain SIP timers, or even use SIP protocol network resources (sip or sips network channels).

WebLogic SIP Server also provides simulator applications for the Sh and Ro protocols. You can use the simulator applications for testing while developing Sh and Ro clients. The simulator applications are not intended for deployment to a production system.

Steps for Configuring Diameter Client Nodes and Relay Agents

To configure Diameter support in a WebLogic SIP Server domain, follow these steps:

1. (Optional) Install the WebLogic SIP Server Diameter Domain. Install the Diameter domain, which contains a sample configuration and template applications configured for different Diameter node types. You may use the Diameter domain as a template for your own domain, or to better understand how different Diameter node types are configured.
2. (Optional) Enable the Diameter console extension. If you are working with the sample Diameter domain, the Diameter console extension is already enabled. If you are starting with a basic Weblogic SIP Server domain, edit the `config.xml` file to enable the extension.
3. Create Diameter network channels. Create the network channels necessary to support Diameter over TCP, TLS, or SCTP transports on engine tier servers and relays.
4. Create and configure the Diameter nodes. Configure the Diameter protocol client applications on engine tier servers with the host name, peers, and routes to relay agents or other network elements, such as an HSS. You can also configure Diameter nodes that operate in standalone mode, without a WebLogic SIP Server instance.

The sections that follow describe each step in detail. See also the "Example Domain Configuration" on page 4-21.

Installing the Diameter Domain

The Configuration Wizard includes a Diameter domain template that creates a domain having four WebLogic SIP Server instances:

- An Administration Server (AdminServer)
- A Diameter Sh client node (hssclient)
- A Diameter relay node (relay)
- An HSS simulator (hss)

You can use the installed Diameter domain as the basis for creating your own domain. Or, you can use the customized Diameter Web Applications as templates for configuring existing WebLogic SIP Server instances to function as HSS client or relay agent nodes. The configuration instructions in the sections that follow assume that you have access to the Diameter domain configuration. Follow these steps to install the domain:

1. Change to the *WLSS_HOME*\common\bin directory, where *WLSS_HOME* is the directory in which you install WebLogic SIP Server (for example, c:\bea\sipserver31\common\bin).
2. Execute the config.cmd or config.sh script to launch the Configuration Wizard.
3. Select Create a new WebLogic Domain and click Next.
4. Select Base this domain on an existing template, and click the Browse button.
5. Select the diameterdomain.jar template and click OK.
6. Click Next.
7. Enter a username and password for the Administrator of the new domain, and click Next.
8. Select the startup mode and JDKs to use with the new domain, and click Next.
9. Select No to accept the default template options, and click Next.
10. Click Create to create the new domain using the default domain name and domain location (*BEA_HOME*/user_projects/domains/diameter).
11. Click Done.

Table 4-1 describes the server configuration installed with the Diameter domain.

Table 4-1 Key Configuration Elements of the Diameter Domain

Server Name	Network Channel Configuration	Diameter Applications	Notes
`AdminServer`	n/a	n/a	The Administration Server provides no SIP or Diameter protocol functionality.
`hssclient`	diameter (TCP over port 3868) sip (UDP/TCP over port 5060)	`WlssShApplication`	The `hssclient` engine functions as a Diameter Sh client node. The server contains network channels supporting both SIP and Diameter traffic. The Diameter node configuration deploys `WlssShApplication` (`com.bea.wcp.diameter.sh.WlssShApplication`) to provide IMS Sh interface functionality for deployed SIP Servlets.
`relay`	diameter (TCP over port 3869)	`RelayApplication`	The `relay` engine functions as a Diameter Sh relay node. The server contains a network channel to support both Diameter traffic. The server does not contain a channel to support SIP traffic, as a relay performs no SIP message processing. The Diameter node configuration deploys `RelayApplication` (`com.bea.wcp.diameter.relay.RelayApplication`) to provide relay services. The node configuration also defines a realm-based route for relaying messages from the `hssclient` engine.
`hss`	diameter (TCP over port 3870)	`HssSimulator`	The `hss` engine's Diameter node configuration deploys only the `HssSimulator` application (`com.bea.wcp.diameter.sh.HssSimulator`). The server is configured with a Diameter network channel.

