BEA WebLogic SIP Server®

Release Notes

Version 3.1
Revised: July 16, 2007

Contents

CHAPTER 1

WebLogic SIP Server 3.1 Features and Changes

Welcome to BEA WebLogic SIP Server 3.1! WebLogic SIP Server™ integrates SIP Servlet technologies with J2EE 1.4 and 1.3 and other leading Internet standards to provide a reliable framework for developing highly available, scalable, and secure telecommunications applications. WebLogic SIP Server's seamless integration of disparate, heterogeneous platforms and applications enables your network to leverage existing software investments and share the enterprise-class services and data that are crucial to building next-generation telephony applications.

The following sections describe the new features and changes made in the WebLogic Server 3.1 general release and in intermediate releases:

What's New in WebLogic SIP Server 3.1?

This section describes new features and functionality introduced in WebLogic SIP Server 3.1.

Integration with WebLogic Diagnostic Framework

WebLogic SIP Server now integrates with the Weblogic Diagnostic Framework (WLDF) to provide improved data collection and logging, watches and notifications, diagnostic image capture, and code instrumentation. See Using the WebLogic Diagnostic Framework (WLDF) in the *Operations Guide*.

Symmetric Response Routing (RFC 3581 rport parameter)

WebLogic SIP Server 3.1 honors the `rport` parameter described in RFC 3581 for symmetric response routing. When a message is received that has the `rport` parameter, the server responds using the remote UDP port number from which the message was received, rather than the port number specified in the `Via` header. You can also configure WebLogic SIP Server to automatically add the `rport` parameter to `Via` headers when acting as a UAC. See enable-rport in the *Configuration Reference Manual*.

Domain Aliases

WebLogic SIP Server 3.1 now enables you to configure the exact domain(s) for which the server is responsible. Domain alias configuration avoids potential proxy problems and clarifies the domain handling support for a given server installation. See domain-alias-name in the *Configuration Reference Manual*.

Additional Monitoring for SIP and Diameter Network Channels

You can now monitor the behavior of WebLogic SIP Server network channels (`sip`, `sips`, `diameter`, `diameters`, and `diameter-sctp` channels) using the Administration Console. To monitor SIP and Diameter channels:

1. Access the Administration Console for your domain.
2. Select the Environment->Servers node.
3. Select the name of a server to monitor.
4. Select the Monitoring->Channels tab.

WebLogic SIP Server channels display statistics only for the Connections, Messages Received, Messages Sent, Bytes Received, and Bytes Sent attributes.

Configurable Handling of Stale Session Data

WebLogic SIP Server uses encoded URIs to identify the call states and application sessions associated with a message. When an application is undeployed or upgraded to a new version, incoming requests may have encoded URIs that specify "stale" or nonexistent call or session IDs. In WebLogic SIP Server 3.1, you can configure the action that the server takes when it encounters stale session data in a request. See stale-session-handling in the *Configuration Reference Manual*.

List of SIP Headers and Parameters Added by WebLogic SIP Server

WebLogic SIP Server may add one or more SIP headers and parameters to existing SIP messages in order to support various features. You must ensure that all network functions allow these headers and parameters to pass unchanged to SIP Server instances. Alternately, Session Border Control functions may archive and restore this information as necessary.

Table 1-1 and Table 1-2 describe the information that WebLogic SIP Server may add to SIP messages.

Table 1-1 WebLogic SIP Server Headers

Header Name	Description
`X-BEA-Proxy-Policy`	Determines the proxy policy used for sending certain requests.
`X-Cluster-Info`	Provides failover hints to the load balancer.
`X-WLSS-Sdways-O-C`	Used by the sideways forwarding mechanism to deliver messages to a compatible cluster during upgrade.
`X-WLSS-Sdways-Req-Cert`	Used by the sideways forwarding mechanism to deliver messages to a compatible cluster during upgrade.
`X-WLSS-Sdways-Resp-Cert`	Used by the sideways forwarding mechanism to deliver messages to a compatible cluster during upgrade.
`X-WLSS-Sdways-R-C`	Used by the sideways forwarding mechanism to deliver messages to a compatible cluster during upgrade.

Table 1-2 WebLogic SIP Server Parameters

Parameter Name	Description
`apsessionid`	Used with the `SipApplicationSession.encodeURI` method to store the session ID.
`cluster`	Provides failover hints to the load balancer.
`wlsscid`	Identifies the cluster ID of the cluster that originated the SIP message during a software upgrade. The sideways forwarding mechanism uses this attribute to ensure that messages are delivered to a compatible cluster.
`wlssladdr`	(New in Version 3.1.) Records the local address on which an incoming request was received by a stateless proxy. This header is required for handling the stateless proxy use case with rport support.
`wlsslport`	(New in Version 3.1.) Records the local port on which an incoming request was received by a stateless proxy. This header is required for handling the stateless proxy use case with rport support.
`wlssrrd`	Records the incoming and outgoing interfaces used in a multihomed configuration.

What's New in WebLogic SIP Server 3.0?

This section describes new features and functionality introduced in WebLogic SIP Server 3.0.

Based on WebLogic Server 9.2

WebLogic SIP Server 3.0 is deployed on the core BEA WebLogic Server 9.2 product, which introduces many key features such as J2EE 1.4 compliance, improved systems management, and higher performance, scalability, and availability. See What's New in WebLogic Server 9.2 in the WebLogic Server 9.2 documentation.

Geographically-Redundant Persistence

WebLogic SIP Server can be installed in a geographically-redundant configuration for customers who have multiple, regional data centers, and require continuing operation even after a catastrophic site failure. The geographically-redundant configuration enables multiple Weblogic SIP Server installations (complete with engine and data tier clusters) to replicate call state transactions between one another. Administrators can then choose to redirect all network traffic to the secondary, replicated site to minimize lost calls if they determine that a regional site has failed. See Configuring Geographically- Redundant Installations in *Configuring and Managing WebLogic SIP Server*.

Diameter Base Protocol and IMS Ro, Rf Interface Support

In addition to the Diameter Sh protocol provider introduced in WebLogic SIP Server 2.2, version 3.0 includes new providers for the Ro and Rf protocols. The base Diameter protocol implementation is also now available for developers who want to implement additional Diameter applications. See the following links in *Developing Applications with WebLogic SIP Server* for more information:

- Using the Diameter Base Protocol API
- Using the Diameter Rf Interface Application for Offline Charging
- Using the Diameter Ro Interface Application for Online Charging

Engine Tier Caching

The engine tier now caches a portion of the SIP call state data available in data tier replicas. When used in combination with a SIP-aware load balancer, the cache increases the performance of accessing call state data. See Enabling the Engine Tier Cache in *Configuring and Managing WebLogic SIP Server*.

RDBMS Storage for Long-Lived Call State Data

WebLogic SIP Server 3.0 enables you to store long-lived call state data in an Oracle RDBMS in order to conserve RAM. The data tier persists a call state's data to the RDBMS after the call dialog has been established, and retrieves or deletes the persisted call state data as necessary to modify or remove the call state. BEA also provides an API for application designers to provide "hints" as to when the data tier should persist call state data. See Storing Call State Data in an RDBMS in *Configuring and Managing WebLogic SIP Server*.

Minimal Transactional Latency with JRockit Deterministic Garbage Collection

WebLogic SIP Server can be used with the JRockit deterministic garbage collector to greatly improve latency performance for SIP transactions. To use WebLogic SIP Server with this garbage collector, see Using JRockit Deterministic Garbage Collection in the *Configuration Guide*.

Production Upgrade for Converged SIP/HTTP Applications

WebLogic SIP Server 3.0 introduces application upgrade support for converged SIP/HTTP applications. Application upgrade support now closely models the upgrade support available in WebLogic Server 9.2, and provides for a SIP "administration channel" that can be used to securely testing applications in a production environment. See Upgrading Deployed SIP Applications in the *Operations Guide*.

