BEA WebLogic SIP Server™

Configuration Reference Manual

Version 3.0
Revised: December 13, 2006

Copyright

Restricted Rights Legend

Trademarks and Service Marks

Contents

1. Engine Tier Configuration Reference (sipserver.xml)

CHAPTER 1

Engine Tier Configuration Reference (sipserver.xml)

The following sections provide a complete reference to the engine tier configuration file, `sipserver.xml`:

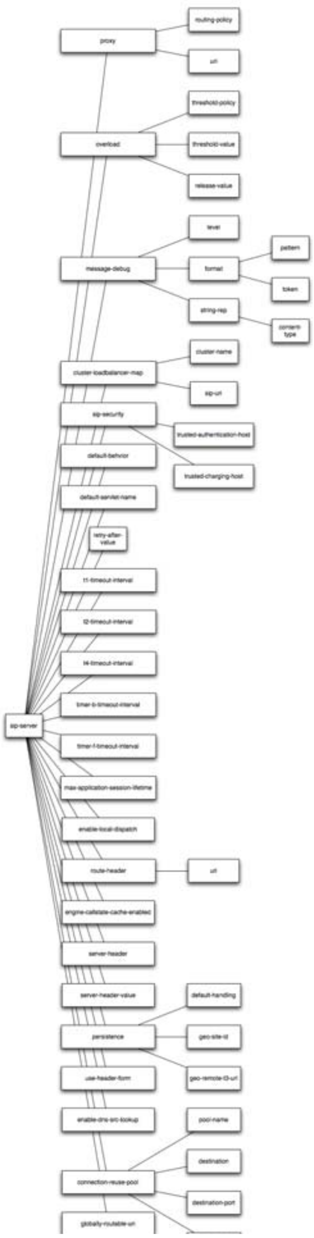
Overview of sipserver.xml

The `sipserver.xml` file is an XML document that configures the SIP container features provided by a WebLogic SIP Server instance in the engine tier of a server installation. `sipserver.xml` is stored in the *`DOMAIN_DIR`*`/config/custom` subdirectory where *`DOMAIN_DIR`* is the root directory of the WebLogic SIP Server domain.

Graphical Representation

Figure 1-1 shows the element hierarchy of the `sipserver.xml` deployment descriptor file.

Figure 1-1 Element Hierarchy of sipserver.xml



Editing sipserver.xml

You should never move, modify, or delete the `sipserver.xml` file during normal operations.

BEA recommends using the Administration Console to modify `sipserver.xml` indirectly, rather than editing the file by hand. Using the Administration Console ensures that the `sipserver.xml` document always contains valid XML. See also Configuring Container Properties Using WLST (JMX) in the *Configuration Guide*.

You may need to manually view or edit `sipserver.xml` to troubleshoot problem configurations, repair corrupted files, or to roll out custom configurations to a large number of machines when installing or upgrading WebLogic SIP Server. When you manually edit `sipserver.xml`, you must reboot WebLogic SIP Server instances to apply your changes.

WARNING: Always use the SIP Servers node in the Administration Console or the WLST utility, as described in Configuring Engine Tier Container Properties in the *Configuration Guide* to make changes to a running WebLogic SIP Server deployment.

Steps for Editing sipserver.xml

If you need to modify `sipserver.xml` on a production system, follow these steps:

1. Use a text editor to open the *`DOMAIN_DIR`*`/config/custom/sipserver.xml` file, where *`DOMAIN_DIR`* is the root directory of the WebLogic SIP Server domain.
2. Modify the `sipserver.xml` file as necessary. See "XML Schema" on page 1-4 for a full description of the XML elements.
3. Save your changes and exit the text editor.
4. Reboot or start servers to have your changes take effect:

WARNING: Always use the SIP Servers node in the Administration Console or the WLST utility, as described in Configuring Engine Tier Container Properties in the *Configuration Guide*, to make changes to a running WebLogic SIP Server deployment.

5. Test the updated system to validate the configuration.

XML Schema

The schema file for `sipserver.xml` is available at http://www.bea.com/ns/wlcp/wlss/300. A local copy (`wcp-sipserver.xsd`) is also installed inside the `wlss-descriptor-binding.jar` library, located in the *`WL_HOME`*`/server/lib/wlss` directory.

