Oracle® Communications Converged Application Server

Technical Product Description

Release 5.0

E17655-03

July 2011

Oracle Communications Converged Application Server Technical Product Description, Release 5.0

E17655-03

Contents

B Acronyms

C References

Preface

This document provides a technical overview of Oracle Communications Converged Application Server, including its features, architecture, standards alignment, and supported platforms. It also describes the service invocation method of the SIP Servlet API.

Audience

This document is intended for anyone who needs to learn about Converged Application Server, especially those who configure and maintain it.

Related Documents

For more information, see the following documents in the Oracle Converged Application Server Release 5.0 documentation set:

- *Converged Application Server Installation Guide*
- *Converged Application Server Release Notes*
- *Converged Application Server Administration Guide*
- *Converged Application Server SIP Application Development Guide*
- *Converged Application Server Diameter Application Development Guide*

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

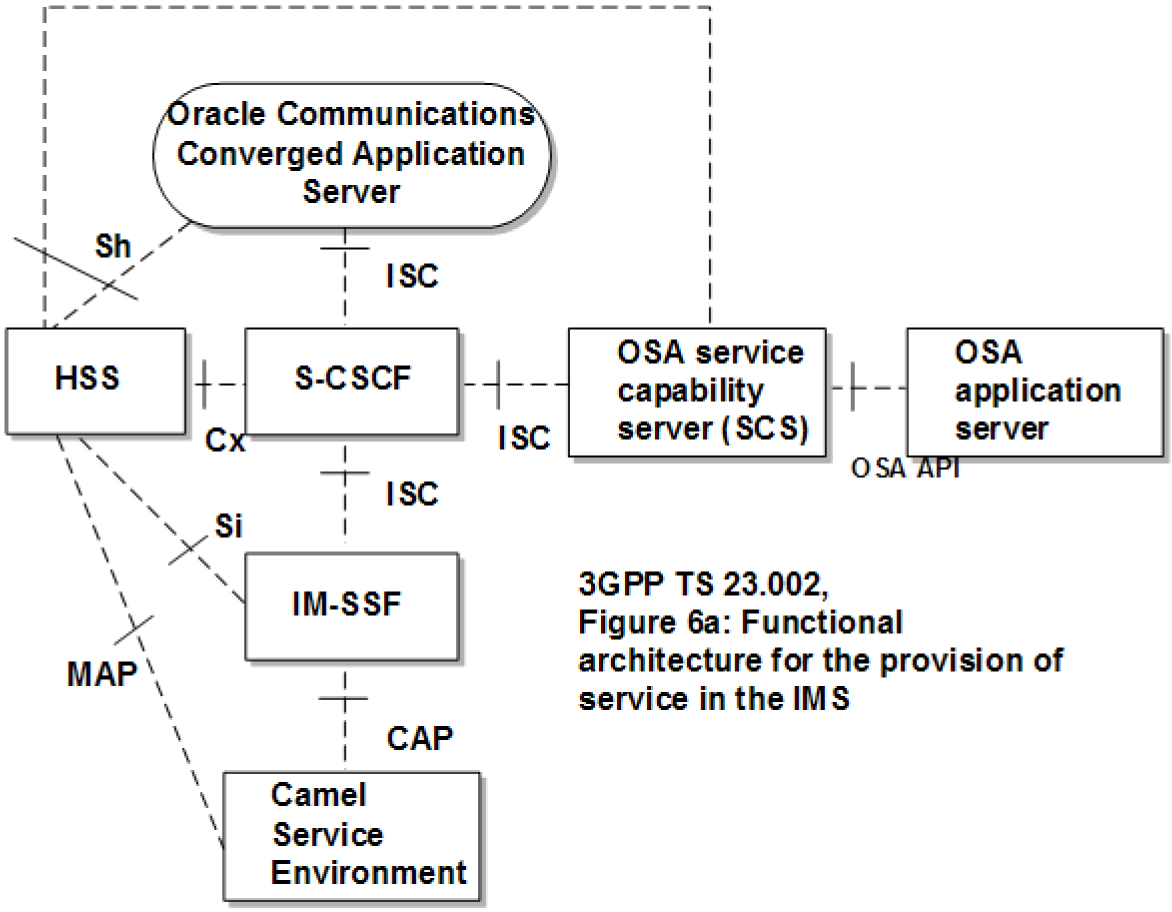
1 Overview of Converged Application Server Architecture

This chapter describes the Oracle Communications Converged Application Server architecture.

About Converged Application Server Architecture

Converged Application Server is a carrier-class Java EE application server that has been extended with support for the Session Initiation Protocol (SIP) and a number of operational enhancements that allow it to meet the demanding requirements of next-generation Internet Protocol-based communications networks. In a typical IMS deployment, Converged Application Server fills the role of the IMS SIP Application Server.