Note: As part of the new upgrade functionality, `SipApplicationRuntimeMBean` is now deprecated for obtaining information about the application name and version string. Use `ApplicationRuntimeMBean` instead.

SCTP Support for Diameter

WebLogic SIP Server supports the SCTP transport protocol on certain operating systems for Diameter network traffic. See Configuring Diameter Client Nodes and Relay Agents in *Configuring Network Resources*.

DNS Support for Proxy Discovery and Response Routing

WebLogic SIP Server 3.0 now supports DNS for resolving the transport, IP address and port number of a proxy required to send a SIP message as described in RFC 3263. DNS may also used when routing responses in order to resolve the IP address and port number of a destination. Prior to version 3.0, DNS resolution had to be performed by the individual UA or proxy application.

See Enabling DNS Support in *Configuring Network Resources*.

IPv6 Support

WebLogic SIP Server supports IPv6 for external network interfaces as described in RFC 2460: Internet Protocol, Version 6 (IPv6) Specification. To use IPv6, your underlying operating system must support the protocol, and you must configure IPv6 network channels on all engine tier server nodes. See IPv4 and IPv6 in *Configuring Network Resources*.

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

Configurable Server Header

The Administrator can optionally configure the contents of the Server header that WebLogic SIP Server inserts into SIP message bodies. The entire header contents can be omitted to reduce the message size for wireless networks, or it can be set to an arbitrary string value. Prior to version 3.0, the header was always populated with the name and version of the WebLogic SIP Server instance. See server-header and server-header-value in the *Configuration Reference*.

Configuration of SIP Message Header Formats

WebLogic SIP Server provides flexible configuration parameters and APIs for controlling whether generated SIP messages use compact or long header forms. Header form rules can be set at three different levels:

- Container-level configuration: Set the default rules for using compacting headers using elements in the `sipserver.xml` file. See use-header-form in the *Configuration Reference*.
- Message-level API: The `WlssSipServletMessage` interface provides the `setUseHeaderForm` method to specify long or compact headers for a given SIP message. See Using Compact and Long Header Formats for SIP Messages in *Developing Applications.*
- Header-level API: The JSR 116 `SipServletMessage` interface provides the `setHeader` method to set a given header name a specific value. See the JSR 116 JavaDoc for `SipServletMessage.`

`WlssSipServletResponse.setUseHeaderForm` can be used in combination with `SipServletMessage.setHeader` and the container-level configuration to customize header formats. See Using Compact and Long Header Formats for SIP Messages in *Developing Applications* for information about how the different settings interact with one another.

Extended API for Resolving TelURLs

WebLogic SIP Server extends the `javax.servlet.sip.TelURL` interface with the `com.bea.wcp.sip.WlssTelURI` interface. The extended interface enables applications to resolve Tel URLs present in the user portion of a SIP URI. The API parses a Tel URL into a domain name using the standard suffix, `e164.arpa`, as described in RFC 3761. It then performs a DNS NAPTR record lookup to produce an ENUM NAPTR record set.

For example, for a Tel URL domain name of `4.3.2.1.5.5.5.5.1.4.1.e164.arpa`, the API performs a DNS lookup to retrieve an ENUM NAPTR record set similar to:

```
$ORIGIN 4.3.2.1.5.5.5.5.1.4.1.e164.arpa
    IN NAPTR 100 10 "u" "E2U+sip"    "!^.*$!sip:user@example.com!"
    IN NAPTR 100 20 "u" "E2U+mailto" "!^.*$!mailto:info@example.com!"
```

Methods in the `WlssTelURI` interface return either the full ENUM record set, an array of SIP URIs present in the record set, or only the highest-precedence SIP URI present in the record set. See `com.bea.wcp.sip.WlssTelURI` in the JavaDoc.

SAR File Deployment

WebLogic SIP Server 3.0 supports deployment of applications in SAR file format. The SAR file is similar in format to WAR files, and can contain deployment descriptor information for both HTTP and SIP Servlets. SAR files need not include a `weblogic.xml` deployment descriptor.

Extended Profile API

WebLogic SIP Server includes a public profile service API, `com.bea.wcp.sip.profile.API`, that you can use to create profile provider implementations. A profile provider performs the work of accessing XML documents from a data repository using a defined protocol. Deployed SIP Servlets and other applications need not understand the underlying protocol or the data repository in which the document is stored; they simply reference profile data using a custom URL using the provider API, and WebLogic SIP Server delegates the request processing to the correct provider. See Developing Custom Profile Providers in *Developing Applications with WebLogic SIP Server*.

Connection Pooling for Re-Use of TCP Connections

WebLogic SIP Server includes a new connection pooling mechanism to minimize unnecessary communication with a Session Border Control (SBC) function or Serving Call Session Control Function (S-CSCF). The server multiplexes a fixed pool of connections to a configured SBC or S-CSCF instead of repeatedly terminating and recreating connections during operation. See connection-reuse-pool in the *Configuration Reference*.

Support for Globally-Routable User Agent URIs (GRUU)

WebLogic SIP Server meets the requirements for obtaining and using Globally-Routable User Agent URIs (GRUU) as described in draft-ietf-sip-gruu-10: Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP).

To specify a GRUU for WebLogic SIP Server to use when acting as a network element, see globally-routable-uri in the *Configuration Reference*.

What’s New in WebLogic SIP Server 2.2?

This section describes new features and functionality introduced in WebLogic SIP Server 2.2.

New RFC Support

WebLogic SIP Server 2.2 now supports the following additional RFCs and specifications:

- RFC 2543: SIP: Session Initiation Protocol (v1) backward compatibility is supported.
- RFC 3262: Reliability of Provisional Responses in the Session Initiation Protocol (SIP) is supported.
- RFC 3311: The Session Initiation Protocol (SIP) UPDATE Method
- RFC 3327: Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts is supported.
- RFC 3515: The Session Initiation Protocol (SIP) Refer Method is supported.

See Standards Alignment in the *Technical Product Description* for a full description of RFC compliance.

Changes for 3GPP Application Server Compliance

WebLogic SIP Server 2.2 introduces a variety of changes in order to comply better with the IMS Application Server specifications described in 3GPP 24.229 Rel 7.0.0:

- The 3GPP specification states that applications that interact with a S-CSCF should add an `orig` parameter when creating new requests based on an identity specified in the top route header. WebLogic SIP Server provides a new `com.bea.wcp.sip.util.TransportUtil` class to help an application determine if a particular request was generated locally, so that the application can add the `orig` parameter if necessary. See the WebLogic SIP Server JavaDoc.

- The SIP container now supports messages having the `Request-Disposition`, `Accept-Contact`, and `Reject-Contact` headers, which can be used by callers to define preferences for how a message should be processed. See RFC 3840.
- When acting as a Proxy, WebLogic SIP Server 2.2 ensures that no more than one `P-Charging-Vector` or `P-Charging-Function-Address` header is present in a given message, and automatically strips any such header if a message is proxied to an untrusted host. See RFC 3455.
- WebLogic SIP Server 2.2 uses 3GPP security processing workflows for SIP and HTTP authentication.
- The `sipserver.xml` file includes a new element, `route-header` which you can use to statically define the S-CSCF route header to be added to new outgoing requests originated from the WebLogic SIP Server container. See route-header in the *Configuration Reference*.
- The UPDATE method described in RFC 3311 is now supported. See "Support for SIP UPDATE Method (RFC3311)" on page 1-10.
- An extended API is provided to help applications modify Contact header parameters. See "Support for Modifying Contact Header Parameters" on page 1-15.