Example sipserver.xml File

The following shows a simple example of a `sipserver.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<sip-server xmlns="http://www.bea.com/ns/wlcp/wlss/300">
  <overload>
    <threshold-policy>queue-length</threshold-policy>
    <threshold-value>200</threshold-value>
    <release-value>150</release-value>
  </overload>
</sip-server>
```

XML Element Description

The following sections describe each element used in the sipserver.xml configuration file. Each section describes an XML element that is contained within the main `sip-server` element shown in Figure 1-1.

overload

The `overload` element enables you to throttle incoming SIP requests according to a configured overload condition. When an overload condition occurs, WebLogic SIP Server destroys new SIP requests by responding with "503 Service Unavailable" until the configured release value is observed, or until the capacity of the server's execute queues is exceeded (see "Overload Control Based on Queue Length" on page 1-7).

User-configured overload controls are applied only to initial SIP requests; SIP dialogues that are already active when an overload condition occurs may generate additional SIP requests that are not throttled.

To configure an overload control, you define the three elements described in Table 1-1.

Table 1-1 Nested overload Elements

Element	Description
`threshold-policy`	A String value that identifies the type of measurement used to monitor overload conditions: • `session-rate` measures the rate at which new SIP requests are generated. WebLogic SIP Server determines the session rate by calculating the number of new SIP application connections that were created in the last 5 seconds of operation. See “Overload Control Based on Session Generation Rate” on page 1-7. • `queue-length` measures the sum of the sizes of the capacity constraint work manager components that processes SIP requests and SIP timers. See “Overload Control Based on Queue Length” on page 1-7. You must use only one of the above policies to define an overload control. See “Selecting an Appropriate Overload Policy” on page 1-6 for more information.

Table 1-1 Nested overload Elements

Element	Description
`threshold-value`	Specifies the measured value that causes WebLogic SIP Server to *start* throttling new SIP requests: • When using the `session-rate` threshold policy, `threshold-value` specifies the number of new SIP requests per second that trigger an overload condition. See “Overload Control Based on Session Generation Rate” on page 1-7. • When using the `queue-length` threshold policy, `threshold-value` specifies the size of the combined number of requests in the SIP transport and SIP timer capacity constraint components that triggers an overload condition. See “Overload Control Based on Queue Length” on page 1-7. After the `threshold-value` is observed, WebLogic SIP Server throttles new SIP requests until the `release-value` value is observed.
`release-value`	Specifies the measured value that causes WebLogic SIP Server to *stop* throttling new SIP requests: • When using the `session-rate` threshold policy, `release-value` specifies the number of new SIP requests per second that terminates session throttling. See “Overload Control Based on Session Generation Rate” on page 1-7. • When using the `queue-length` threshold policy, `release-value` specifies the combined number of requests in the capacity constraints that terminates session throttling. See “Overload Control Based on Queue Length” on page 1-7.

Selecting an Appropriate Overload Policy

WebLogic SIP Server provides two different policies for throttling SIP requests:

- The `session-rate` policy throttles sessions when the volume new SIP sessions reaches a configured rate (a specified number of sessions per second).
- The `queue-length` policy throttles requests after the sum of the requests in the `wlss.transport.capacity` and `wlss.timer.capacity` capacity constraint components

reaches a configured size. These capacity constraints are configured as part of the `wlss.transport` and `wlss.timer` execute queues.

Note that you must select only one of the available overload policies. You cannot use both policies simultaneously.

The `session-rate` policy is generally used when a back-end resource having a known maximum throughput (for example, an RDBMS) is used to set up SIP calls. In this case, the `session-rate` policy enables you to tie the WebLogic SIP Server overload policy to the known throughput capabilities of the back-end resource.

With the `queue-length` policy, WebLogic SIP Server monitors both CPU and I/O bottlenecks to diagnose an overload condition. The `queue-length` policy is generally used with CPU-intensive SIP applications in systems that have no predictable upper bound associated with the call rate.

The following sections describe each policy in detail.

Overload Control Based on Session Generation Rate

WebLogic SIP Server calculates the session generation rate (sessions per second) by monitoring the number of application sessions created in the last 5 seconds. When the session generation rate exceeds the rate specified in the `threshold-value` element, WebLogic SIP Server throttles initial SIP requests until the session generation rate becomes smaller than the configured `release-value`.

The following example configures WebLogic SIP Server to begin throttling SIP requests when the new sessions are created at a rate higher than 50 sessions per second. Throttling is discontinued when the session rate drops to 40 sessions per second:

```
<overload>
  <threshold-policy>session-rate</threshold-policy>
  <threshold-value>50</threshold-value>
  <release-value>40</release-value>
</overload>
```

Overload Control Based on Queue Length

By default, SIP messages are handled by a work manager named `wlss.transport` and SIP timers are processed by a work manager named `wlss.timer`. These work managers are configured in the `config.xml` file for your WebLogic SIP Server installation. Each work manager has an associated capacity constraint component that sets the number of requests allotted for SIP message handling and timer processing.

WebLogic SIP Server performs `queue-length` overload control by monitoring the combined lengths of the configured capacity constraints. When the sum of the requests in the two constraints exceeds the length specified in the `threshold-value` element, WebLogic SIP Server throttles initial SIP requests until the total requests are reduced to the configured `release-value`.

Listing 1-1 shows a sample `overload` configuration from `sipserver.xml`. Here, WebLogic SIP Server begins throttling SIP requests when the combined size of the `wlss.transport.capacity` and `wlss.timer.capacity` constraints exceeds 200 requests. Throttling is discontinued when the combined length returns to 200 or fewer simultaneous requests.

Listing 1-1 Sample overload Definition

```
<overload>
  <threshold-policy>queue-length</threshold-policy>
  <threshold-value>200</threshold-value>
  <release-value>150</release-value>
</overload>
```

Two Levels of Overload Protection

User-configured overload controls (defined in `sipserver.xml`) represent the first level of overload protection provided by WebLogic SIP Server. They mark the onset of an overload condition and initiate simple measures to avoid dropped calls (generating 503 responses for new requests).

If the condition that caused the overload persists or worsens, then the work manager component used to perform work in the SIP Servlet container may itself become overloaded. At this point, the server no longer longer utilizes threads to generate 503 responses, but instead begins to drop messages. In this way, the configured size of the SIP container's work manager components (`wlss.transport.capacity` and `wlss.timer.capcity`) represent the second and final level of overload protection employed by the server.

Always configure overload controls in `sipserver.xml` conservatively, and resolve the circumstances that caused the overload in a timely fashion. Overload conditions should never be permitted to last until the point where the execute queue capacities are exceeded.

message-debug

The `message-debug` element is used to enable and configure access logging for WebLogic SIP Server. This element should be used only in a development environment, because access logging logs *all* SIP requests and responses. See Enabling Access Logging in *Developing Applications with WebLogic SIP Server* for information about configuring and using access logging.

If you want to perform more selective logging in a production environment, see Logging SIP Requests and Responses in the *Operations Guide*.

proxy—Setting Up an Outbound Proxy Server

RFC 3261 defines an outbound proxy as "A proxy that receives requests from a client, even though it may not be the server resolved by the Request-URI. Typically, a UA is manually configured with an outbound proxy, or can learn about one through auto-configuration protocols."

In WebLogic SIP Server an outbound proxy server is specified using the `proxy` element in `sipserver.xml`. The proxy element defines one or more proxy server URIs. You can change the behavior of the proxy process by setting a proxy policy with the `proxy-policy` tag. Listing 1-2 describes the possible values for the `proxy` elements.

The default behavior is as if **domain** policy is in effect. The **proxy** policy means that the request is sent out to the configured outbound proxy and the route headers in the request preserve any routing decision taken by WebLogic SIP Server. This enables the outbound proxy to send the request over to the intended recipient after it has performed its actions on the request. The **proxy** policy comes into effect only for the initial requests. As for the subsequent request the Route Set takes precedence over any policy in a dialog. (If the outbound proxy wants to be in the Route Set it can turn record routing on).

Also if a proxy application written on WebLogic SIP Server wishes to override the configured behavior of outbound proxy traversal, then it can add a special header with name "X-BEA-Proxy-Policy" and value "domain". This header is stripped from the request while sending, but the effect is to ignore the configured outbound proxy. The X-BEA-Proxy-Policy custom header can be used by applications to override the configured policy on a request-by-request basis. The value of the header can be "domain" or "proxy". Note, however, that if the policy is overridden to “proxy,” the configuration must still have the outbound proxy URIs in order to route to the outbound proxy.

Table 1-2 Nested proxy Elements

Element	Description
`routing-policy`	An optional element that configures the behavior of the proxy. Valid values are: • **domain** - Proxies messages using the routing rule defined by RFC 3261, ignoring any outbound proxy that is specified. • **proxy** - Sends the message to the downstream proxy specified in the default proxy URI. If there are multiple proxy specifications they are tried in the order in which they are specified. However, if the transport tries a UDP proxy, the settings for subsequent proxies are ignored.
`uri`	The TCP or UDP URI of the proxy server. You must specify at least one URI for a `proxy` element. Place multiple URIs in multiple `uri` elements within the `proxy` element.

Listing 1-2 shows the default proxy configuration for WebLogic SIP Server domains. The request in this case is created in accordance with the SIP routing rules, and finally the request is sent to the outbound proxy “sipoutbound.bea.com”.

Listing 1-2 Sample proxy Definition