Enabling the Diameter Console Extension

WebLogic SIP Server provides a console extension to help you create and configure Diameter nodes. The actual configuration generated by the extension is stored in a `diameter.xml` configuration file, stored in the `config/custom` subdirectory of the domain directory.

The sample Diameter domain already enables the Diameter console extension. If you are working with a domain that does not enable the extension, edit the `config.xml` file for the domain to specify the custom resource for the extension. Listing 4-1 highlights the `config.xml` entries necessary to enable the console extension. Use a text editor to add the highlighted lines in the correct location in `config.xml`.

Listing 4-1 config.xml Entries for Enabling the Diameter Console Extension

```
<custom-resource>
    <name>sipserver</name>
    <target>hssclient</target>
    <descriptor-file-name>custom/sipserver.xml</descriptor-file-name>

<resource-class>com.bea.wcp.sip.management.descriptor.resource.SipServerRe
source</resource-class>

<descriptor-bean-class>com.bea.wcp.sip.management.descriptor.beans.SipServ
erBean</descriptor-bean-class>
  </custom-resource>
  <custom-resource>
    <name>diameter</name>
    <target>hssclient,relay,hss</target>
    <deployment-order>200</deployment-order>
    <descriptor-file-name>custom/diameter.xml</descriptor-file-name>
    <resource-class>com.bea.wcp.diameter.DiameterResource</resource-class>
```

```
<descriptor-bean-class>com.bea.wcp.diameter.management.descriptor.beans.Di
ameterBean</descriptor-bean-class>
  </custom-resource>
  <custom-resource>
    <name>ProfileService</name>
    <target>hssclient</target>
    <deployment-order>300</deployment-order>
    <descriptor-file-name>custom/profile.xml</descriptor-file-name>

<resource-class>com.bea.wcp.profile.descriptor.resource.ProfileServiceReso
urce</resource-class>

<descriptor-bean-class>com.bea.wcp.profile.descriptor.beans.ProfileService
Bean</descriptor-bean-class>
  </custom-resource>
  <admin-server-name>AdminServer</admin-server-name>
</domain>
```

Creating TCP, TLS, and SCTP Network Channels for the Diameter Protocol

WebLogic SIP Server's Diameter implementation supports the Diameter protocol over the TCP, TLS, and SCTP transport protocols. (SCTP transport is provided with certain restrictions as described in "Configuring and Using SCTP for Diameter Messaging" on page 4-9.)

To enable incoming Diameter connections on a server, you must configure a dedicated network channel of the appropriate protocol type:

- "diameter" channels use TCP transport
- "diameters" channels use TCP/TLS transport
- "diameter-sctp" channels use TCP/SCTP transport.

Servers that use a TCP/TLS channel for Diameter (diameters channels) must also enable two-way SSL. WebLogic SIP Server may automatically upgrade Diameter TCP connections to use TLS as described in the Diameter specification (RFC 3558).

To configure a TCP or TCP/TLS channel for use with the Diameter provider, follow these steps:

1. Access the Administration Console for the WebLogic SIP Server domain.
2. Click Lock & Edit to obtain a configuration lock.
3. In the left pane, select the name of the server to configure.
4. In the right pane, select Protocols->Channels to display the configured channels.
5. Click New to configure a new channel.
6. Fill in the fields of the Identity Properties page as follows:
   - **Name:** Enter an administrative name for this channel, such as “Diameter TCP/TLS Channel.”
   - **Protocol:** Select “diameter” to support the TCP transport, “diameters” to support both TCP and TLS transports, or “diameter-sctp” to support TCP transport.

     Note: If a server configures at least one TLS channel, the server operates in TLS mode and will reject peer connections from nodes that do not support TLS (as indicated in their capabilities exchange).
7. Click Next to continue.
8. Fill in the fields of the Network Channel Addressing page as follows:
   - **Listen Address:** Enter the IP address or DNS name for this channel. On a multi-homed machine, enter the exact IP address of the interface you want to configure, or a DNS name that maps to the exact IP address.
   - **Listen Port:** Enter the port number used to communication via this channel. Diameter nodes conventionally use port 3868 for incoming connections.
   - **External Listen Port**: Re-enter the Listen Port value.
9. Click Next to continue.
10. Chose attributes in the Network Channel Properties page as follows:
    - **Enabled**: Select this attribute to ensure that the new channel accepts network traffic.
    - **Tunneling Enabled**: Un-check this attribute for Diameter channels.