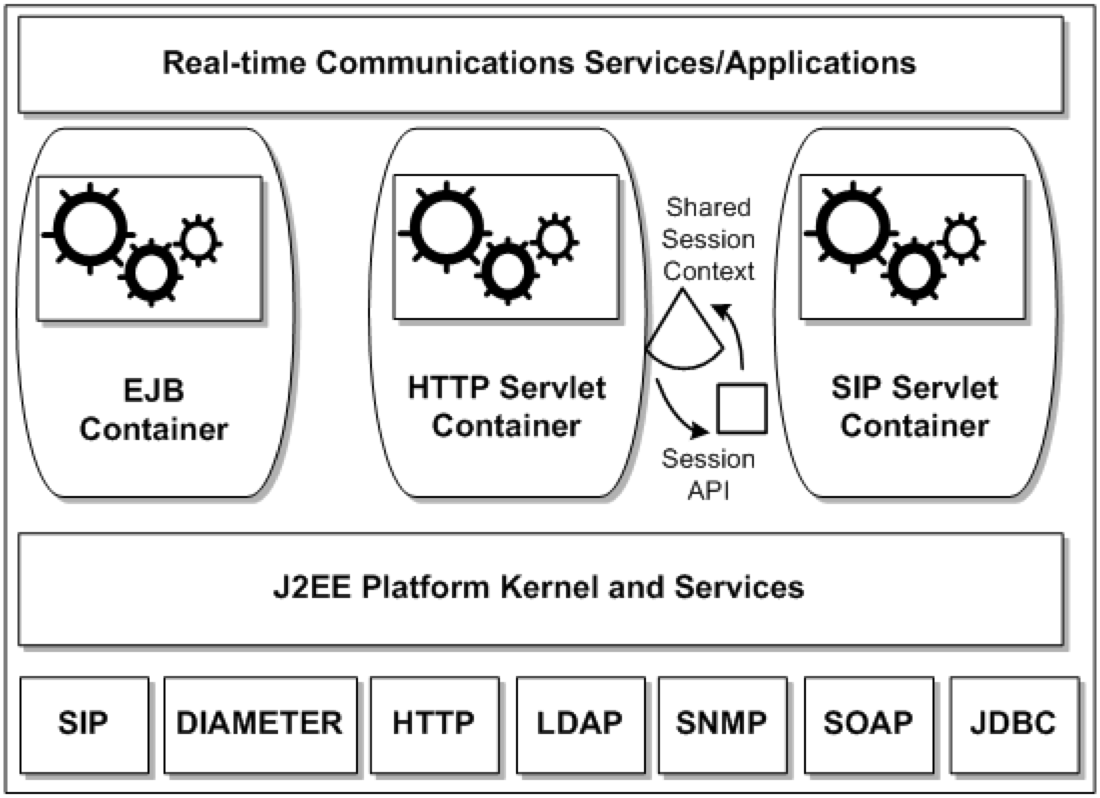
Figure 1–1 Converged Application Server in the IMS Service Architecture



The Converged Application Server implementation is based on parts of Oracle's widely deployed and time-tested Java EE-compliant WebLogic Server product. Converged Application Server supports all of the standard Oracle WebLogic Server programming interfaces and facilities such as JTA, JAF, JMS, JNDI, JDBC, and EJB. Converged Application Server also supports the protocols typically associated with a standards-compliant Java EE application server, including RMI over IIOP, HTTP 1.1, LDAP, SMTP, POP, IMAP, and SNMPv2.

Converged Application Server builds upon the base Java EE programming model by integrating a SIP Servlet Container that is compliant with the JSR-000116 SIP Servlet API specification. This "converged" container provides an execution environment for applications containing both HTTP and SIP protocol handling components, as well as other protocols such as Diameter.

Figure 1–2 Converged Application Server Extended Java EE for Next Generation Networks



The "SIP Stack" of Converged Application Server is fully integrated into the SIP Servlet container and is substantially more powerful and easier to use than a traditional protocol stack. The SIP Servlet API defines a higher layer of abstraction than simple protocol stacks provide and frees the developer from any concern for the mechanics of the SIP protocol itself.

Note: In this context "mechanics" refers to the syntactic validation of received requests, handling of transaction layer timers, generation of non application-related responses, generation of fully-formed SIP requests from request objects (which involves correct preparation of system headers and generation of syntactically correct SIP messages) and handling of lower-layer transport protocols (such as TCP, UDP or SCTP).

The Servlet container distributes request and response objects to components in a structured way, maintains awareness of the state of the larger, converged SIP and HTTP application session, and manages the end-to-end object lifecycle, including resource, transaction, and session state management. The converged SIP and HTTP container thereby frees the developer from much work (and opportunity for error) and allows deployed applications to inherit the high-availability, performance, and operational features provided by the robust Converged Application Server container implementation.

The SIP Servlet API greatly simplifies the task of implementing SIP User Agents, Proxies and Back-to-Back-User-Agents, and it narrows the developers exposure to operational concerns such as resource management, reliability, manageability and interaction between services. (See Chapter 2, "Developing SIP Applications with Converged Application Server," for more information.)