```
<proxy>
     <routing-policy>proxy</routing-policy>
     <uri>sip:sipoutbound.bea.com:5060</uri>
     <!-- Other proxy uri tags can be added. -->
</proxy>
```

t1-timeout-interval

This element sets the value of the SIP protocol T1 timer, in milliseconds. Timer T1 also specifies the initial values of Timers A, E, and G, which control the retransmit interval for INVITE requests and responses over UDP.

Timer T1 also affects the values of timers F, H, and J, which control retransmit intervals for INVITE responses and requests; these timers are set to a value of 64*T1 milliseconds. See the *SIP: Session Initiation Protocol* for more information about SIP timers. See also Configuring NTP for Accurate SIP Timers in the *Configuration Guide*.

If `t1-timeout-interval` is not configured, WebLogic SIP Server uses the SIP protocol default value of 500 milliseconds.

t2-timeout-interval

This elements sets the value of the SIP protocol T2 timer, in milliseconds. Timer T2 defines the retransmit interval for INVITE responses and non-INVITE requests. See the *SIP: Session Initiation Protocol* for more information about SIP timers. See also Configuring NTP for Accurate SIP Timers in the *Configuration Guide*.

If `t2-timeout-interval` is not configured, WebLogic SIP Server uses the SIP protocol default value of 4 seconds.

t4-timeout-interval

This elements sets the value of the SIP protocol T4 timer, in milliseconds. Timer T4 specifies the maximum length of time that a message remains in the network. Timer T4 also specifies the initial values of Timers I and K, which control the wait times for retransmitting ACKs and responses over UDP. See the *SIP: Session Initiation Protocol* for more information about SIP timers. See also Configuring NTP for Accurate SIP Timers in the *Configuration Guide*.

If `t4-timeout-interval` is not configured, WebLogic SIP Server uses the SIP protocol default value of 5 seconds.

timer-b-timeout-interval

This elements sets the value of the SIP protocol Timer B, in milliseconds. Timer B specifies the length of time a client transaction attempts to retry sending a request. See the *SIP: Session Initiation Protocol* specification for more information about SIP timers. See also Configuring NTP for Accurate SIP Timers in the *Configuration Guide*.

If `timer-b-timeout-interval` is not configured, the Timer B value is derived from timer T1 (64*T1, or 32000 milliseconds by default).

timer-f-timeout-interval

This elements sets the value of the SIP protocol Timer F, in milliseconds. Timer F specifies the timeout interval for retransmitting non-INVITE requests. See the *SIP: Session Initiation Protocol* specification for more information about SIP timers. See also Configuring NTP for Accurate SIP Timers in the *Configuration Guide*.

If `timer-f-timeout-interval` is not configured, the Timer F value is derived from timer T1 (64*T1, or 32000 milliseconds by default).

max-application-session-lifetime

This element sets the maximum amount of time, in minutes, that a SIP application session can exist before WebLogic SIP Server invalidates the session. `max-application-session-lifetime` acts as an upper bound for any timeout value specified using the `session-timeout` element in a `sip.xml` file, or using the `setExpires` API.

A value of -1 (the default) specifies that there is no upper bound to application-configured timeout values.

enable-local-dispatch

`enable-local-dispatch` is a server optimization that helps avoid unnecessary network traffic when sending and forwarding messages. You enable the optimization by setting this element "true." When `enable-local-dispatch` enabled, if a server instance needs to send or forward a message and the message destination is the engine tier's cluster address or the local server address, then the message is routed internally to the local server instead of being sent via the network. Using this optimization can dramatically improve performance when chained applications process the same request as described in Composing SIP Applications in *Developing Applications with WebLogic SIP Server*.

You may want to disable this optimization if you feel that routing internal messages could skew the load on servers in the engine tier, and you prefer to route all requests via a configured load balancer.

By default `enable-local-dispatch` is set to "false."

cluster-loadbalancer-map

The `cluster-loadbalancer-map` element is used only when upgrading WebLogic SIP Server software, or when upgrading a production SIP Servlet to a new version. It is not required or used during normal server operations.

During a software upgrade, multiple engine tier clusters are defined to host the older and newer software versions. `A cluster-loadbalancer-map` defines the virtual IP address (defined on your load balancer) that correspond to an engine tier cluster configured for an upgrade. WebLogic SIP Server uses this mapping to ensure that engine tier requests for timers and call state data are received from the correct "version" of the cluster. If a request comes from an incorrect version of the software, the `cluster-loadbalancer-map` entries are used to forward the request to the correct cluster.

Each `cluster-loadbalancer-map` entry contains the two elements described in

Table 1-3 Nested cluster-loadbalancer-map Elements

Element	Description
`cluster-name`	The configured name of an engine tier cluster.
`sip-uri`	The internal SIP URI that maps to the engine tier cluster. This corresponds to a virtual IP address that you have configured in your load balancer. The internal URI is used to forward requests to the correct cluster version during an upgrade.

Listing 1-3 shows a sample `cluster-loadbalancer-map` entry used during an upgrade.

Listing 1-3 Sample cluster-loadbalancer-map Entry