- **HTTP Enabled for this Protocol**: Un-check this attribute for Diameter channels.
- **Outbound Enabled**: Select this attribute to ensure that the node can initiate Diameter messages using the channel.

11. Click Next to continue.
12. For “diameters” channels, select the following two attributes:
    - **Two Way SSL Enabled**: Two-way SSL is required for TLS transport.
    - **Client Certificate Enforced**: Select this attribute to honor available client certificates for secure communication.
13. Click Finish to create the new channel.
14. Select the name of the newly-created channel in the Network Channel table.
15. Display the advanced configuration items for the newly-created channel by clicking the Advanced link.
16. Change the **Idle Connection Timeout** value from the default (65 seconds) to a larger value that will ensure the Diameter connection remains consistently available.

    Note: If you do not change the default value, the Diameter connection will be dropped and recreated every 65 seconds with idle traffic.
17. Click Save.
18. Click Activate Changes.

The servers installed with the Diameter domain template include network channel configurations for Diameter over TCP transport. Note that the relays server includes only a diameter channel and *not* a sip or sips channel. Relay agents should not host SIP Servlets or other applications, therefore no SIP transports should be configured on relay server nodes.

Configuring Two-Way SSL for Diameter TLS Channels

Diameter channels that use TLS (diameters channels) require that you also enable two-way SSL, which is disabled by default. Follow these steps to enable two-way SSL for a server. If you have not already configured SSL, see Configuring SSL in the WebLogic Server 9.2 Documentation for instructions.

Configuring and Using SCTP for Diameter Messaging

SCTP is a reliable, message-based transport protocol that is designed for use in telephony networks. SCTP provides several benefits over TCP:

- SCTP preserves the internal structure of messages when transmitting data to an endpoint, whereas TCP transmits raw bytes that must be received in order.
- SCTP supports multihoming, where each endpoint may have multiple IP addresses. The SCTP protocol can transparently failover to another IP address should a connection fail.
- SCTP provides multistreaming capabilities, where multiple streams in a connection transmit data independently of one another.

WebLogic SIP Server supports SCTP for Diameter network traffic, with several limitations:

- Only 1 stream per connection is currently supported.
- SCTP can be used only for Diameter network traffic; SIP traffic cannot use a configured SCTP channel.
- TLS is not supported over SCTP.

In addition, WebLogic SIP Server only supports SCTP channels on the following software platforms:

- Red Hat Enterprise Linux 4.0 AS, ES, WS (Kernel 2.6.9, GCC 3.4 or higher) on 32- or 64-bit hardware
- Sun Solaris 10 on SPARC

SCTP channels can operate on either IPv4 or IPv6 networks. Creating TCP, TLS, and SCTP Network Channels for the Diameter Protocol describes how to create a new SCTP channel. To enable multihoming capabilities for an existing SCTP channel, specify the IPv4 address `0.0.0.0` as the listen address for the channel (or use the `::` address for IPv6 networks).

Configuring Diameter Nodes

The Diameter node configuration for WebLogic SIP Server engines is stored in the `diameter.xml` configuration file (`domain_home/config/custom/diameter.xml`). If you want to provide diameter services (client, server, or relay functionality) on an engine tier server, you must create a new node configuration and target the configuration to an existing engine server instance.

Diameter node configurations are divided into several categories:

- General configuration defines the host identity and realm for the node, as well as basic connection information and default routing behavior.
- Application configuration defines the Diameter application(s) that run on the node, as well as any optional configuration parameters passed to those applications.
- Peer configuration defines the other Diameter nodes with which this node operates.
- Routes configuration defines realm-based routes that the node can use when resolving messages.

The sections that follow describe how to configure each aspect of a Diameter node.

Creating a New Node Configuration (General Node Configuration)

Follow these steps to create a new Diameter node configuration and target it to an existing WebLogic SIP Server engine tier instance:

1. Log in to the Administration Console for the WebLogic SIP Server domain you want to configure.
2. Click Lock & Edit to obtain a configuration lock.
3. Select the Diameter node in the left pane of the Console.
4. Click New in the right pane to create a new Diameter configuration.