The SIP Servlet API is the ideal choice for exposing the full capabilities of the SIP protocol in a Java and Java EE middleware solution. No equivalent standard exists for defining an Application Programming Interface or programming model that is as well suited to meet both the needs of the application developer and the operational requirements of the service provider.

Converged Application Server also incorporates a number of architectural features that allow for its deployment as a highly-available, fault tolerant Single System Image cluster. The Converged Application Server cluster architecture is based on a multi-tier model in which a load balancer distributes SIP requests to a stateless Engine Tier (often referred to as “Engines”). The engine tier processes all signaling traffic and replicates transaction and session state information to SIP data tier “partitions”. Each partition may consist of one or more replicas distributed across servers or server blades (cluster members). This clustering capability, combined with the load balancer, transparently provides services with Telco-grade availability, scalability, and fault tolerance (session retention), ensuring that ongoing sessions are not affected by the failure of individual cluster members. A production deployment of Converged Application Server has no single point of failure.

2

Developing SIP Applications with Converged Application Server

This chapter describes the environment for developing applications with Oracle Communications Converged Application Server.

Overview of Developing SIP Applications with Converged Application Server

JSR 289: SIP Servlet API extends the basic concept of the Servlet, originally introduced as a programming model for implementation of applications which handle HTTP and the programmatic generation of HTML. The Servlet model is one of the most widely-known and used programming models in the Java community.

The SIP Servlet API specification describes not only the programming API but also the Servlet container function. The container is the Server (software) that hosts or “contains” applications written using the API. The SIP Servlet container hosts SIP applications. The container performs a number of SIP functions as specified by various RFCs, thus taking the burden off of the applications themselves. At the same time, the container exposes the application to SIP protocol messages through the SIP Servlet API. In this way, the application can perform various actions based on the SIP messages it receives from the container. Different applications can be coded and deployed to the container in order to provide various telecommunication or multimedia services.

Goals of the SIP Servlet API Specification

The sections that follow describe the primary goals of the SIP Servlet API specification.

SIP Protocol Support

The SIP Servlet API enables applications to perform a complete set of SIP Signaling functions. The SIP Protocol specification defines different types of high level SIP roles, namely User Agents (UAs) which include UA Clients, UA Servers, and Back to back user agents (B2BUAs). The SIP protocol also defines the roles of Proxies, Registrars, and Redirect Servers. The SIP Servlet API is a allows any of these roles to be coded as SIP Servlet application.

SIP is an extensible protocol, which is one of its strengths. Applications can extend the base protocol to add new features as necessary. In fact, there are a number of RFCs that define extensions to the base IETF RFC 326 SIP: Session Initiation Protocol. The SIP Servlet API is also designed to allow developers to easily extend functionality. This is accomplished by dividing up the SIP processing between the container functions the applications. Most of the base protocol processing is performed by the container, leaving some of the higher level tasks for the applications to perform. This clever division is what lends a great deal of power and flexibility to the SIP Servlet API.

Simplicity and Ease of Use

The SIP Servlet container handles “non-application-specific” complexity outside of the application code itself. Concerns like network connectivity, protocol transactions, dialog management and route processing are required by virtually all applications, and it would be enormously wasteful and error-prone to require each application to implement this support. With the SIP Servlet API, all of these tasks are managed by the container, leaving applications to provide higher level functions.

As an example, consider a SIP Proxy component:

1. A SIP Servlet within the SIP Servlet container receives a SIP request object and proxies it. A SIP Proxy must add its own Via header to the request; the header is required by the base SIP protocol to indicate which entities were traversed by the request. The Via header also stores the branch identifier which acts as the transaction identifier.

   Because the maintenance of transactions and their associated state machine is maintained by the containers, it is the container that actually inserts the via headers to the Request.

2. The downstream SIP entity which next receives the request sends the response back along the path built up by the SIP entities in the path of the request that have inserted themselves into the `via` or `record-route` headers.

3. The container gets the response, removes the via header it inserted in the original request and then processes the response. The application code does not need to manage the `Via` header at all, which makes the life of application developer much easier.

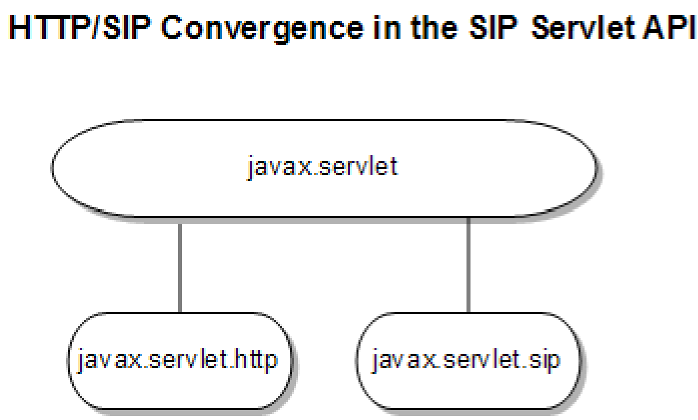
There are many cases in which the SIP Servlet container handles this sort of mundane, but essential, protocol detail.