```
<cluster-loadbalancer-map>
   <cluster-name>EngineCluster</cluster-name>
   <sip-uri>sip:172.17.0.1:5060</sip-uri>
</cluster-loadbalancer-map>
<cluster-loadbalancer-map>
   <cluster-name>EngineCluster2</cluster-name>
   <sip-uri>sip:172.17.0.2:5060</sip-uri>
```

```
</cluster-loadbalancer-map>
```

See Upgrading Software in the *Operations Guide* for more information.

default-behavior

This element defines the default behavior of the WebLogic SIP Server instance if the server cannot match an incoming SIP request to a deployed SIP Servlet (or if the matching application has been invalidated or timed out). Valid values are:

- `proxy`—Act as a proxy server.
- `ua`—Act as a User Agent.

`proxy` is used as the default if you do not specify a value.

When acting as a User Agent (UA), WebLogic SIP Server acts in the following way in response to SIP requests:

- ACK requests are discarded without notice.
- CANCEL or BYE requests receive response code 481 - Transaction does not exist.
- All other requests receive response code 500 - Internal server error.

When acting as a proxy requests are automatically forwarded to an outbound proxy (see "proxy—Setting Up an Outbound Proxy Server" on page 1-9) if one is configured. If no proxy is defined, Weblogic SIP Server proxies to a specified Request URI only if the Request URI does not match the IP and port number of a known local address for a SIP Servlet container, or a load balancer address configured for the server. This ensures that the request does not constantly loop to the same servers. When the Request URI matches a local container address or load balancer address, WebLogic SIP Server instead acts as a UA.

default-servlet-name

This element specifies the name of a default SIP Servlet to call if an incoming initial request cannot be matched to a deployed Servlet (using standard `servlet-mapping` definitions in `sip.xml`). The name specified in the `default-servlet-name` element must match the `servlet-name` value of a deployed SIP Servlet. For example:

```
<default-servlet-name>myServlet</default-servlet-name>
```

If the name defined in `default-servlet-name` does not match a deployed Servlet, or no value is supplied (the default configuration), WebLogic SIP Server registers the name com.bea.wcp.sip.engine.BlankServlet as the default Servlet. The BlankServlet name is also used if a deployed Servlet registered as the default-servlet-name is undeployed from the container.

BlankServlet's behavior is configured with the default-behavior element. By default the Servlet proxies all unmatched requests. However, if the default-behavior element is set to "ua" mode, BlankServlet is responsible for returning 481 responses for CANCEL and BYE requests, and 500/416 responses in all other cases. BlankServlet does not respond to ACK, and it always invalidates the application session.

retry-after-value

Specifies the number of seconds used in the Retry-After header for 5xx responses. This value can also include a parameter or a reason code, such as "Retry-After: 18000;duration=3600" or "Retry-After: 120 (I'm in a meeting)."

If the this value is not configured, WebLogic SIP Server uses the default value of 180 seconds.

sip-security

WebLogic SIP Server enables you to configure one or more trusted hosts for which authentication is not performed. When WebLogic SIP Server receives a SIP message, it calls getRemoteAddress() on the SIP Servlet message. If this address matches an address defined in the server's trusted host list, no further authentication is performed for the message.

The sip-security element defines one or more trusted hosts, for which authentication is not performed. The sip-security element contains one or more trusted-authentication-host or trusted-charging-host elements, each of which contains a trusted host definition. A trusted host definition can consist of an IP address (with or without wildcard placeholders) or a DNS name. Listing 1-4 shows a sample sip-security configuration.

Listing 1-4 Sample Trusted Host Configuration

```
<sip-security>
   <trusted-authentication-host>myhost1.mycompany.com</trusted-authenticat
ion-host>
   <trusted-authentication-host>172.*</trusted-authentication-host>
</sip-security>
```

route-header

3GPP TS 24.229 Version 7.0.0 requires that IMS Application Servers generating new requests (for example, as a B2BUA) include the S-CSCF route header. In WebLogic SIP Server, the S-CSCF route header must be statically defined as the value of the `route-header` element in `sipserver.xml`. For example:

```
<route-header>
   <uri>Route: sip:wlss1.bea.com</uri>
</route-header>
```

engine-call-state-cache-enabled

WebLogic SIP Server provides the option for engine tier servers to cache a portion of the call state data locally, as well as in the data tier, to improve performance with SIP-aware load balancers. When a local cache is used, an engine tier server first checks its local cache for existing call state data. If the cache contains the required data, and the local copy of the data is up-to-date (compared to the data tier copy), the engine locks the call state in the data tier but reads directly from its cache.

By default the engine tier cache is disabled. To enable caching, set `engine-call-state-cache-enabled` to true:

```
<engine-call-state-cache-enabled>true</engine-call-state-cache-enabled>
```

See Enabling the Engine Tier Cache in the *Configuration Guide* for more information.

server-header

WebLogic SIP Server enables you to control when a Server header is inserted into SIP messages. You can use this functionality to limit or eliminate Server headers to reduce the message size for wireless networks, or to increase security.

By default, WebLogic SIP Server inserts no Server header into SIP messages. Set the `server-header` to one of the following string values to configure this behavior:

- `none` (the default) inserts no Server header.
- `request` inserts the Server header only for SIP requests generated by the server.
- `response` inserts the Server header only for SIP responses generated by the server.
- `all` inserts the Server header for all SIP requests and responses.

For example, the following element configures WebLogic SIP Server to insert a Server header for all generated SIP messages:

```
<server-header>all</server-header>
```

See also "server-header-value" on page 1-17.

server-header-value

WebLogic SIP Server enables you to control the text that is inserted into the Server header of generated messages. This provides additional control over the size of SIP messages and also enables you to mask the server entity for security purposes. By default, WebLogic SIP Server does not insert a Server header into generated SIP messages (see "server-header" on page 1-16). If Server header insertion is enabled but no `server-header-value` is specified, WebLogic SIP Server inserts the value "WebLogic SIP Server." To configure the header contents, enter a string value. For example:

```
<server-header-value>MyCompany Application Server</server-header-value>
```

persistence

The `persistence` element defines enables or disables writing call state data to an RDBMS and/or to a remote, geographically-redundant WebLogic SIP Server installation. For sites that utilize geographically-redundant replication features, the `persistence` element also defines the site ID and the URL at which to persist call state data.

The `persistence` element contains the sub-elements described in Table 1-4.

Table 1-4 Nested persistence Elements

Element	Description
default-handling	Determines whether or not WebLogic SIP Server observes persistence hints for RDBMS persistence and/or geographical-redundancy. This element can have one of the following values: • **all**—Specifies that call state data may be persisted to both an RDBMS store and to a geographically-redundant WebLogic SIP Server installation. This is the default bevavior. Note that actual replication to either destination also requires that the available resources (JDBC datasource and remote JMS queue) are available. • **db**—Specifies that long-lived call state data is replicated to an RDBMS if the required JDBC datasource and schema are available. • **geo**—Specifies that call state data is persisted to a remote, geographically-redundant site if the configured site URL contains the necessary JMS resources. • **none**—Specifies that only in-memory replication is performed to other replicas in the data tier cluster. Call state data is not persisted in an RDBMS or to an external site.
geo-site-id	Specifies the site ID of this installation. All installations that participate in geographically-redundant replication require a unique site ID.
geo-remote-t3-url	Specifies the remote Weblogic SIP Server installation to which this site replicates call state data. You can specify a single URL corresponding to the engine tier cluster of the remote installation. You can also specify a comma-separated list of addresses corresponding to each engine tier server. The URLs must specify the t3 protocol.

Listing 1-5 shows a sample configuration that uses RDBMS storage for long-lived call state as well as geographically-redundant replication. Call states are replicated to two engine tier servers in a remote location.

Listing 1-5 Sample persistence Configuration

```
<persistence>
```

```
  <default-handling>all</default-handling>
  <geo-site-id>1</geo-site-id>
  <geo-remote-t3-url>t3://remoteEngine1:7050,t3://remoteEngine2:7051</geo-
remote-t3-url>
</persistence>
```

See Storing Long-Lived Call State Data in an RDBMS and Configuring Geographically-Redundant Installations in *Configuring WebLogic SIP Server* for more information.

use-header-form

This element configures the server-wide, default behavior for using or preserving compact headers in SIP messages. You can set this element to one of the following values:

- `compact`—WebLogic SIP Server uses the compact form for all system-generated headers. However, any headers that are copied from an originating message (rather than generated) use their original form.
- `force compact`—WebLogic SIP Server uses the compact form for all headers, converting long headers in existing messages into compact headers as necessary.
- `long`—WebLogic SIP Server uses the long form for all system-generated headers. However, any headers that are copied from an originating message (rather than generated) use their original form.
- `force long`—WebLogic SIP Server uses the long form for all headers, converting compact headers in existing messages into long headers as necessary.

enable-dns-srv-lookup

This element enables or disables WebLogic SIP Server DNS lookup capabilities. If you set the element to "true," then 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.

By default, DNS resolution is not used (“false”).

Note: 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.

connection-reuse-pool

WebLogic SIP Server includes a connection pooling mechanism that can be used to minimize communication overhead with a Session Border Control (SBC) function or Serving Call Session Control Function (S-CSCF). You can configure multiple, fixed pools of connections to different addresses.

WebLogic SIP Server opens new connections from the connection pool on demand as the server makes requests to a configured address. The server then multiplexes new SIP requests to the address using the already-opened connections, rather than repeatedly terminating and recreating new connections. Opened connections are re-used in a round-robin fashion. Opened connections remain open until they are explicitly closed by the remote address.

Note that connection re-use pools are not used for incoming requests from a configured address.

To configure a connection re-use pool, you define the four nested elements described in Table 1-5.

Table 1-5 Nested connection-reuse-pool Elements

Element	Description
`pool-name`	A String value that identifies the name of this pool. All configured `pool-name` elements must be unique to the domain.
`destination`	Specifies the IP address or host name of the destination SBC or S-CSCF. WebLogic SIP Server opens or re-uses connection in this pool only when making requests to the configured address.

Table 1-5 Nested connection-reuse-pool Elements

Element	Description
`destination-port`	Specifies the port number of the destination SBC or S-CSCF.
`maximum-connections`	Specifies the maximum number of opened connections to maintain in this pool.

Listing 1-6 shows a sample `connection-reuse-pool` configuration having two pools.

Listing 1-6 Sample connection-reuse-pool Configuration