5. Fill in the fields of the Create a New Configuration page as described in Table 4-2, then click Finish.

Table 4-2 Diameter Node General Configuration Properties

Property Name	Description
Name	Enter the an administrative name for this Diameter node configuration.
Host	Enter the host identity of this Diameter node, or leave the field blank to automatically assign the host name of the target engine tier server as the Diameter node's host identity. Note that the host identity may or may not match the DNS name. When configuring Diameter support for multiple Sh client nodes, it is best to omit the `host` element from the `diameter.xml` file. This enables you to deploy the same Diameter Web Application to all servers in the engine tier cluster, and the host name is dynamically obtained for each server instance.
Realm	Enter the realm name for which this node has responsibility, or leave the field blank to use the domain name portion of the target engine tier server's fully-qualified host name (for example, host@bea.com). You can run multiple Diameter nodes on a single host using different realms and listen port numbers. **Note:** An HSS, Application Server, and relay agents must all agree on a realm name or names. The realm name for the HSS and Application Server need not match.
Address	Enter the listen address for this Diameter node, using either the DNS name or IP address, or leave the field blank to use the host identity as the listen address. **Note:** The host identity may or may not match the DNS name of the Diameter node. BEA recommends configuring the Address property with an explicit DNS name or IP address to avoid configuration errors.
TLS	Select this option if the Diameter node us configured with support for TLS (diameters network channels). This field is used to advertise TLS capabilities when the node is interrogated by another Diameter node.
Debug	Select this option if you want to enable debug message output. Debug messages are disabled by default.
Message Debug	Select this option if you want to enable tracing for Diameter messages processed by this node. Message tracing is disabled by default.

Table 4-2 Diameter Node General Configuration Properties

Property Name	Description
Dynamic Peers Allowed	Select this option to allow dynamic discovery of Diameter peer nodes. Dynamic peer support is disabled by default. BEA recommends enabling dynamic peers only when using the TLS transport, because no access control mechanism is available to restrict hosts from becoming peers.
Peer Retry Delay	Enter the amount of time, in seconds, this node waits before retrying a request to a Diameter peer. The default value is 30 seconds.
Request Timeout	Enter the amount of time, in milliseconds, this node waits for an answer message before timing out.
Watchdog Timeout	Enter the number of seconds this node uses for the value of the Diameter Tw watchdog timer interval.
Targets	Enter one or more target engine tier server names. The Diameter node configuration only applies to servers listed in this field.
Default Route Action	Specify an action type that describes the role of this Diameter node when using a default route. The value of this element can be one of the following: • none • local • relay • proxy • redirect
Default Route Servers	Specifies one or more target servers for the default route. Any server you include in this element must also be defined as a peer to this Diameter node, or dynamic peer support must be enabled.

6. Click Activate Changes to apply the configuration to target servers.

After creating a general node configuration, the configuration name appears in the list of Diameter nodes. You can select the node to configure Diameter applications, peers, and routes, as described in the sections that follow.

Configuring Diameter Applications

Each Diameter node can deploy one or more applications. You configure Diameter applications in the Administration Console using the Configuration->Applications page for a selected Diameter node. Follow these steps:

1. Log in to the Administration Console for the WebLogic SIP Server domain you want to configure.
2. Click Lock & Edit to obtain a configuration lock.
3. Select the Diameter node in the left pane of the Console.
4. Select the name of a Diameter node configuration in the right pane of the Console.
5. Select the Configuration->Applications tab.
6. Click New to configure a new Diameter application, or select an existing application configuration from the table.
7. Fill in the application properties as follows:
   - **Application Name**: Enter a name for the application configuration.
   - **Class Name**: Enter the classname of the application to deploy on this node.
   - **Parameters**: Enter optional parameters to pass to the application upon startup.
8. Click Finish to create the new application configuration.
9. Click Activate Changes to apply the configuration to the Diameter node.

WebLogic SIP Server includes several Diameter applications to support clients using the Sh, Rf, and Ro interfaces, Diameter relays, and simulators for the Sh and Ro interfaces. The sections that follow provide more information about configuring these WebLogic SIP Server Diameter applications.