Converged Applications

The SIP Servlet API specification is closely aligned with the Java EE specifications, and it is expected that containers that host SIP Servlet applications also make Java EE features available to developers. The most notable of these features is the HTTP Servlet container. There are many use cases in which a converged application, using both SIP and HTTP functions, is required, from conferencing and click-to-call applications to Presence and User Agent Configuration Management applications. Converged applications can also combine other protocols such as Diameter to perform advanced functions such as modifying subscriber profile data.

Figure 2–1 illustrates that `javax.servlet.http` and `javax.servlet.sip` converge in the SIP Servlet API.

Figure 2–1 HTTP/SIP Convergence in the SIP Servlet API



Application Composition

The SIP Servlet API enables multiple applications to execute on the same request or response, independently of one another. This is another very powerful feature of the SIP Servlet API. The promise is that application developers are able to write applications providing features that are independent of each other, but can be deployed to the same host SIP Servlet container. The applications can be “composed” (or sequenced) to provide a service on a call. This composition is facilitated by the container. See Appendix A, "SIP Servlet API Service Invocation," for more information.

Highly Reliable Implementation

Application data stored in container-managed session objects can benefit from replication and failover. Almost all applications that perform some useful functions require some state between different Requests and Responses. Some state information is mandated by the SIP protocol itself, such as the transaction state machine with its Server and Client Transactions, and the Dialog state machine.

The container also has a notion of message context which encapsulates the SIP level state, and the concept of Sessions, which are the SIP Servlet API constructs. Applications can save their own state in the Session objects maintained by the container. A carrier-grade container will replicate this state such that the call becomes fault tolerant of a container instance, as is done in Converged Application Server.

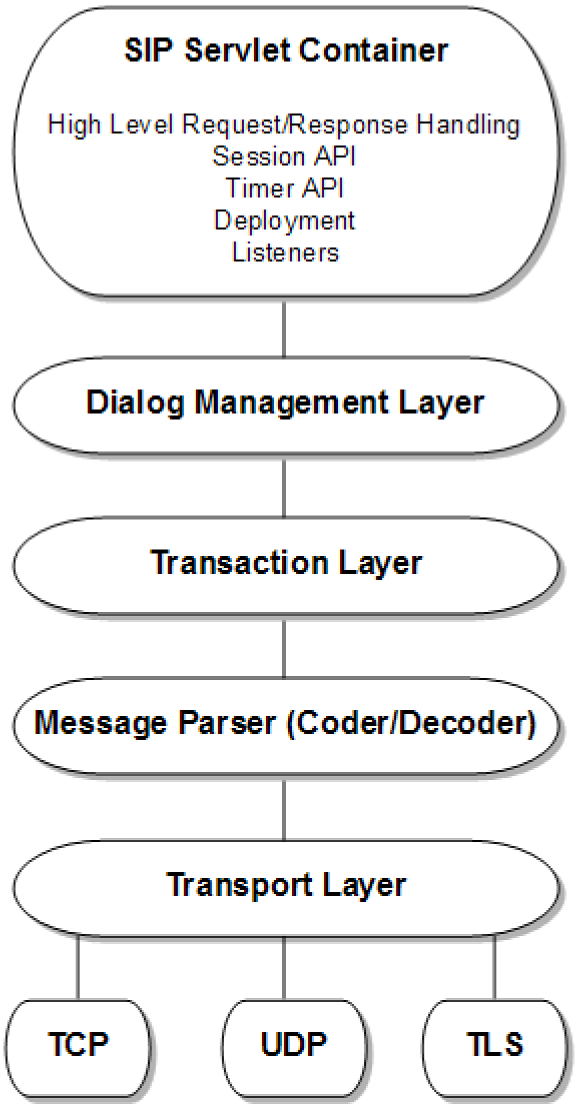
Overview of the SIP Servlet Container

Figure 2–2 shows the logical layers of a Converged Application Server SIP Servlet Container. The five layers shown from the bottom are what are known as the SIP stack, the functionality of which is defined in RFC 3261 and the associated RFCs that extend the base protocol.

SIP, being a transaction-based protocol, has a well-defined transaction layer. SIP requests are always followed by one or more provisional Responses and one final response, with the exception of the ACK which has no response. The transaction machinery is designed to keep track of the provisional and final responses.

Figure 2–2 shows the message processing layers in the Converged Application Server SIP Servlet container which are the following from top to bottom: Dialog Management Layer, Transaction Layer, Message Parser, the Transport Layer, and the bottom layer comprising of TCP, UDP, and TLS.