Diameter Sh Interface, Relay Node Support, and Profile Service API

WebLogic SIP Server implements the Diameter Sh interface as a Web Application that you can deploy to the server. Sh interface functions are implemented as a provider that transparently generates and responds to the Diameter command codes defined in the Sh application specification. A higher-level profile service API enables SIP Servlets to manage user profile data as an XML document using XML Document Object Model (DOM). WebLogic SIP Server supports converged SIP and Diameter applications. See Using the Profile Service API (Diameter Sh Interface) in *Developing Applications with WebLogic SIP Server*.

WebLogic SIP Server also provides a Diameter relay node application, which you can configure for use when connecting to an HSS. An HSS simulator application is included for development or testing purposes. See Configuring Diameter Sh Client Nodes and Relay Agents in *Configuring Network Resources*.

Support for SIP UPDATE Method (RFC3311)

As described in RFC 3311, the SIP UPDATE method enables a UAC to update the parameters of a session, such as the media streams or codecs used for communication. To fully support the behavior described in RFC 3311, WebLogic SIP Server now automatically generates a 500 response with a `Retry-After` header value if a deployed B2BUA receives an UPDATE request, relinquishes control (returns control from the `service()` method), and then subsequently receives another UPDATE. This change was made to comply with section 5.2 of RFC 3311, which requires a UAS to return a 500 response and `Retry-After` header if a second UPDATE is received before making a final response to a prior UPDATE.

Path Header Support (RFC3327)

RFC 3327 defines the extension header field, Path, which provides a mechanism for multiple SIP proxies to add themselves to a defined path between a UAC and registrar. The Path header functions similar to the SIP Record-Route header, but is defined outside of a Dialog.

WebLogic SIP Server provides API support to enable multiple proxy applications to add themselves to a Path header in a REGISTER request. A deployed registrar application could then manage and maintain the Path headers.

To provide this support, WebLogic SIP Server extends the `javax.servlet.sip.Proxy` interface with `com.bea.wcp.sip.WlssProxy`. The extended interface provides getter/setter methods for the `AddToPath` attribute, which determines whether `proxyTo()` operations automatically add a Path header to the proxied request. The interface also provides a method to retrieve the complete URI defined in the Path header for a request, so that the proxy can add arbitrary attributes to the header.

See the com.bea.wcp.sip.WlssProxy JavaDoc for more information.

Note that WebLogic SIP Server does not provide a method for a proxy application to push an arbitrary Path entry (as well as its own Path as supported in the extended API).

Support for SIP REFER Method (RFC3515)

WebLogic SIP Server supports the SIP refer-to header and REFER method for applications that implement services such as unattended or attended call transfer, and third-party call control.

A “REFER” request can come within an existing dialog, or out of dialog. If REFER arrives out of a dialog, it starts a new dialog. WebLogic SIP Server creates an implicit subscription after the 202 Accepted response is sent, and the dialog remains active until the subscription expires. The subscription and original dialog are maintained until a subsequent NOTIFY or method explicitly terminates the subscription. The subscription for REFER can also be terminated with a SUBSCRIBE request sent with expiration = 0.

If a NOTIFY request doesn't find a REFER subscription it is rejected with a 481 status response.:

Because of the implicit subscription described above, WebLogic SIP Server maintains a SIP dialog after the REFER method is used even if a subsequent BYE message is sent. This behavior ensures that call transfer applications can re-establish the original session (via an INVITE) if the call transfer fails. See Section 2.4 Transfer - Unattended in http://tools.ietf.org/wg/sipping/draft-ietf-sipping-service-examples/draft-ietf-sipping-service-examples-09.txt for more information.

WebLogic SIP Server extends the JSR 116 `SipServlet` class to provide a convenience method for handling SIP REFER requests. To provide a handler your own SIP Servlet, extend `com.bea.wcp.sip.WlssSipServlet` and implement the `doRefer()` method. For example:

```
package myapp;

import javax.io.IOException;
import javax.servlet.ServletException;
import com.bea.wcp.sip.WlssSipServlet;

public class MyApp extends WlssSipServlet {

  protected void doRefer(SipServletRequest req)
    throws ServletException, IOException {

  // Respond to REFER method...

  }

}
```

Changes to SIP Request Authentication for 3GPP TS 24.229

The authentication process for SIP requests having the P-Asserted-Identity header was changed to conform to the behavior described in 3GPP TS 24.229: IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP). In version 2.2, if the SIP message contains a Privacy header with either an “id” or “user” value, the user is treated as being anonymous. Previously, only the “id” value was considered. See Configuring P-Asserted-Identity Assertion in *Configuring Security*.

WebLogic SIP Server 2.2 authentication behavior meets the requirements of 3GPP TS 24.229 except as follows:

- The server performs authentication only for protected resources. To fully conform to 3GPP TS 24.229, SIP Servlets should use declarative security to assign permissions to all resources. See Securing SIP Servlet Resources in *Developing Applications with WebLogic SIP Server*.

- If an anonymous user fails an authorization check, WebLogic SIP Server forces authentication by challenging the user using the `auth-method` configured in the `login-config` element of the application's `sip.xml` descriptor. This is done instead of immediately returning a 403 Forbidden response.
- The server does not evaluate the From header when handling messages with P-Asserted-Identity. This behavior is not necessary because the server automatically forces authentication by challenging the user if an anonymous user fails an authorization check.
- WebLogic SIP Server does not enforce a maximum number of authentication challenges. Instead, a user is locked for a period of time if repeated authentication attempts fail. See Protecting User Accounts in the WebLogic Server 9.2 documentation for more information.
- 3GPP TS 24.229 specifies that a server should provide the option to return a 2xx final response for a non-anonymous user when the user is not authorized to access a requested resource. WebLogic SIP Server does not provide the option to return these "fake" 2xx responses, and instead issues a 403 Forbidden response. If necessary, application code can easily construct "fake" 2xx responses using one of two different method:
  - **Using programmatic security rather than declarative security.** An Application can use the `HttpServletRequest.isUserInRole()` method to determine if a user is authorized to access a particular resource. If the user is not authorized, the application can construct a "fake" 2xx response to return to the UAC and mask the authorization failure.
  - **Using application composition.** One Servlet in an application chain can act as a filter that converts 403 Forbidden responses to a 2xx final response.

Support for X-3GPP-Asserted-Identity Header (3GPP TS 33.222)

WebLogic SIP Server now supports the `X-3GPP-Asserted-Identity` header for HTTP authentication as specified in 3GPP TS 33.222. Two new security providers, `X3gppAssertedIdentityAsserter` or `X3gppAssertedIdentityStrictAsserter` are available to enable and configure support. See Configuring 3GPP HTTP Authentication Providers in *Configuring Security*.

Converged Application Support

WebLogic SIP Server now supports the development and deployment of *converged applications* that combine SIP protocol functionality with features from J2EE such as HTTP sessions, Enterprise JavaBeans (EJBs), and Java Message Service (JMS).

An extended API is provided to help you obtain `SipApplicationSession` objects and to create and associate HTTP sessions with `SipApplicationSession` objects.

An extended API is also provided to help non-SIP Servlets concurrently modify a `SipApplicationSession` object in a replicated environment. See Developing Converged Applications in *Developing Applications with WebLogic SIP Server*. See also the converged application example installed with WebLogic SIP Server (`wlss_home/samples/server/examples/src/convergence/readme.html`).

Production Application Upgrade

You can upgrade a deployed SIP application to a different version without losing calls that the application is current processing. WebLogic SIP Server supports application upgrades by allowing two different versions of the same application to be deployed simultaneously. The server itself automatically routes new calls to the latest-deployed application version, while directing messages for already-established calls to the older application version. After a period of time, the new application version processes all SIP messages, while the older version processes no messages and can be safely undeployed. See Upgrading Deployed SIP Applications in the *Operations Guide*.

Improved Failover Detection

A new failover detection mechanism is provided to improve failover performance in the event that a data tier server process immediately fails, or the network connection between an engine and data tier server is physically compromised. See Improving Failover Performance for Physical Network Failures in *Configuring and Managing WebLogic SIP Server*.