You can also use the base Diameter API included in WebLogic SIP Server to create and deploy your own Diameter applications. See Using the Diameter Base Protocol API in *Developing Applications with WebLogic SIP Server* for more information.

Configuring the Sh Client Application

The Sh client application is implemented as a provider to the base Diameter protocol support in WebLogic SIP Server. The application transparently generates and responds to the Diameter command codes defined in the Sh application specification. A higher-level API enables SIP Servlets to manage user profile data as an XML document using XML Document Object Model (DOM). Subscriptions and notifications for changed profile data are managed by implementing a profile listener interface in a SIP Servlet. See Using the Diameter Sh Interface Application in *Developing Applications with WebLogic SIP Server* for more information about the API.

The Diameter nodes on which you deploy the Sh client application should be configured with:

- The host names of any relay agents configured in the domain, defined as Diameter peer nodes. If no relay agents are used, all engine tier servers must be added to the list of peers, or dynamic peers must be enabled.
- One or more routes to access relay agent nodes (or the HSS) in the domain.

To configure the Sh client application, you specify the `com.bea.wcp.diameter.sh.WlssShApplication` class. `WlssShApplication` accepts the following parameters:

- `destination.host` configures a static route to the specified host. Include a `destination.host` param definition only if servers communicate directly to an HSS (static routing), without using a relay agent. Omit the `destination.host` param completely when routing through relay agents.
- `destination.realm`—configures a static route to the specified realm. Specify the realm name of relay agent servers or the HSS, depending on whether or not the domain uses relay agents.

Listing 4-2 shows a sample node configuration for an Sh client node that uses a relay.

Listing 4-2 Sample Diameter Node Configuration with Sh Client Application

```
<?xml version='1.0' encoding='utf-8'?>
<diameter xmlns="http://www.bea.com/ns/wlcp/diameter/300"
xmlns:sec="http://www.bea.com/ns/weblogic/90/security"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wls="http://www.bea.com/ns/weblogic/90/security/wls">
  <configuration>
    <name>hssclient</name>
    <target>hssclient</target>
    <host>hssclient</host>
    <realm>bea.com</realm>
    <!-- Omit the host and realm elements to dynamically assign the hostname
     and domain name of individual engine tier servers. -->
    <message-debug-enabled>true</message-debug-enabled>
```

```
<application>
  <name>WlssShApplication</name>
  <class-name>com.bea.wcp.diameter.sh.WlssShApplication</class-name>
  <param>
  <!-- Include a destination.host param definition only if servers will
       communicate directly to an HSS (static routing), without using
       a relay agent. Omit the destination.host param completely when
       routing through relay agents. -->
  <!-- Specify the realm name of relay agent servers or the HSS,
       depending on whether or not the domain uses relay agents. -->
    <name>destination.realm</name>
    <value>hss.com</value>
  </param>
</application>
<peer>
<!-- Include peer entries for each relay agent server used in the domain.
     If no relay agents are used, include a peer entry for the HSS
     itself, as well as for all other Sh client nodes (all other engine
     tier servers in the domain).
     Alternately, use the allow-dynamic-peers functionality in
     combination with TLS transport to allow peers to be recognized
     automatically. -->
  <host>relay</host>
  <address>localhost</address>
  <!-- The address element can specify either a DNS name or IP address,
       whereas the host element must specify a diameter host identity.
       The diameter host identity may or may not match the DNS name. -->
```

```
      <port>3869</port>
    </peer>
    <!-- Enter a default route to a selected relay agent. If the domain does
         not use a relay agent, specify a default route to relay messages
         directly to the HSS. -->
    <default-route>
      <action>relay</action>
      <server>relay</server>
    </default-route>
  </configuration>
</diameter>
```

Configuring the Rf Client Application

The WebLogic SIP Server Rf client application enables SIP Servlets to issue offline charging messages using the IMS Rf interface. To configure the Rf application, specify the class `com.bea.wcp.diameter.charging.RfApplication`. The Rf application accepts the following parameters:

- `cdf.host` specifies the host name of the Charging Data Function (CDF).
- `cdf.realm` specifies the realm of the CDF.

See Using the Diameter Rf Interface Application for Offline Charging in *Developing Applications with WebLogic SIP Server* for more information about using the Rf application API in deployed applications.