Figure 2–2 Message Processing Layers in the Converged Application Server SIP Servlet Container



SIP Dialog Handling

A dialog is a point-to-point session between two SIP endpoints that is uniquely identified by a dialog identifier. Not all SIP requests create dialogs. However, the ones that do create dialogs have a well-defined mechanism of establishing and tearing down the dialog (INVITE, SUBSCRIBE/NOTIFY, REFER).

The SIP stack shown in this diagram is not strictly in accordance with RFC 3261. It differs from the specification in that there is a layer called Transaction User (TU) above the Transaction layer, and the dialog management layer is not explicitly a layer in 3261. The “Dialog layer” is a very visible constituent of a SIP Servlet container because the dialogs correspond roughly to the `SipSession` objects. In Figure 2–2, the TU layer is actually split between the Dialog management layer and the big Container block.

The primary purpose of the Container is to host SIP Servlet applications that are written to the container's SIP Servlet API implementation. It exposes objects like `SipServletRequest`, `SipServletResponse`, different types of Sessions, facilities such as Timer, Logging, and so forth.

Although SIP is a human-readable, text-based protocol, and is well-defined in RFC 3261, writing SIP applications can be a challenging task. The SIP Servlet API is designed to make it very easy for application developers to build SIP applications. While the SIP Servlet API allows access to all the headers present in a SIP Request, it does not require applications to understand or modify all of them for correct protocol behavior. Also, there are some headers that are strictly off limits for applications. The SIP Servlet API defines the so-called "system headers" which are to be managed only by the container. These headers include `From`, `To`, `Call-ID`, `CSeq`, `Via`, `Route` (except through `pushRoute`), `Record-Route`, and `Contact`. Applications can add attributes to the Record-Route header and Contact header fields in all request messages, as well as 3xx and 485 responses. Additionally, for containers such as Converged Application Server that implement the reliable provisional responses extension, RAck and RSeq are also considered to be system headers. The system header management performed by the container offloads a tremendous amount of complexity from applications.

The `From`, `To`, `Call-ID`, and `CSeq` message headers collectively identify a given SIP dialog. The SIP Servlet container keeps track of the dialog state and dialog-related data for the hosted applications. The SIP Servlet API container is responsible for managing `Record-Route`, `Contact`, and `Via` headers because the network listen points, failure management, multi-homing, transport switching, and so forth are also handled by the container. Applications can participate in the routing decisions of a Request emanating from the container by explicitly modifying Request-URI or adding Route headers with pushRoute. As a result, applications have no responsibility for resource management. The SIP Servlet API draws heavily from Java EE standardization and common practices, such as the declarative usage of container features like security, mapping, environment resources, and so forth.

Perhaps the greatest advantage of the SIP Servlet API is the API itself. The SIP Servlet API abstracts a large number of complex SIP tasks behind intuitive constructs. The Proxy interface, representing the proxy functionality in SIP, is an excellent example. A proxy can:

- Be stateful or stateless.
- Recurse automatically (send Requests automatically) on getting a 3xx class response to the Contact address(es) in the Response.
- Use Record-Route to ensure that subsequent requests also go through it.
- Act as a forking proxy to proxy to multiple destinations, either in parallel or in sequence.

With the SIP Servlet API, all of these options are simple attributes of the Proxy object. The container-managed Proxy deals with all low level details like finding a target set (based on Request-URI or Route headers), applying RFC rules if a strict router is upstream or downstream, creating multiple client transactions, correlating responses, choosing the best response, and so forth.

Using the SIP Servlet API

This section describes additional important interfaces and constructs of the SIP Servlet API, and includes examples.

The SipServlet Object

The SipServlet class extends the `GenericServlet` class in the servlet base package. The service method dispatches the SIP message to either `doRequest()` or `doResponse()`, and in turn the requests are directed to the doXXX methods for Requests such as `doInvite`, `doSubscribe`, and so forth, or to doXXX methods for Responses such as `doSuccessResponse` and `doErrorResponse`.

The servlet-mapping element defined in the deployment descriptor can define the rule that MUST be satisfied before invoking a particular Servlet. The mapping rules have a well-defined grammar in JSR 116. Example 2–1 shows a mapping that invokes a Servlet only if the Request is an INVITE and the host part of the Request-URI contains the string "oracle.com". See Appendix A, "SIP Servlet API Service Invocation," for more information on servlet mapping rules.

Example 2–1 Example Servlet Mapping Rule

```
pattern
  <and>
    <equal>
      <var>request.method</var>
      <value>INVITE</value>
    </equal>
    <contains ignore-case="true">
      <var>request.from.uri.host</var>
      <value>oracle.com</value>
    </contains>
  </and>
</pattern>
```

There is normally only one `SipServlet` object accessed by concurrent Requests, so it is not a place to define any call- or session- specific data structure. The doXXX methods in the application generally implement the business logic for a given request. Consider Example 2–2.