Content Indirection

WebLogic SIP Server now supports SIP applications that indirectly specify the content of the SIP message body (for example, using an HTTP URL). Content indirection is generally used move large message bodies out of the SIP signalling network, or to allow bandwidth-constrained applications to determine whether or not they should download the additional content. Instead of providing the content directly in the message, the message body contains only a link to the specified content and metadata that specifies the size and type of information to be retrieved.

To support content indirection, WebLogic SIP Server provides an API to enable SIP Servlets to easily determine whether or not a message uses content indirection. For messages that include indirect content, the API provides for easy retrieval of the content metadata. See Using Content Indirection in SIP Servlets in *Developing Applications with WebLogic SIP Server*. See also the draft specification for content indirection at http://tools.ietf.org/wg/sip/draft-ietf-sip-content-indirect-mech/draft-ietf-sip-content-indirect-mech-05.txt.

Reliable Provisional Responses (RFC 3262)

WebLogic SIP Server supports reliable provisional responses (defined in http://www.ietf.org/rfc/rfc3262.txt). For a particular dialog there may be multiple reliable provisional responses at a given time. For this reason, the existing `SipSession.createRequest()` method in the SIP Servlet API is not sufficient for handling PRACK. Also, the basic `doAck()` method cannot be used for generating PRACK from the reliable provisional responses themselves.

Because of these issues, WebLogic SIP Server provides two alternatives for creating PRACK from the reliable provisional responses themselves:

1. `WlssSipServletResponse.createPrack()`—SIP Servlets can acknowledge a provisional response (and stop retransmissions of provisional responses) by creating a PRACK request using the `com.bea.wcp.sip.WlssSipServletResponse.createPrack()` method. This method returns a PRACK request having the RAck header sequence number of the response that is being acknowledged.

2. `SipServletResponse.createAck()`—The implementation of the JSR116 `SipServletResponse.createAck()` method was modified to automatically return a PRACK request instead of an ACK request for provisional (non-2xx) responses.

See the WebLogic SIP Server JavaDoc for `com.bea.wcp.sip.WlssSipServletResponse`.

Support for Modifying Contact Header Parameters

WebLogic SIP Server provides limited support for modifying Contact header parameters, which is required by the 3GPP IMS specification. In version 2.2, applications can use the following `SipServletMessage` method call to return a Contact address object whose parameters can be modified:

`SipServletMessage.getAddressHeader("Contact");`

The object returned allows only limited modification of parameters in this release. Specifically, the object does not permit applications to add a URI user part to the Contact header, or to modify parameters that could influence the network interface that the SIP Container uses for transmitting messages.

In addition to the above change, the code was modified to allow a UAS to modify Contact header parameters for a 200 response to an OPTIONS request, as required in RFC 3261.

New Example Applications

WebLogic SIP Server 2.2 introduces a number of new and revised example applications. Source and build files for the new examples are installed at *`WLSS_HOME`*`\samples\server\examples\src`, where *`WLSS_HOME`* is the directory in which you installed WebLogic SIP Server (for example, `c:\bea\wlss220`).

See the *`WLSS_HOME`*`\samples\server\examples\src\index.html` file for information about all examples installed with WebLogic SIP Server.

Support for Generating SNMP Traps from SIP Servlets

WebLogic SIP Server 2.2 introduces a new runtime MBean, `SipServletSnmpTrapRuntimeMBean`, that enables applications to easily generate SNMP traps. The WebLogic SIP Server MIB contains seven new OIDs that are reserved for traps generated by an application. See Generating SNMP Traps from Application Code in *Developing Applications with WebLogic SIP Server*.

Default SIP Servlet Configuration

A new configuration element in `sipserver.xml`, `default-servlet-name`, enables you to specify a default SIP Servlet that the container calls for incoming initial requests when no other Servlet can be matched via `servlet-mapping` definitions in `sip.xml`. See default-servlet-name in the *Configuration Reference*.

The `default-servlet-name` element cannot be modified using the Administration Console. You must modify this value directly in `sipserver.xml` using a text editor, and redeploy the `sipserver/config` Web Application to all engine tier servers, to use this feature. You can redeploy the `config` Web Application using either the Administration Console or the `weblogic.Deployer` utility. To redeploy from the Administration Console:

1. For single-server domains, expand the Servers node and select the name of the WebLogic SIP Server instance. For multiple-server domains, expand the Clusters node and select the name of the engine tier cluster.
2. Select the Deployments->Web Modules tab in the right pane.
3. Select the config application from the table.
4. Select the Deploy tab.
5. Click Redeploy next to the single server instance, or the engine tier cluster on which the application is deployed.

To redeploy using the `weblogic.Deployer` utility, enter a command to:

```
java java weblogic.Deployer -username weblogic -password weblogic -verbose
   -targets config@BEA_ENGINE_TIER_CLUST -name sipserver -adminurl
   t3://localhost:7001 -redeploy"
```

What's New in WebLogic SIP Server 2.1?

This section details features that were introduced between WebLogic SIP Server 2.1 and earlier versions. This section includes information about the following:

- "Architectural Changes" on page 1-17
- "Application Porting Guidelines" on page 1-18
- "New Security Features" on page 1-18
- "Configuration Changes" on page 1-19
- "Container Changes for send() Calls" on page 1-20

Architectural Changes

The architecture of WebLogic SIP Server 2.1 was dramatically improved to provide increased performance, higher availability, and flexibility in configuring available resources. The most visible change in architecture is in WebLogic SIP Server's use of two separate clusters, referred to as the engine tier and data tier, that can be sized independently of one another to increase the call throughput or availability of an installation. (Development installations and small production installations can also use a single server instance as needed.) See Overview of the WebLogic SIP Server Architecture for more information about these changes.

WARNING: When you configure a domain with multiple engine and data tier servers, you must accurately synchronize all server system clocks to a common time source (to within one or two milliseconds) in order for the SIP protocol stack to function properly. See Configuring NTP for Accurate SIP Timers for more information.

Application Porting Guidelines

SIP Servlets developed for previous versions of WebLogic SIP Server must observe new coding practices and requirements in order to operate in the version 2.1 distributed environment:

- Applications should not create threads. When the `do`*`xxx`* method of a SIP Servlet is invoked by the SIP Servlet container, the container automatically locks the associated call state. If the `do`*`xxx`* method spawns additional threads or accesses a different call state, deadlock scenarios can occur.
- All Servlets should be non-blocking.
- All Session data must be serializable.
- Servlets must use the `distributable` tag, in order to deploy them to a cluster of engine tier WebLogic SIP Server instances. The `distributable` tag is not required and is ignored if you deploy to a single combined-tier (non-replicated) WebLogic SIP Server instance.

These and other porting requirements are described in Requirements and Best Practices for WebLogic SIP Server Applications in *Developing Applications with WebLogic SIP Server*.

New Security Features

WebLogic SIP Server now supports the `P-Asserted-Identity` SIP header as described in RFC3325. See Trusted Host Forwarding with P-Asserted-Identity in *Configuring Security*.

WebLogic SIP Server now supports Client-Cert authentication as well as BASIC and Digest authentication. See Configuring Client-Cert-Based Authentication for SIP Applications in *Configuring Security*. Client-Cert authentication is disabled by default; the switch to enable is defined in `ClusterMBean` and `ServerMBean`.

WebLogic SIP Server 2.1 now includes an RDBMS (JDBC) Digest Identity Assertion provider as well as the LDAP provider included in previous versions. In addition, both the RDBMS and LDAP providers support reverse-encrypted passwords as well as clear-text and hashed password values. See Configuring Digest Authentication for more information.