Configuring the Ro Client Application

The WebLogic SIP Server Ro client application enables SIP Servlets to issue online charging messages using the IMS Ro interface. To configure the Rf application, specify the class `com.bea.wcp.diameter.charging.RoApplication`. The Ro application accepts the following parameters:

- `ocs.host` specifies the host identity of the Online Charging Function (OCF). The OCF you specify host must also be configured as the peer for the Diameter node on which the Ro application is deployed.

- `ocs.realm` can be used instead of `ocs.host` for realm-based routing when using more than one OCF host. The corresponding realm definition must also exist in the Diameter node's configuration.

See Using the Diameter Ro Interface Application for Online Charging in *Developing Applications with WebLogic SIP Server* for more information about using the Ro application API in deployed applications.

Configuring a Diameter Relay Agent

Relay agents are not required in a Diameter configuration, but BEA recommends using at least two relay agent servers to limit the number of direct connections to the HSS, and to provide multiple routes to the HSS in the event of a failure.

Note: You must ensure that relay servers *do not* also act as WebLogic SIP Server engine tier servers or data tier servers. This means that the servers should not be configured with "sip" or "sips" network channels.

Relay agent nodes route Sh messages between client nodes and the HSS, but they do not modify the messages except as defined in the Diameter Sh specification. Relays always route responses from the HSS back the client node that initiated the message, or the message the response is dropped if that node is unavailable.

To configure a Diameter relay agent, simply configure the node to deploy an application with the class `com.bea.wcp.diameter.relay.RelayApplication`.

The node on which you deploy the relay application should also configure:

- All other nodes as peers to the relay node.
- A default route that specifies the relay action.

Listing 4-3 shows the sample `diameter.xml` configuration for a relay agent node.

Listing 4-3 Diameter Relay Node Configuration

```
<?xml version='1.0' encoding='utf-8'?>
<diameter xmlns="http://www.bea.com/ns/wlcp/diameter/300"
xmlns:sec="http://www.bea.com/ns/weblogic/90/security"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wls="http://www.bea.com/ns/weblogic/90/security/wls">
  <configuration>
```

```
<name>relay</name>
    <target>relay</target>
    <host>relay</host>
    <realm>bea.com</realm>
    <message-debug-enabled>true</message-debug-enabled>
    <application>
      <name>RelayApplication</name>
      <class-name>com.bea.wcp.diameter.relay.RelayApplication</class-name>
    </application>
    <!-- Define peer connection information for each Diameter node, or use
         the allow-dynamic-peers functionality in combination with TLS
         transport to allow peers to be recognized automatically. -->
    <peer>
      <host>hssclient</host>
      <address>localhost</address>
      <port>3868</port>
    </peer>
    <peer>
      <host>hss</host>
      <address>localhost</address>
      <port>3870</port>
    </peer>
    <route>
      <realm>bea.com</realm>
      <application-id>16777217</application-id>
      <action>relay</action>
      <server>hssclient</server>
```

```
    </route>
    <!-- Enter a default route for this agent to relay messages
         to the HSS. -->
    <default-route>
      <action>relay</action>
      <server>hss</server>
    </default-route>
  </configuration>
</diameter>
```

Configuring the Sh and Rf Simulator Applications

WebLogic SIP Server contains two simulator applications that you can use in development or testing environments to evaluate Diameter client applications. To configure a simulator application, you simply deploy the corresponding class to a configured Diameter node:

- `com.bea.wcp.diameter.sh.HssSimulator` simulates an HSS in your domain for testing Sh client applications.
- `com.bea.wcp.diameter.rf.RfSimulator` simulates an CDF host for testing Rf client applications

Note: These simulators are provided for testing or development purposes only, and is not meant as a substitute for a production HSS or CDF.

Diameter nodes that deploy simulator applications can be targeted to running engine tier servers, or they may be started as standalone Diameter nodes. When started in standalone mode, simulator applications accept the command-line options described in Table 4-3. See Working with Diameter Nodes in *Developing Applications with WebLogic SIP Server* for more information.