Example 2–2 Example SIP Servlet

```
package test;
import javax.servlet.sip.SipServlet;
import javax.servlet.sip.SipServletRequest;
import java.io.IOException;
public class SimpleUasServlet extends SipServlet {
  protected void doInvite(SipServletRequest req)
     throws IOException {
    req.createResponse(180).send();
    req.createResponse(200).send();
  }
  protected void doBye(SipServletRequest req) throws IOException {
    req.createResponse(200).send();
    req.getApplicationSession().invalidate();
  }
}
```

Example 2–2 shows a simple UAS Servlet that is invoked on an incoming INVITE Request (triggered by a rule similar to the one defined in Example 2–1). The container invokes the application by invoking the `doInvite` method. The application chooses to send a 180 Response (line 8) followed by a 200 Response (line 9). The application does nothing with the ACK, which would be sent by the UAC. In this case the container receives the ACK and silently ignores it. If it were a stateful proxy it would have proxied it.

SIP Factory

As its name suggests, this class is used to create various SIP Servlet API objects such as `Request`, `SipApplicationSession`, `Addresses`, and so forth. An application acting as a UA can use it to create a new Request. Requests created through the factory have a new Call-ID (with the exception of a particular method for B2BUAs in which the application can chose to re-use the existing Call-ID on the upstream leg) and do not have a tag in the To header. The Factory object can be retrieved using the `javax.servlet.sip.SipFactory` attribute on the ServletContext.

See the "findme" example installed with Converged Application Server for an example of obtaining a factory object using `SipFactory`.

SIP Messages

There are two classes of SIP messages: `SipServletRequest` and `SipServletResponse`. These classes respectively represent SIP Requests (INVITE, ACK, INFO, and so forth) and Responses (1xx, 2xx, and so forth). Messages are delivered to the application through various doXXX methods defined in the `SipServlet` class.

SIP is an asynchronous protocol and therefore it is not obligatory for an application to respond to a Request when the `doRequest`(doXXX) method is invoked. The application may respond to the Request at a later stage, because they have access to the original Request object.

Both the `SipServletRequest` and `SipServletResponse` objects are derived from the base `SipServletMessage` object, which provides some common accessor/mutator methods such as `getHeader()`, `getContent()`, and `setContent()`. The `SipServletRequest` defines many useful methods for Request processing:

- `SipServletRequest.createResponse()` creates an instance of the `SipServletResponse` object. This represents the Response to the Request that was used to create it. Similarly, `SipServletRequest.createCancel()` creates a CANCEL Request to a previously sent Request.

  Note: The CANCEL is sent if the UAC decides to not proceed with the call if it has not received a response to the original request. Sending a CANCEL if you have received a 200 response or not received a 100 response would be wrong protocol behavior, luckily the SIP Servlet API steps up to rescue here too. The UAC application can create and send a CANCEL oblivious to these details. The container makes sure that the CANCEL is sent out only if a 1xx class response is received and any response >200 is not received.

- `SipServletRequest.getProxy()` returns the associated Proxy object to enable an application to perform proxy operations.

- `SipServletRequest.pushRoute(SipURI)` enables a UAC or a proxy to route the request through a server identified by the SipURI. The effect of this method is to add a Route header to the request at the top of the Route header list.

Another method of interest is `SipServletRequest.isInitial()`. It is important to understand the concept of initial and subsequent requests, because an application may treat each one differently. For example, if an application receives a Re-INVITE request, it is delivered to the Servlet's `doInvite()` method, but the `isInitial()` method returns `false`.

Initial requests are usually requests outside of an established dialog, of which the container has no information. Upon receiving an initial Request, the container determines which application should be invoked; this may involve looking up the Servlet-mapping rules. Some Requests create dialogs, so any Request received after a dialog is established falls into the category of a "subsequent" Request. Closely-linked with the dialog construct in SIP is the `SipSession` object (see "SipSession").

In the `SipServletResponse` object, one particular method of interest is `createAck()`. `createAck()` creates an ACK Request on a 2xx Response received for the INVITE transaction. ACKs for non-2xx responses of the INVITE transaction are created by the container itself.

SipSession

The `SipSession` roughly corresponds to a SIP dialog. For UAs the session maintains the dialog state as specified by the RFC, in order to correctly create a subsequent request in a dialog. If an application is acting as a UA (a UAC or a B2BUA), and after having processed an initial request wants to send out a subsequent request in a dialog (such as a Re-INVITE or BYE), it must use `SipSession.createRequest()` rather than one of `SipFactory` methods. Using a factory method would result in requests being created "out of dialog".

The `SipSession` is also a place for an application to store any session-specific state that it requires. An application can set or unset attributes on the `SipSession` object, and these attributes are made available to the application over multiple invocations.

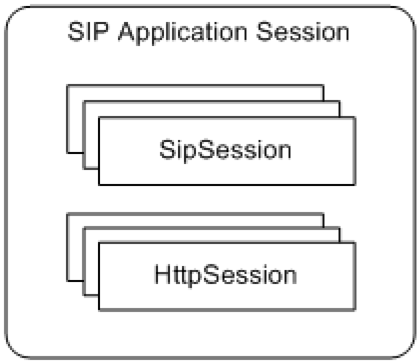
`SipSession` also provides the `SipSession.setHandler(String nameOfAServlet)` method, which assigns a particular Servlet in the application to receive subsequent Requests for that `SipSession`.