Configuration Changes

The following sections describe changes in the way you configure and manage WebLogic SIP Server 2.1.

datatier.xml Changes (Formerly statetier.xml)

The configuration file used to define partitions and replicas in the data tier is now named `datatier.xml`. In addition, the main XML element defined in the file has changed to `data-tier` (formerly `state-tier`). The location of the data tier configuration file has also changed; both `datatier.xml` and `statetier.xml` are located in the *`DOMAIN_DIR`*`/sipserver/config` subdirectory, where *`DOMAIN_DIR`* is the root directory of the WebLogic SIP Server domain.

Load Balancer Configuration Changes

Load balancer addresses are no longer defined in the `sipserver.xml` configuration file. Instead, the load balancer address and port number must be defined in the **External Listen Address** and **External Listen Port** fields of a SIP channel on each engine tier server. See Configuring Load Balancer Addresses in *Configuring Network Resources*.

Changes in Queue Length-Based Overload Protection

When using queue- length-based overload protection controls, WebLogic SIP Server now monitors the *sum* of the lengths of the `sip.transport.Default` and `sip.timer.Default` execute queues, rather than only the length of `sip.transport.Default`. Also, the default overload configuration initiates overload protection when the combined queue size reaches 200 simultaneous requests, and releases overload control when the combined size falls to 150 simultaneous requests. See overload in the *Configuration Reference*.

sipserver.xml Changes

The schema for `sipserver.xml` has changed in WebLogic SIP Server. See the Engine Tier Configuration Reference (sipserver.xml) in the *Configuration Reference*. Notable changes include:

- Proxy entries now use single-elements that include a full URI (rather than multiple elements for each portion of a URI).
- The terminology and XML elements for regulating incoming SIP calls has changed. See overload.

In addition to schema changes, the location of `sipserver.xml` has changed in this release. The `sipserver.xml` is included as part of the `sipserver` enterprise application that implements the SIP container features of WebLogic SIP Server. See Overview of WebLogic SIP Server Configuration and Management.

Network Configuration Using Channels

WebLogic SIP Server 2.1 no longer uses the (previously-deprecated) `connector` element in `sipserver.xml` for configuring network connections. Instead, all connections are defined using:

- The listen address and listen port for the WebLogic SIP Server instance, and
- All network channel resources configured for the WebLogic SIP Server instance.

Multiple network channels can be defined to support multiple transport protocols or to configure multiple network interfaces on multi-homed server hardware. See the Managing WebLogic SIP Server Network Resources in *Configuring Network Resources*.

Access Logging Configuration Changes

Access logging is no longer configured by defining a filter in the `sipserver.xml` configuration file. See Enabling Access Logging in *Developing Applications with WebLogic SIP Server* for information about the new XML configuration elements.

Container Changes for send() Calls

In previous WebLogic SIP Server releases, if an application called the `send()` method within a SIP request method such as `doInvite()`, `doAck()`, `doNotify()`, and so forth, the SIP Servlet container immediately transmitted the `send()` call to the network. In WebLogic SIP Server 2.1, `send()` calls are instead buffered in the order in which they are called, and transmitted in order only after the control has returned to the SIP Servlet container.

WARNING: Applications must not wait or sleep after a call to `send()`, because the request or response is not transmitted until control returns to the SIP Servlet container.

What's New in WebLogic SIP Server 2.0 SP2

WebLogic SIP Server 2.0.2 introduced the following features:

- Digest Authentication support was introduced using a new LDAP Digest Authentication Identity Asserter and a separate authorization provider. See Configuring Digest Authentication for WebLogic SIP Server.

- Deployment Descriptors in `weblogic.xml` introduced support for mapping principal and role names to roles defined in `sip.xml`. See Securing SIP Servlet Resources.
- SNMP support was enabled by default; a patch was no longer required.
- Operator documentation for WebLogic SIP Server SNMP Traps was made available. See Understanding and Responding to SNMP Traps.
- Administrator documentation was made available to help configure Administration Servers in a WebLogic SIP Server domain.
- Developer documentation was made available to describe how to use the Eclipse IDE to develop SIP Servlets with WebLogic SIP Server. See Developing SIP Servlets Using Eclipse.

Deprecated Features in WebLogic SIP Server 2.0 SP1

WebLogic SIP Server 2.0.1 was a restricted release of a BEA product and was subject to change in future releases. The following features were specifically called out as having been deprecated in WebLogic SIP Server 2.0.1:

- Load balancer URIs will be obtained from Network Channel attributes, instead of `sipserver.xml`, in a future release of WebLogic SIP Server.
- Future releases of WebLogic SIP Server will not use the built-in WLSSecurityManagerFilter and custom security manager to implement digest authentication.
- `sipserver.xml` may not be used to configure SIP Servlet container features in future releases.
- The session backup implementation was deprecated and subject to change in future releases.
- Creating general-purpose SIP filters by extending a base filter class was deprecated in this release.

CHAPTER 2

Resolved Problems in WebLogic SIP Server 3.1 MP1

The following table summarizes the issues that were resolved in WebLogic SIP Server 3.1 MP1. See also Table 2-2, "Problems Resolved in Version 3.1," on page 2-9.

Table 2-1 Problems Resolved in Version 3.1 MP1

Change Request Number	Description
CR318098	When proxying a request, WebLogic SIP Server treated the Reason header as an Unknown Header and removed it when sending a CANCEL request to a UAS. The code was modified so that, upon receiving a CANCEL from the previous hop, the server copies any Reason header field into the new CANCEL request before proxying.
CR320740	When a SIP agent returned different To: header tags in the 183 Session Progress and 200 OK messages, ACK requests created on the 200 OK could have the To: header tag of the 183 response. This problem was addressed by updating the way in which derived sessions are created for proxy and B2BUA scenarios.

Table 2-1 Problems Resolved in Version 3.1 MP1

Change Request Number	Description
CR321374, CR359194	When browsing WebLogic SIP Server MBeans using SNMP commands, the following problems occurred: 1. The SNMPWALK, SNMPGETNEXT, and SNMPGET commands did not work in a basic standalone WebLogic SIP Server domain. 2. The SNMPWALK, SNMPGETNEXT, and SNMPGET commands worked only for Managed Servers running in Managed Server Independence (MSI) mode. 3. No information was provided for the following WLSS MIB OID's: – sipServerRuntimeTable (.1.3.6.1.4.1.140.626.8) – sipApplicationRuntimeTable (.1.3.6.1.4.1.140.626.9) – replicaRuntimeTable (.1.3.6.1.4.1.140.626.10) 4. ActiveApplicationSessionCount (.1.3.6.1.4.1.140.626.8.1.5) and ActiveSipSessionCount (.1.3.6.1.4.1.140.626.8.1.5) were always reported as zero, and DestroyedApplicationSessionCount (.1.3.6.1.4.1.140.626.8.1.10) and DestroyedSipSessionCount (.1.3.6.1.4.1.140.626.8.1.11) were neither visible nor accessible. The code was modified to correct the above issues. Ensure that you enable Anonymous Admin Lookup by following these steps: 1. Access the Administration Console for the domain. 2. Click Lock & Edit. 3. Select the name of your domain in the left pane. 4. Select Security->General in the right pane. 5. Select Anonymous Admin Lookup Enabled. 6. Click Save. 7. Click Activate Changes. Also, note that WebLogic SIP Server instances ignore the SNMP port number specified on the SNMP configuration page. Servers use the default port 1610 to start the SNMP agent and, if this port is not available, the port number is incremented until an unused port is discovered. You can specify the starting port number using the `-DWLSS.SNMPPort=`*`port_number`* startup argument. The server logs display the actual SNMP port on which the server is listening.
CR321567	Applications were not handling NOTIFY messages without a previous SUBSCRIBE. The code was modified to allow NOTIFY to be propagated to an application even if there is no Subscription.