Table 4-3 Command-Line Options for Simulator Applications

Option	Description
`-r`, `-realm` *`realm_name`*	Specifies the realm name of the Diameter node.
-h, -host *`host_name`*	Specifies the host identity of the node.
-a, -address *`address`*	Specifies the listen address for this node.
`-p`, `-port` *`port_number`*	Specifies the listen port number for this node.

Table 4-3 Command-Line Options for Simulator Applications

Option	Description
`-d`, `-debug`	Enables debug output.
`-m`, `-mdebug`	Enables Diameter message tracing.

Configuring Peer Nodes

A Diameter node should define peer connection information for each other Diameter node in the realm, or enable dynamic peers in combination with TLS transport to allow peers to be recognized automatically. You configure Diameter peer nodes in the Administration Console using the Configuration->Peers page for a selected Diameter node. Follow these steps:

1. Log in to the Administration Console for the WebLogic SIP Server domain you want to configure.
2. Click Lock & Edit to obtain a configuration lock.
3. Select the Diameter node in the left pane of the Console.
4. Select the name of a Diameter node configuration in the right pane of the Console.
5. Select the Configuration->Peers tab.
6. Click New to define a new peer entry.
7. Fill in the fields of the Create a New Peer page as follows:
   - **Host**: Enter the peer node's host identity.
   - **Address**: Enter the peer node's address (DNS name or IP address).
   - **Port Number**: Enter the listen port number of the peer node.
   - **Protocol**: Select the protocol used to communicate with the peer (TCP or SCTP).

     Note: WebLogic SIP Server attempts to connect to the peer using *only* the protocol you specify (TCP or SCTP). The other protocol is not used, even if a connection fails using the selected protocol. TCP is used as by default if you do not specify a protocol.
   - **Watchdog**: Indicate whether the peer supports the Diameter Tw watchdog timer interval.
8. Click Finish to create the new peer entry.

9. Click Activate Changes to apply the configuration.

Configuring Routes

Certain Diameter nodes, such as relays, should configure realm-based routes for use when resolving Diameter messages. You configure Diameter routes in the Administration Console using the Configuration->Routes page for a selected Diameter node. Follow these steps:

1. Log in to the Administration Console for the WebLogic SIP Server domain you want to configure.
2. Click Lock & Edit to obtain a configuration lock.
3. Select the Diameter node in the left pane of the Console.
4. Select the name of a Diameter node configuration in the right pane of the Console.
5. Select the Configuration->Routes tab.
6. Click New to configure a new Route.
7. Fill in the fields of the Create a New Route page as follows:
   - **Name**: Enter an administrative name for the route.
   - **Realm**: Enter the target realm for this route.
   - **Application ID**: Enter the target Diameter application ID for this route.
   - **Action**: Select an action that this node performs when using the configured route. The action type may be one of: none, local, relay, proxy, or redirect.
   - **Server Names**: Enter the names of target servers that will use the route.
8. Click Finish to create the new route entry.
9. Click Activate Changes to apply the configuration.

See Listing 4-3, “Diameter Relay Node Configuration,” on page 4-17 for an example `diameter.xml` node configuration containing a route entry.

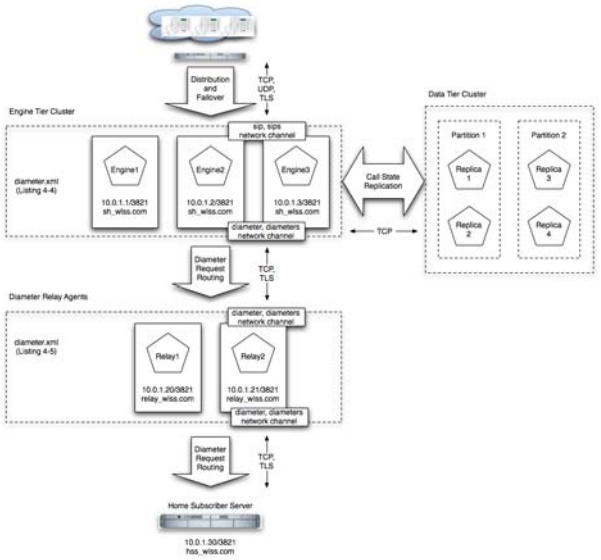
Example Domain Configuration

This section describes a sample WebLogic SIP Server configuration that provides basic Diameter Sh protocol capabilities. The layout of the sample domain includes the following:

- Three engine tier servers which host SIP applications and also deploy the Diameter Sh application for accessing user profiles.
- Four data tier servers arranged into two partitions with two replicas each.
- Two servers that act as Diameter relay agents and forward diameter requests to an HSS.