```
<connection-reuse-pool>
   <pool-name>SBCPool</pool-name>
   <destination>MySBC</destination>
   <destination-port>7070</destination-port>
   <maximum-connections>10</maximum-connections>
</connection-reuse-pool>
<connection-reuse-pool>
   <pool-name>SCSFPool</pool-name>
   <destination>192.168.1.6</destination>
   <destination-port>7071</destination-port>
   <maximum-connections>10</maximum-connections>
</connection-reuse-pool>
```

globally-routable-uri

This element enables you to specify a Globally-Routable User Agent URI (GRUU) that WebLogic SIP Server automatically inserts into Contact and Route-Set headers when communicating with network elements. The URI specified in this element should be the GRUU for the entire WebLogic SIP Server cluster. (In a single-server domain, use a GRUU for the server itself.)

Note that User Agents (UAs) deployed on WebLogic SIP Server typically obtain GRUUs via a registration request. In this case, the application code is responsible both for requesting and subsequently handling the GRUU. To request a GRUU the UA would include the “+sip.instance” Contact header field param in each Contact for which GRUU is required. Upon receiving a GRUU, the UA would use the GRUU as the URI for the contact header field when generating new requests.

CHAPTER 2

Data Tier Configuration Reference (datatier.xml)

The following sections provide a complete reference to the data tier configuration file, `datatier.xml`:

Overview of datatier.xml

The `datatier.xml` configuration file identifies servers that manage the concurrent call state for SIP applications, and defines how those servers are arranged into data tier *partitions*. A *partition* refers to one or more data tier server instances that manage the same portion of the call state. Multiple servers in the same partition are referred to as *replicas* because they all manage a copy of the same portion of the call state.

`datatier.xml` is stored in the *`DOMAIN_DIR`*`/config/custom` subdirectory where ***`DOMAIN_DIR`*** is the root directory of the WebLogic SIP Server domain.

Editing datatier.xml

You can edit `datatier.xml` using either the Administration Console or a text editor. Note that changes to the data tier configuration cannot be applied to servers dynamically; you must restart servers in order to change data tier membership or reconfigure partitions.

XML Schema

This schema file is available at `http://www.bea.com/ns/wlcp/wlss/300`. The schema is also also bundled within the `wlss-descriptor-binding.jar` library, installed in the *`WL_HOME`*`/server/lib/wlss` directory.

Example datatier.xml File

Listing 2-1 shows the template `datatier.xml` file created using the Configuration Wizard. See also Example Data Tier Configurations and Configuration Files in the *Configuration Guide*.

Listing 2-1 Default datatier.xml File