Table 2-1 Problems Resolved in Version 3.1 MP1

Change Request Number	Description
CR321961	In a forking scenario, WebLogic SIP Server forwarded to the application all the responses it received from all forked branches, rather than forwarding only the "best" response. The code was modified so that only the best final response is forwarded to the application in supervised mode. Applications now receive just one response.
CR325071	Angle brackets were not preserved in ALERT_INFO even when following the name-addr format. The code was modified to preserve angle brackets in ALERT_INFO.
CR325192	The remote target was not updated when a 1xx was received with to tag. This caused an incorrect request uri for any subsequent request created using `SipSession`. The code was modified to update the remote target using the Contact Header value available in the 1xx response. Subsequent request now carry the correct request uri.
CR325538	WebLogic SIP Server did not call the `rcvMessage(Message msg)` in Ro online charging applications for classes implementing the `com.bea.wcp.diameter.SessionListener` interface if the received CCA message did not contain a Session-Id. The code was modified to look for a Session-Id in the original Request if the Answer has no Session-Id
CR327764	`java.lang.IllegalStateException` could be thrown during a forking proxy test under load when trying to send a response to CANCEL if the ServerTransaction was already in the completed state. The code was modified to allow a response to the actual CANCEL request being created/sent even if the final response has been sent. A response for CANCEL is now always sent as an acknowledgement of its receipt
CR330315	The container failed to set the Record Route header using the External listen port even if it was configured to do so. The code was modified to honor the external listen port configuration.
CR331068	The following error sometimes occurred with forking proxies under load: `java.lang.InternalError: Proxy is null` This occurred because the proxy was not set in CANCEL messages. The problem was addressed with a code fix.

Table 2-1 Problems Resolved in Version 3.1 MP1

Change Request Number	Description
CR332708	WebLogic SIP Server could throw a `NullPointerException` under heavy load during the `resetCache()` operation. This problem was addressed with a code fix.
CR333893, CR358944	Because of the RFC3261 limitation described at http://bugs.sipit.net/show_bug.cgi?id=706, transactions could remain in PROCEEDING state and the associated call states were never cleaned up. In certain conditions involving very high loads, this could lead to Out of Memory errors. The code was modified to account for the RFC limitation. The container now automatically cleans up a call state that has a PROCEEDING transaction when all application sessions associated with the callstate are invalidated.
CR333988	In certain scenarios where a request was created using SipFactory, the server did not correctly set the call-id. The code was modified so that a request created using SipFactory includes the call-id from the original request when the call-id should remain the same.
CR336977	The container could throw a `NullPointerException` with messages containing an empty via header. This problem was addressed with a code fix.
CR339985	When performing a JNDI lookup in the MBeanServer, the SIP Servlet context could not found in the Context Loader Listener and would yield a `NullPointerException`. This problem was resolved with a code fix.
CR334695	In proxy mode, WebLogic SIP Server did not create an implicit subscription when a REFER request was received, instead sending 481 for NOTIFY. The code was modified to add an implicit subscription when a proxy is handling a REFER request.
CR338929	A `NullPointerException` was thrown when deploying a SIP application having an empty `sip.xml` file. The code was modified to set the default Servlet to BlankServlet (CR338929) when `sip.xml` is empty or has no Servlet definition or mappings.

Table 2-1 Problems Resolved in Version 3.1 MP1

Change Request Number	Description
CR342706	SipSession was transitioned to the initial state even when a 3xx-6xx response was received for a subsequent request in the early/initial state. The code was modified to ensure that the transition to the initial state occurs only after receiving a 3xx-6xx response, corresponding to Initial Request only, in the early/initial state.
CR343371	The container did not clean up expired transactions in a timely manner, which could lead to large call state sizes and increased memory usage. The code was modified to more efficiently clean up expired transactions.
CR344763	The processing flag was not set correctly if a `RuntimeException` caused the Message Queue processing thread to die. As a result of the incorrect flag, messages could accumulate on the queue even though there was no associated thread to perform the work. The code was modified to ensure that there is a Thread associated with the Message Queue to process queued messages even if a `RuntimeException` occurs.
CR345018	WebLogic SIP Server failed to invoke the `doAck()` method for certain messages because it could not determine the correct target address necessary to retrieve the application session. The code was modified to also examine the popped ROUTE when determining the target address. This change ensures that the correct call state can be associated with the request.
CR347844	When an INVITE with multiple contacts was received, the following error occurred: `java.lang.IllegalArgumentException: Multiple Contacts in SIP request` The code was modified to add a validation to check for multiple contacts in the INVITE header.
CR349288	The server failed to destroy or invalidate a `SipApplicationSession` even after the specified `session-timeout` or default timeout of 30 minutes was reached. The code was modified to ensure that: • `max-application-session-lifetime` determines the upper limit of `session-timeout`, and • `max-application-session-lifetime` is not used as the `session-timeout` value, and • the actual `session-timeout` value is honored.

Table 2-1 Problems Resolved in Version 3.1 MP1

Change Request Number	Description
CR349747	Using the `weblogic.Admin` utility to look for the `SipServletSnmpTrapRuntime` bean caused the following error: `Unable to determine parent types for SipServletSnmpTrapRuntime: while calcluating parent for mydomain:ServerRuntime=Server-0,Name=Server-0,Type=SipServletSnmpTrapRuntime,Location=Server-0,SipServerRuntime=Server-0` This problem was addressed by correcting the parent type for `SipServletSnmpTrapRuntime`. However, note that the `weblogic.Admin` utility is deprecated in WebLogic Server version 9.2.
CR350920	Rebooting replica server instances could sometimes result in a `NullPointerException` when the replicas were not updated with keyCount and highKeyCount statistics. The code was modified to properly initialize rebooted replicas when joining the data tier.
CR351150	An application was able to send a CANCEL even after receiving 200/INVITE. The container was modified to disallow sending a CANCEL when a client transaction is in the COMPLETED state. Attempting to do so now logs a message.
CR353471	A `NullPointerException` could be thrown when updating `max-application-session-lifetime` via the Administration Console. The code was modified to properly validate `max-application-session-lifetime` entries made in the Administration Console.
CR354210	`req.getRemoteAddr()` did not work in a clustered environment when Engine tier caching was turned off and the request was retrieved from a `SipApplicationSession`. The method returned null instead of returning the actual remote address. This problem was resolved with a code fix.
CR354549	A race condition could cause repeated 200 OK and 487 retransmissions to occur even after the maximum retransmission timeout (64*T1) occurred. This problem was resolved with a code fix.

Table 2-1 Problems Resolved in Version 3.1 MP1

Change Request Number	Description
CR355168	A `NullPointerException` was sometimes thrown when creating a derived SIP Session. The code was modified to prevent derived sessions from being created if the parent session is itself invalid. This fix prevents the `NullPointerException`.
CR356017, CR329432	An application that implemented `SipErrorListener` did not receive a callback via `noAckReceived(SipErrorEvent)` even when the 200 Ok retransmission threshold was reached (after 64*T1 seconds). The code was modified to implement the `noAckReceived` callback for 2xx responses for INVITE until the retransmission threshold is reached.
CR359640	The Administration Console reported a `NoSuchMethodError` when deploying `%BEA_HOME%\weblogic92\samples\server\examples\build\webservicesJwsEjbClientEar` and similar EAR deployments. The code was modified to properly handle EAR deployment.
CR365710	When starting a replicated domain, if more than two replica servers were started before starting an engine, the second replica would shut down due to the replica peer checking process. The code was modified so that the peer check is performed only after at least one engine has registered with the replicas. This ensures that the partition view is communicated with all replicas via an engine only when an engine is available.