SipApplicationSession

The `SipApplicationSession` logically represents an instance of the application itself. An application may have one or more protocol sessions associated with it, and these protocol sessions may be of type `SipSession` or `HttpSession` as of JSR 116. Applications can also store application-wide data as an attribute of the `SipApplicationSession`.

Any attribute set on a `SipApplicationSession` object or its associated `SipSession` is visible only to that particular application. The SIP Servlet API defines a mechanism by which more than one application can be invoked on the same call. This feature is known as application composition. `SipApplicationSession` provides a `getSessions()` method that returns the protocol sessions associated with the application session. Figure 2–3 shows the containment hierarchy of the different sessions in the SIP Servlet API.

Figure 2–3 SipApplicationSession



The `encodeUri(URI)` method in the `SipApplicationSession` interface is of particular interest. This method encodes the SipApplication identifier with the URI specified in the argument. If the container receives a new request with this encoded URI, even if on a different call, it associates the encoded `SipApplicationSession` with this Request. This method can link two disparate calls, and it can be used in a variety of other ways. `SipApplicationSession` is also associated with application session timers (see "Application Timers").

Application Timers

The SIP Servlet API provides a timer service that applications can use. The `TimerService` interface can be retrieved using a `ServletContext` attribute, and it defines a `createTimer(SipApplicationSession appSession, long delay, boolean isPersistent, java.io.Serializable info)` method to start an application-level timer.

The `SipApplicationSession` is implicitly associated with application-level timers. When a timer fires, the container invokes an application-defined `TimerListener` and passes it the `ServletTimer` object. The listener can use the `ServletTimer` object to retrieve the `SipApplicationSession`, which provides the correct context for the timer's expiry.

SIP Servlet Application Example: Converged SIP and HTTP Application

In terms of the SIP Servlet API, a converged application is one that involves more than one protocol, in this case SIP and HTTP. Example 2–3 presents an example of a simple JSP page which can be accessed through an HTTP URL.

Example JSP Showing HTTP and SIP Servlet Interaction

Example 2–3 Example JSP Showing HTTP and SIP Interaction

```
<html>
<body>
 <%
  if (request.getMethod().equals("POST")) {
   javax.servlet.sip.SipFactory factory =
     (javax.servlet.sip.SipFactory)
application.getAttribute(javax.servlet.sip.SipServlet.SIP_FACTORY);
   javax.servlet.sip.SipApplicationSession appSession =
      factory.createApplicationSession();
   javax.servlet.sip.Address to =
      factory.createAddress("sip:localhost:5080");
```

```
11:   javax.servlet.sip.Address from =
12:     factory.createAddress("sip:localhost:5060");
13:   javax.servlet.sip.SipServletRequest invite =
14:      factory.createRequest(appSession, "INVITE", from, to);
15:   javax.servlet.sip.SipSession sess = invite.getSession(true);
16:     sess.setHandler("sipClickToDial");
17:   //invite.setContent(content, contentType);
18:   invite.send();
19:  }
20:%>
21:<p>
22:Message sent ...
23:</p>
24:</body>
25:</html>
```

The JSP shown in Example 2–3 would need to be packaged in the same application as a SIP Servlet. The entire application is a skeleton of a click-to-dial application (called `sipClickToDial`), where by clicking on a Web page you initiate a SIP call.

The HTTP Servlet creates a SIP Request from a factory and sends it to a SIP URI. When an HTTP POST Request is sent to the HTTP Servlet it obtains the `SipFactory` on line 5-6. Next, it creates an application session (line 7-8). The application session is the center piece for all of the application's SIP and HTTP interactions. The overall purpose is to send out a SIP Request, which is done in lines 13-14, but first the application creates the `From` and `To` headers to be used when forming the INVITE request.

On line 16 the application assigns a handler to the `SipSession` that is associated with the INVITE Request that was created, and this ensures that the Response sent by a UAS that receives the request is dispatched to a SIP Servlet for processing.

SIP Servlet Application Example: SUBSCRIBE and NOTIFY

In the example shown in Example 2–4 below, the application receives a SUBSCRIBE Request and sends out a NOTIFY Request. The application then waits for the notification recipient for three seconds, and if does not receive a success response (a 2xx class response), then it may take some other action (for example, log a message).