```
<st:data-tier
xmlns:st="http://bea.com/wcp/sip/management/internal/webapp">
 <st:partition>
   <st:name>partition-0</st:name>
   <st:server-name>replica1</st:server-name>
   <st:server-name>replica2</st:server-name>
 </st:partition>
</st:data-tier>
```

XML Element Description

`datatier.xml` contains one or more `partition` elements that define servers' membership in a data tier partition. All data tier clusters must have at least one `partition`.' Each partition contains the XML elements described in Table 2-1.

Table 2-1 Nested partition Elements

Element	Description
`name`	A String value that identifies the name of the partition. BEA recommends including the number of the partition (starting at 0) in the text of the name for administrative purposes. For example, "partition-0."
`server-name`	Specifies the name of a WebLogic SIP Server instance that manages call state in this partition. You can define up two three servers per `partition` element. Multiple servers in the same partition maintain the same call state data, and are referred to as *replicas*. BEA recommends including the number of the server (starting with 0) and the number of the partition in the server name for administrative purposes. For example, "replica-0-0."

CHAPTER 3

Diameter Configuration Reference (diameter.xml)

The following sections provide a complete reference to the Diameter configuration file, `diameter.xml`:

Overview of diameter.xml

The `diameter.xml` file configures attributes of a Diameter node, such as:

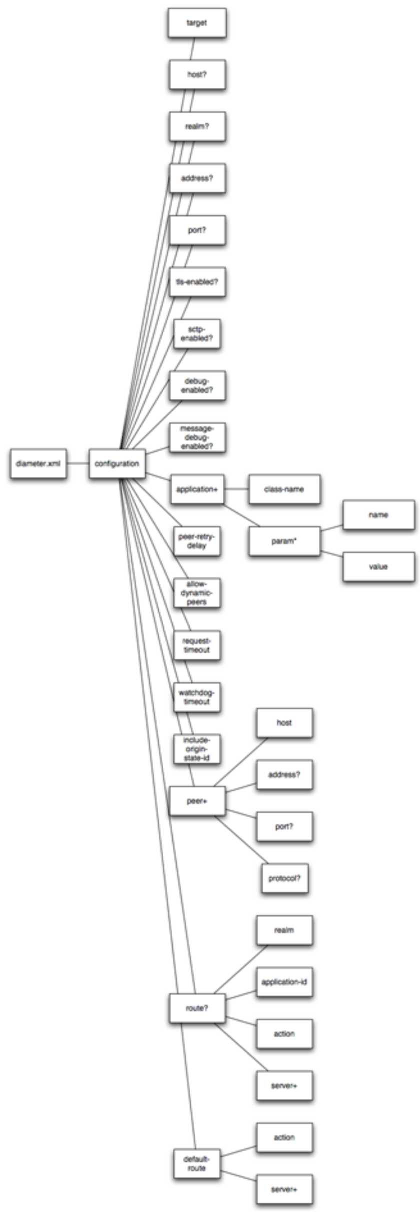
- The host identity of the Diameter node
- The Diameter applications that are deployed on the node
- Connection information for Diameter peer nodes
- Routing information and default routes for handling Diameter messages.

The Diameter protocol implementation reads the configuration file at boot time. `diameter.xml` is stored in the *`DOMAIN_DIR`*`/config/custom` subdirectory where *`DOMAIN_DIR`* is the root directory of the WebLogic SIP Server domain.

Graphical Representation

Figure 3-1 shows the element hierarchy of the `diameter.xml` file.

Figure 3-1 Element Hierarchy of diameter.xml



Editing diameter.xml

You should never move, modify, or delete the `diameter.xml` file during normal operations.

BEA recommends using the Administration Console to modify `diameter.xml` indirectly, rather than editing the file by hand. Using the Administration Console ensures that the `diameter.xml` document always contains valid XML.

You may need to manually view or edit `diameter.xml` to troubleshoot problem configurations, repair corrupted files, or to roll out custom Diameter node configurations to a large number of machines when installing or upgrading WebLogic SIP Server. When you manually edit `diameter.xml`, you must reboot Diameter nodes to apply your changes.

WARNING: Always use the Diameter node in the Administration Console or the WLST utility, as described in Configuring Engine Tier Container Properties in the *Configuration Guide* to make changes to a running WebLogic SIP Server deployment.

Steps for Editing diameter.xml

If you need to modify `diameter.xml` on a production system, follow these steps:

1. Use a text editor to open the *`DOMAIN_DIR`*`/config/custom/diameter.xml` file, where *`DOMAIN_DIR`* is the root directory of the WebLogic SIP Server domain.
2. Modify the `diameter.xml` file as necessary. See "XML Element Description" on page 3-5 for a full description of the XML elements.
3. Save your changes and exit the text editor.
4. Reboot or start servers to have your changes take effect:

WARNING: Always use the Diameter node in the Administration Console or the WLST utility, as described in Configuring Engine Tier Container Properties in the *Configuration Guide*, to make changes to a running WebLogic SIP Server deployment.

5. Test the updated system to validate the configuration.

XML Schema

The full schema for the `diameter.xml` file is available at http://www.bea.com/ns/wlcp/diameter.xsd. The schema for the `applications` element is available at http://www.bea.com/ns/wlcp/diameter_app.xsd.

Both schema files are also bundled within the `wlssdiameter.jar` library, installed in the *`WL_HOME`*`/server/lib/wlss` directory.

Example diameter.xml File

See Configuring Diameter Sh Client Nodes and Relay Agents in *Configuring Network Resources* for multiple listings of example `diameter.xml` configuration files.

XML Element Description

The following sections describe each XML element in `diameter.xml`.

configuration

The top level `configuration` element contains the entire diameter node configuration.

target

Specifies one or more target WebLogic SIP Server instances to which the node configuration is applied. The target servers must be defined in the `config.xml` file for your domain.

host?

Specifies the host identity for this Diameter node. If no `host` element is specified, the identity is taken from the local server's host name. Note that the host identity may or may not match the DNS name.

Note: 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?

Specifies the realm name for which this Diameter node has responsibility. You can run multiple Diameter nodes on a single host using different realms and listen port numbers. The 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.

If you omit the `realm` element, the realm named is derived using the domain name portion of the host name, if the host name is fully-qualified (for example, host@bea.com).

address?

Specifies the listen address for this Diameter node, using either the DNS name or IP address. If you do not specify an address, the node uses 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` element with an explicit DNS name or IP address to avoid configuration errors.

port?

Specifies the TCP or TLS listen port for this Diameter node. The default port is 3868.

tls-enabled?

This element is used only for standalone node operation to advertise TLS capabilities.

WebLogic SIP Server ignores the `tls-enabled` element for nodes running within a server instance. Instead, TLS transport is reported as enabled if the server instance has configured a Network Channel having TLS support (a diameters channel). See Creating Network Channels for the Diameter Protocol in *Configuring Network Resources*.

sctp-enabled?

This element is used only for standalone node operation to advertise SCTP capabilities.

WebLogic SIP Server ignores the `sctp-enabled` element for nodes running within a server instance. Instead, SCTP transport is reported as enabled if the server instance has configured a Network Channel having SCTP support (a diameter-sctp channel). See Creating Network Channels for the Diameter Protocol in *Configuring Network Resources*.

debug-enabled?

Specifies a boolean value to enable or disable debug message output. Debug messages are disabled by default.

message-debug-enabled?

Specifies a boolean value to enable or disable tracing of Diameter messages. This element is disabled by default.

application

Configures a particular Diameter application to run on the selected node. WebLogic SIP Server includes applications to support nodes that act as Diameter Sh, Ro, and Rf clients, Diameter relay agents, or Home Subscriber Servers (HSS). Note that the HSS application is a simulator that is provided only for development or testing purposes.

class-name

Specifies the application class file to load.

param*

Specifies one or more optional parameters to pass to the application class.

name

Specifies the name of the application parameter.

value

Specifies the value of the parameter.

peer-retry-delay?