Table 2-1 Problems Resolved in Version 3.1 MP1

Change Request Number	Description

CR366214	The WebLogic SIP Server `doAction` API could cause unnecessary locking of call state data. For example application code such as:
	`   WlssApplicationSession appSession =`
	`sessions.getApplicationSession(appSessionId);`
	`   appSession.doAction(new WlssAction() { ... }`
	causes the call state to be locked and unlocked once to obtain the application session ID, and then once again to perform the `doAction` operation. To address this problem, the following new API was added:
	`Sessions.doAction(WlssAction a, String appSessionId)`
	To make use of the new API, replace application code that uses the above pattern with the following pattern:
	`sessions.doAction(confAppSessionId, new WlssAction() {`
	`  public Object run() throws Exception {`
	`    // this will not cause an attempt to lock again`
	`    SipApplicationSession conf =`
	`sessions.getApplicationSession(confAppSessionId);`
	`    conf.setAttribute('name', 'yourName');`
	`    return null;`
	`  }`
	`});`
	In the above pattern, the `sessions.getApplicationSession` method inside `run()` does not cause lock and unlock operations.
	Note that the older API remains for backward compatibility.

CR367390	This release improves server performance on Intel Quad Core processors when running in a non-replicated configuration. The code was modified to enable the database persistence flag by default, which in turn enables local serialization of call state data. This setting is ignored for replicated configurations.

The following table summarizes the issues that were resolved in WebLogic SIP Server 3.1.

Table 2-2 Problems Resolved in Version 3.1

Change Request Number	Description
CR239639	When a SIP request was sent via application code, WebLogic SIP Server immediately started the transaction timer even though the actual message was queued for sending only after the application's service method ended. If the application code (or the system load) caused a delay in exiting the service method, retransmissions for the request could appear in a shorter amount of time than the configured SIP timers would indicate. (In extreme cases, the application request and its retransmission would appear on the wire at nearly the same time.) To address this problem, the code was modified to buffer the timer scheduling along with the request. This ensures more accurate timer behavior.
CR290540	In previous versions, `setCharacterEncoding()` did not throw an `UnsupportedEncodingException`. Existing Servlet code that called this method and used a catch claus for `UnsupportedEncodingException` had to be modified before recompiling for deployment to WebLogic SIP Server. This problem was addressed with a code fix.
CR291406	If you ran WebLogic SIP Server on a Windows platform with the JRockit JVM, you had to disable JRockit native IO in order to use SSL. If you did not disable native IO, an exception was generated similar to: `java.io.IOException: couldn't initialize IOPort: The parameter is incorrect.` This problem was addressed with a code fix.

Table 2-2 Problems Resolved in Version 3.1

Change Request Number	Description

CR297102	While proxying an ACK message, if a `TooManyHopsException` was thrown WebLogic SIP Server would attempt to send a response to the ACK. This caused the exception:
	`<Oct 17, 2006 1:50:18 PM PDT> <Error> <WLSS.Session> <BEA-331412> <Failed to`
	`respond 483 to too many hops request.`
	`java.lang.IllegalStateException: Cannot createResponse for ACK`
	The code was modified to drop the ACK after logging a message in this circumstance. The ACK can them be re-sent by the UAC when it receives a retransmission request.

CR298765	WebLogic SIP Server allowed you to deploy SIP applications with deployment descriptors that did not conform to their respective schemas. This could lead to exceptions or other unexpected behavior at runtime. The code was modified to reject application deployments when deployment descriptors do not conform to the schema.
CR299384	WebLogic SIP Server would log "unknown header" messages for the `WLSS-Default-Handler` header, even though this header is created and used by the server code itself. The header was registered and no longer generates error messages.
CR300878	When proxying a request over TCP, if an `IOException` was raised, WebLogic SIP Server logged the exception but did not send a final response upstream. Because of this, a UAC would not know whether an invite request was proxied successfully. The code was modified to detect transport errors while proxying, and create and send a 503 error response upstream in response to such errors.

Table 2-2 Problems Resolved in Version 3.1

Change Request Number	Description

CR301268	If a Servlet used the `load-on-startup` deployment descriptor element to initialize the Servlet on startup (rather than on first request), and the `init` method created a new call state, WebLogic SIP Server would throw the following exception if any partition in the data tier was not yet online:
	`Unexpected exception com.bea.wcp.sip.engine.server.SetupException : [WLSS.Engine:330027]Failed to initialize "proxy" servlet class test.ProxyServlet`
	`java.lang.IllegalStateException: PartitionClient offline`
	The code was modified to delay initialization a Servlet if all of the following conditions are met:
	• The Servlet uses `load-on-startup`
	• The Servlet overrides the `SipServlet.init` method
	• The Servlet is not deployed on the Administration Server
	• A configured partition is not yet online.
	Note that the initialization delay is not applied to Administration Server deployments, because doing so could prevent replica servers from loading. Never deploy any applications to the Administration Server in a production system.

CR301664	WebLogic SIP Server used a fixed overload duration of 30 seconds. This could cause poor performance and continual overload situations with periodic spikes in `wlss.transport` work manager queue. The code was modified to set the initial overload duration to a much shorter 512 milliseconds, and then dynamically increase the duration if necessary in response to recurring overload conditions.

Table 2-2 Problems Resolved in Version 3.1

Change Request Number	Description
CR303194	On Windows platforms, if you installed the WebLogic SIP Server product nested inside of other folders, the Administration Console extension could not load due to the length of the path being too long. To work around the problem, the following environment variable could be set before starting the Administration Server: `set JAVA_OPTIONS=-Dweblogic.j2ee.application.tmpDir=d:/TEMP` This option is now automatically included in the server startup script, `commEnv.sh`.
CR303219	WebLogic SIP Server allowed SIP requests to access the retiring version of a deployed SIP application. This behavior was inconsistent with the base WebLogic Server application upgrade functionality, which disallows HTTP requests to a retiring application. For example, if you retired a converged SIP application, HTTP requests to the application would be rejected while SIP requests were permitted. The code was modified to be consistent with WebLogic Server behavior; the server now disallows SIP requests to a retiring application.
CR303769	Earlier WebLogic SIP Server versions ignored the encoding set through the `SipServletMessage.setCharacterEncoding()` method, and only honored the encoding if set using `contentType` argument of the `setContent()` method. This problem was addressed with a code fix.
CR304389	The server did not register SIP container runtime MBeans with the compatibility MBean server. This would lead to exceptions such as: `javax.servlet.ServletException: Unable to lookup type 'SipServletSnmpTrapRuntime'` The code was modified to address this problem.

Table 2-2 Problems Resolved in Version 3.1

Change Request Number	Description
CR305182	When using WebLogic SIP Server with geographically-redundant installations, each write to a secondary site would log an error message similar to: `<Dec 13, 2006 12:48:08 PM PST> <Error> <Security> <BEA-090513> <ServerIdentity failed validation, downgrading to anonymous.>` This problem was addressed with a code fix.
CR306926	WebLogic SIP Server provided no way to configure the timeout duration for SCTP connections. The code was modified to honor a custom channel property, `SctpConnectTimeoutMillis`, to configure the property. See Configuring Custom Timeout, MTU, and Other Properties in *Configuring Network Resources*.
CR308370	WebLogic SIP Server could omit the tag parameter in the `To` header for PRACK messages. The code was modified to ensure that PRACK messages always include the `To` tag.
CR309866	WebLogic SIP Server exhibited poor scalability performance on Sun Sparc Enterprise T2000 servers when using the Sun JVM. These performance problems are addressed in Release 13 of the Sun JVM.
CR310215	WebLogic SIP Server would throw a `NullPointerException` if an application used `SipServletSnmpTrapRuntimeMBean` to generate an SNMP trap outside of a `do`*`xxx`* method. The code was modified to allow trap generation outside of a `do`*`xxx`* method. Note, however, that traps generate outside of a `do`*`xxx`* method use the string "Non Sip-Servlet Scope Application" instead of a Servlet name.
CR310657	The Diameter implementation used incorrect values (3 and 4) for the CHECK_BALANCE and PRICE_ENQUIRY values of the `Requested-Action` AVP. The code was modified to use the correct values of 2 and 3, respectively, as described in RFC4006.