Figure 4-1 shows the individual servers in the sample configuration.

Figure 4-1 Sample Diameter Domain



Listing 4-4 shows the contents of the `diameter.xml` file used to configure engine tier servers (Sh Clients) in the sample domain. Listing 4-5 shows the `diameter.xml` file used to configure the relay agents.

Listing 4-4 diameter.xml Configuration for Sample Engine Tier Cluster (Sh Clients)

```
<?xml version='1.0' encoding='utf-8'?>

<diameter xmlns="http://www.bea.com/ns/wlcp/diameter/300"
xmlns:sec="http://www.bea.com/ns/weblogic/90/security"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wls="http://www.bea.com/ns/weblogic/90/security/wls">
<configuration>
    <name>clientnodes</name>
    <target>Engine1</target>
    <target>Engine2</target>
    <target>Engine3</target>
    <realm>sh_wlss.com</realm>
    <application>
      <name>WlssShApplication</name>
      <class-name>com.bea.wcp.diameter.sh.WlssShApplication</class-name>
      <param>
        <name>destination.realm</name>
        <value>relay_wlss.com</value>
      </param>
    </application>
    <peer>
      <host>Relay1</host>
      <address>10.0.1.20</address>
      <port>3821</port>
    </peer>
    <peer>
      <host>Relay2</host>
      <address>10.0.1.21</address>
      <port>3821</port>
    </peer>
    <default-route>
      <action>relay</action>
```

```
      <server>Relay1</server>
    </default-route>
    <route>
      <action>relay</action>
      <server>Relay2</server>
    </route>
  </configuration>
</diameter>
```

Listing 4-5 diameter.xml Configuration for Sample Relay Agents

```
<?xml version='1.0' encoding='utf-8'?>
<diameter xmlns="http://www.bea.com/ns/wlcp/diameter/300"
xmlns:sec="http://www.bea.com/ns/weblogic/90/security"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wls="http://www.bea.com/ns/weblogic/90/security/wls">
  <configuration>
    <name>relaynodes</name>
    <target>Relay1</target>
    <target>Relay2</target>
    <realm>relay_wlss.com</realm>
    <application>
      <name>RelayApplication</name>
      <class-name>com.bea.wcp.diameter.relay.RelayApplication</class-name>
    </application>
    <peer>
      <host>Engine1</host>
      <address>10.0.1.1</address>
      <port>3821</port>
```

```
</peer>
<peer>
  <host>Engine2</host>
  <address>10.0.1.2</address>
  <port>3821</port>
</peer>
<peer>
  <host>Engine3</host>
  <address>10.0.1.3</address>
  <port>3821</port>
</peer>
<peer>
  <host>Relay1</host>
  <address>10.0.1.20</address>
  <port>3821</port>
</peer>
<peer>
  <host>Relay2</host>
  <address>10.0.1.21</address>
  <port>3821</port>
</peer>
<peer>
  <host>hss</host>
  <address>hssserver</address>
  <port>3870</port>
</peer>
<default-route>
```

```
      <action>relay</action>
      <server>hss</server>
    </default-route>
  </configuration>
</diameter>
```

Troubleshooting Diameter Configurations

SIP Servlets deployed on WebLogic SIP Server use the available Diameter applications to initiate requests for user profile data, accounting, and credit control, or to subscribe to and receive notification of profile data changes. If a SIP Servlet performing these requests generates an error similar to:

`Failed to dispatch Sip message to servlet` *`ServletName`*

`java.lang.IllegalArgumentException: No registered provider for protocol:` *`Protocol`*

The message may indicate that you have not properly configured the associated Diameter application for the protocol. See "Configuring Diameter Applications" on page 4-12 for more information.

If you experience problems connecting to a Diameter peer node, verify that you have configured the correct protocol for communicating with the peer in "Configuring Peer Nodes" on page 4-20. Keep in mind that WebLogic SIP Server tries only the protocol you specify for the peer configuration (or TCP if you do not specify a protocol).