Specifies the number of seconds this node waits between retries to Diameter peers. The default value is 30 seconds.

allow-dynamic-peers?

Specifies a boolean value that enables or disables dynamic peer configuration. 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.

request-timeout

Specifies the number of milliseconds to wait for an answer from a peer before timing out.

watchdog-timeout

Specifies the number of seconds used for the Diameter Tw watchdog timer.

include-origin-state

Specifies whether the node should include the origin state AVP in requests and answers.

peer+

Specifies connection information for an individual Diameter peer. You can choose to configure connection information for individual peer nodes, or allow any node to be dynamically added as a peer. BEA recommends using dynamic peers only if you are using the TLS transport, because there is no way to filter or restrict hosts from becoming peers when dynamic peers are enabled.

When configuring Sh client nodes, the `peers` element should contain peer definitions for each Diameter relay agent deployed to your system. If your system does not use relay agents, you must include a peer entry for the Home Subscriber Server (HSS) in the system, as well as for all other engine tier nodes that act as Sh client nodes.

When configuring Diameter relay agent nodes, the `peers` element should contain peer entries for all Diameter client nodes that access the peer, as well as the HSS.

host

Specifies the host identity for a Diameter peer.

address?

Specifies the listen address for a Diameter peer. If you do not specify an address, the host identity is used.

port?

Specifies the TCP or TLS port number for this Diameter peer. The default port is 3868.

protocol?

Specifies the protocol used by the peer. This element may be one of `tcp` or `sctp`.

route?

Defines a realm-based route that this node uses when resolving messages.

When configuring Sh client nodes, you should specify a route to each Diameter relay agent node deployed in the system, as well as a `default-route` to a selected relay. If your system does not use relay agents, simply configure a single `default-route` to the HSS.

When configuring Diameter relay agent nodes, specify a single `default-route` to the HSS.

realm

The target realm used by this route.

application-id

The target application ID for the route.

action

An action type that describes the role of the Diameter node when using this route. The value of this element can be one of the following:

- none
- local
- relay
- proxy
- redirect

server+

Specifies one or more target servers for this route. Note that any server specified in the `server` element must also be defined as a `peer` to this Diameter node, or dynamic peer support must be enabled.

default-route?

Defines a default route to use when a request cannot be matched to a configured route.

action

Specifies the default routing action for the Diameter node. See “action” on page 3-9.

server+

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.

CHAPTER 4

Profile Service Provider Configuration Reference (profile.xml)

The following sections provide a complete reference to the profile provider configuration file, `profile.xml`:

Overview of profile.xml

The `profile.xml` file configures attributes of a profile service provider, such as:

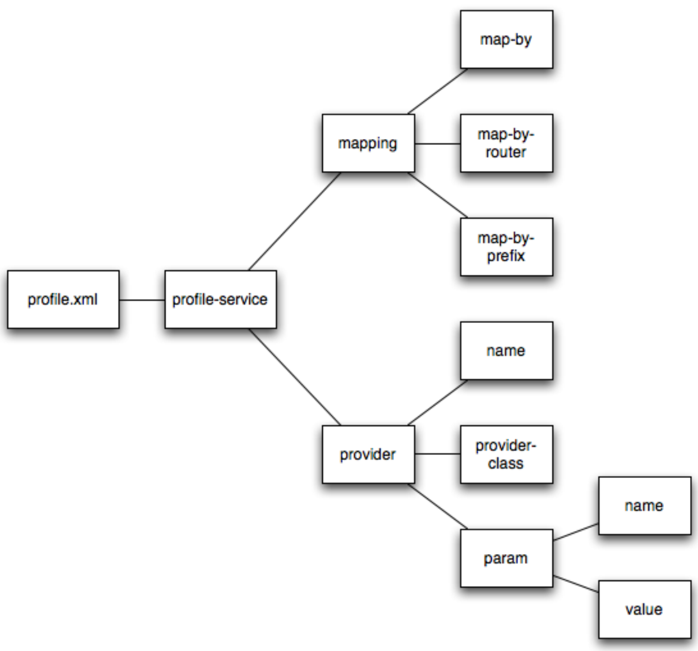
- The name of the provider
- The class name of the provider implementation
- Optional arguments passed to the provider
- Mapping rules for using the provider.

`profile.xml` is stored in the *`DOMAIN_DIR`*`/config/custom` subdirectory where ***`DOMAIN_DIR`*** is the root directory of the WebLogic SIP Server domain.

Graphical Representation

Figure 4-1 shows the element hierarchy of the `profile.xml` file.

Figure 4-1 Element Hierarchy of profile.xml



Editing profile.xml

BEA recommends using the Administration Console profile service extension to modify `profile.xml` indirectly, rather than editing the file by hand. Using the Administration Console ensures that the `profile.xml` document always contains valid XML. See Configuring Profile Providers Using the Administration Console in Developing Applications with WebLogic SIP Server.

You may need to manually view or edit `profile.xml` to troubleshoot problem configurations, repair corrupted files, or to roll out custom profile provider configurations to a large number of machines when installing or upgrading WebLogic SIP Server. When you manually edit `profile.xml`, you must reboot servers to apply your changes.

Steps for Editing profile.xml

If you need to modify `profile.xml` on a production system, follow these steps:

1. Use a text editor to open the *`DOMAIN_DIR`*`/config/custom/profile.xml` file, where *`DOMAIN_DIR`* is the root directory of the WebLogic SIP Server domain.
2. Modify the `profile.xml` file as necessary. See "XML Element Description" on page 4-3 for a full description of the XML elements.
3. Save your changes and exit the text editor.
4. Reboot or start servers to have your changes take effect:
5. Test the updated system to validate the configuration.

XML Schema

The full schema for the `profile.xml` file is available at http://www.bea.com/ns/wlcp/wlss/profile/300. The schema file is also bundled within the `profile-service-descriptor-binding.jar` library, installed in the *`WL_HOME`*`/server/lib/wlss` directory.

Example profile.xml File

See Developing Custom Profile Providers in *Developing Applications with WebLogic SIP Server* for sample listings of `profile.xml` configuration files.

XML Element Description

The following sections describe each XML element in `profile.xml`.

profile-service

The top level `profile-service` element contains the entire profile service configuration.

mapping

Specifies how requests for profile data are mapped to profile provider implementations.

map-by

Specifies the technique used for mapping documents to providers:

- `router` uses a custom router class, specified by `map-by-router`, to determine the provider.
- `prefix` uses the specified `map-by-prefix` entry to map documents to a provider.
- `provider-name` uses the specified `name` element in the `provider` entry to map documents to a provider.

map-by-prefix

Specifies the prefix used to map documents to profile providers when mapping by prefix.

map-by-router

Specifies the router class (implementing `com.bea.wcp.profile.ProfileRouter`) used to map documents to profile providers with router-based mapping.

provider

Configures the profile provider implementation and startup options.

name

Specifies a name for the provider configuration. The `name` element is also used for mapping documents to the provider if you specify the `provider-name` mapping technique.

provider-class

Specifies the profile provider class (implementing `com.bea.wcp.profile.ProfileServiceSpi`).

param

Uses the `name` and `value` elements to specify optional parameters to the provider implementation.

CHAPTER 5

WebLogic SIP Server Startup Command Options

Table 5-1 provides a reference to the startup configuration options available to WebLogic SIP Server and other WebLogic SIP Server utilities.

Table 5-1 Startup Command Options

Application	Startup Option Link
Installer	-Djava.io.tmpdir
WebLogic SIP Server	-Dwlss.udp.listen.on.ephemeral -Dwlss.udp.lb.masquerade -Dweblogic.management.discover -Dweblogic.RootDirectory -DWLSS.SNMPAgentPort
WlssEchoServer	-Dwlss.ha.echoserver.port -Dwlss.ha.echoserver.logfile -Dreplica.host.monitor.enabled -Dwlss.ha.heartbeat.interval -Dwlss.ha.heartbeat.count -Dwlss.ha.heartbeat.SoTimeout
	See also the options for WebLogic Server utilities in the WebLogic Server 9.2 Documentation.