Table 2-2 Problems Resolved in Version 3.1

Change Request Number	Description
CR310782	The Diameter implementation did not allow for a Diameter application to receive and process ASR requests. This meant that Diameter applications could not add AVPs to the termination CCR or be notified when a session was finished. The code was modified so that if a `SessionListener` is registered, the Diameter implementation passes ASR requests to the application listener for handling. In this case, it is the responsibility of the application to generate CCRs as well as send ASAs.
CR320065	When only the Ro application was configured, the Diameter CER was missing the `Supported-Vendor-Id` AVP. The code was modified to allow configuration of `Supported-Vendor-Id` values in the `diameter.xml`, in one or more `supported-vendor-id` elements.
CR328219	WebLogic SIP Server exhibited poor scalability performance on Sun Sparc Enterprise T2000 servers when using the JRockit JVM. These performance problems are addressed with a patch to JRockit R27.3.1.

WebLogic SIP Server 3.1 Known Issues

The following table summarizes known issues and problems in WebLogic SIP Server 3.1.

Note: This section describes only those issues associated with the SIP Servlet container and data replication features of WebLogic SIP Server 3.1. See also the WebLogic Server 9.2 Known and Resolved Issues for information about known problems with WebLogic Server 9.2, which provides the underlying OA&M and J2EE capabilities of WebLogic SIP Server 3.1.

Table 3-1 Version 3.1 Known Issues

Change Request Number	Description
n/a	By default, new Diameter network channels are created with a default Idle Connection Timeout value of 65 seconds. Change this attribute from the default in order to ensure that connections are not dropped and recreated every 65 seconds. See Creating Network Channels for the Diameter Protocol.
n/a	WebLogic SIP Server MIB objects are read-only. You cannot modify a WebLogic SIP Server configuration using SNMP.

Table 3-1 Version 3.1 Known Issues

Change Request Number	Description
n/a	This version of Weblogic SIP Server exhibits two behaviors that do not conform to the JSR 116 specification: • MIME content is returned as a String object, rather than as a `javax.mail.Multipart` object as encouraged by the specification. • `isPersistent`, used for re-instantiating `ServletTimer` after server restarts, is not implemented. Also, WebLogic SIP Server does not support dialog stateless proxies, an optional feature described in the API JavaDoc for the `Proxy` interface, `setStateful()` method: "This proxy parameter is a hint only. Implementations may choose to maintain transaction state regardless of the value of this flag, but if so the application will not be invoked again for this transaction."
n/a	If you attempt to install WebLogic SIP Server 3.0 on Fedora Core 3 or 4 with `selinux` running, the installer throws a `java.lang.UnsatisfiedLinkError` exception. You cannot install WebLogic SIP Server while `selinux` is active.
n/a	If you configure two or more data tier replicas using the default WebLogic Server Listen Address configuration (which specifies no listen address), multiple data tier instances on the same machine cannot connect to one another. This problem occurs because, using the default Listen Address configuration, JNDI objects in the first booted server bind to all local IP addresses. To avoid this problem, always enter a valid IP address for each configured data tier server instance.
n/a	In a WebLogic SIP Server installation with two engine tier nodes and two data tier nodes in a partition (two replicas), if the connection to the data tier becomes "split" such that each engine tier server can only reach a different data tier node, one of the replicas is forced offline. To recover from this situation, always configure the Node Manager utility to restart data tier replicas automatically when a replica fails. This enables the replica to rejoin its associated partition and update its copy of the call state data without having to manually restart the server.

Table 3-1 Version 3.1 Known Issues

Change Request Number	Description
CR303216	During an overload condition, WebLogic SIP Server may log messages similar to: `<ACK received in state PROCEEDING:class=[ServerTransaction],` `objid=[25292416], key=[z9hG4bKc227250e04757a91cbdde388192e21f5],` `state=[3,PROCEEDING], method=[INVITE]>` This occurs even if the ACK could be safely ignored (for example, if the ACK was generated by the server for a 503 response). There is no workaround to this problem, but it should occur only rarely (during overload conditions).
CR314296	WebLogic SIP Server does not support monitoring network channels that use connectionless protocols. For this reason, you cannot monitor UDP traffic using the Monitoring->Channels tab of the Administration Console.
CR267829	When starting a replicated domain, if a partition has no running replicas and two replicas are started at the same time, the second replica shuts down if one or more engine tier servers are already running. To avoid this problem, always start all data tier servers ***before*** starting any engine tier servers in a replicated domain.

Table 3-1 Version 3.1 Known Issues

Change Request Number	Description
CR272491, CR189353	On Linux and UNIX systems, the default TCP connection timeout interval is usually very long and can cause Managed Servers to disconnect from the Administration Server under certain failure conditions. Specifically, if a single Managed Server in a domain fails abruptly or is disconnected from the network (for example, due to a removed network cable), the Administration Server tries to communicate to the failed server for the length of the TCP connection timeout value. During this time, the Administration Server does not send heartbeat messages to the remaining Managed Servers in the domain. Failing to send the heartbeat messages causes the remaining Managed Servers to consider the Administration Server as being offline, and they disconnect from the Administration Server. This finally causes the Administration Server to throw `PeerGoneExceptions` for the disconnected servers after the TCP timeout interval has been reached and the connection is closed. To work around this issue without changing the operating system TCP connection timeout value, use the `-Dweblogic.client.SocketConnectTimeoutInSecs` startup option when booting the Administration Server. BEA recommends using a value of 60 seconds to avoid numerous missed heartbeats (`-Dweblogic.client.SocketConnectTimeoutInSecs=60`).
CR294126	When an application in a replicated domain configuration is undeployed, WebLogic SIP Server uses timer processing to clean up the remaining call state data for the application. However, in a non-replicated configuration, the server attempts to invalidate remaining session data but does not destroy call states associated with the application; this may result in the server "leaking" call states that existed during application undeployment.

CR300715	Testing on Solaris platforms has shown that the following JVM arguments to improve performance with the Sun JVM for replica servers:
	
	`-server -Xms1024m -Xmx1024m -XX:+UseParNewGC -XX:+UseConcMarkSweepGC`
	
	For engine tier servers, these example arguments have shown to improve performance:
	
	`-server -Xms768m -Xmx768m -XX:+UseParallelGC -XX:MaxGCPauseMillis=400 -XX:+DisableExplicitGC`
	
	Note that these JVM settings have only been tested on Solaris platforms. For other platforms, begin with the example JVM arguments described in Tuning JVM Garbage Collection for Production Deployments.

Table 3-1 Version 3.1 Known Issues

Change Request Number	Description
CR302859	In order to use SCTP with IPv4 on Solaris, you must set the `-Dsctp.preferIPv4Stack=true` Java option when starting the server. You can edit your startup script to include this option, or set the environment variable: `export JAVA_OPTIONS=-Dsctp.preferIPv4Stack=true`
CR333737	The main examples index file installed at `*wlss_home*/samples/sipserver/examples/src/index.html` includes an extraneous reference to an example named `centrex`. Note that the documentation on this page is in error. No `centrex` example is installed with WebLogic SIP Server version 3.1.
CR346262	If you install the 64-bit version of WebLogic SIP Server installer package on Solaris, you must add the `-d64` option with the Sun JDK in order to specify 64-bit mode. If you omit the `-d64` option, the Sun JDK automatically defaults to 32-bit mode and the installer fails to install required 64-bit native libraries. This yields the following error on startup: `<Oct 4, 2007 4:54:28 AM EDT> <Error> <Socket> <BEA-000438> <Unable to load performance pack. Using Java I/O instead. Please ensure that a native performance library is in: *path*>`
CR347957	The Windows XP IPv6 implementation does not support dual mode sockets, and cannot be used with any available JVMs. 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