Example 2–4 Example of SUBSCRIBE and NOTIFY Handling

```
public class Sample_TimerServlet extends SipServlet
  implements TimerListener {
  private TimerService timerService;
  private static String TIMER_ID = "NOTIFY_TIMEOUT_TIMER";
  public void init() throws ServletException {
    try {
      timerService =
(TimerService)getServletContext().getAttribute
      ("javax.servlet.sip.TimerService");
    }
    catch(Exception e) {
      log ("Exception initializing the servlet "+ e);
    }
  }
  protected void doSubscribe(SipServletRequest req)
  throws ServletException, IOException {
    req.createResponse(200).send();
    req.getSession().createRequest("NOTIFY").send();
    ServletTimer notifyTimeoutTimer =
      timerService.createTimer(req.getApplicationSession(), 3000,
```

```
21:  false, null);
22:    req.getApplicationSession().setAttribute(TIMER_ID,
23:notifyTimeoutTimer);
24:  }
25:  protected void doSuccessResponse(SipServletResponse res)
26:  throws javax.servlet.ServletException, java.io.IOException {
27:    if (res.getMethod().equals("NOTIFY")) {
28:      ServletTimer notifyTimeoutTimer =
29:  (ServletTimer)(res.getApplicationSession().getAttribute(TIMER_ID));
30:      if (notifyTimeoutTimer != null) {
31:        notifyTimeoutTimer.cancel();
32:        res.getApplicationSession().removeAttribute(TIMER_ID);
33:      }
34:    }
35:  }
36:  public void timeout(ServletTimer timer) {
37:    // This indicates that the timer has fired because a 200 to
38:    // NOTIFY was not received. Here you can take any timeout
39:    // action.
40:    // .........
41:    timer.getApplicationSession().removeAttribute
("NOTIFY_TIMEOUT_TIMER");
42:  }
43:}
```

In Example 2–4, the Servlet itself implements `TimerListener` so that it will be notified of the timeout. The example starts by obtaining the `TimerService` from the `ServletContext` in lines 7-9. The timer is then set for 3000 ms (3 seconds) upon receiving the SUBSCRIBE request on line 20. Note that the timer could be set at any stage. There is also an option to attach an object to the timer. The object could be used as an identifier or an invokable message at a later stage. This sample simply associates the timer with a literal.

After sending the NOTIFY the application creates the timer and saves its reference in the `SipApplicationSession` for later use on line 22.

If the application receives a 200 response to the NOTIFY, it can then extract the timer reference and cancel the timer (line 25). However, if no response is received in 3 seconds, then the timer fires and the container calls the `timeout()` callback method (line 36).

Converged Application Server Profile API

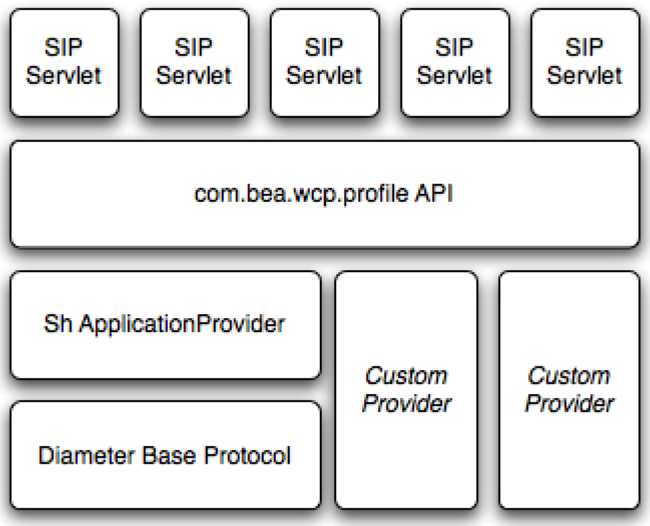
The IMS specification defines the Sh interface as the method of communication between the Application Server (AS) function and the Home Subscriber Server (HSS), or between multiple IMS Application Servers. The AS uses the Sh interface in two basic ways:

- To query or update a user's data stored on the HSS
- To subscribe to and receive notifications when a user's data changes on the HSS

The user data available to an AS may be defined by a service running on the AS (repository data), or it may be a subset of the user's IMS profile data hosted on the HSS. The Sh interface specification, 3GPP TS 29.328 V5.11.0, defines the IMS profile data that can be queried and updated via Sh. All user data accessible via the Sh interface is presented as an XML document with the schema defined in 3GPP TS 29.328.

The IMS Sh interface is implemented as a provider to the base Diameter protocol support in Converged Application Server. The provider 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). Subscriptions and notifications for changed profile data are managed by implementing a profile listener interface in a SIP Servlet.

Figure 2–4 Profile Service API and Sh Provider Implementation



Converged Application Server includes only a single provider for the Sh interface. Future versions of Converged Application Server may include new providers to support additional interfaces defined in the IMS specification. Applications using the profile service API will be able to use additional providers as they are made available.

Using Document Keys for Application-Managed Profile Data

Servlets that manage profile data can explicitly obtain an Sh XML document from a factory using a key, and then work with the document using DOM.

The document selector key identifies the XML document to be retrieved by a Diameter interface, and uses the format *protocol*`://`*uri*`/reference_type[/`*access_key*`]`.

Table 2–1 summarizes the required document selector elements for each type of Sh data reference request.

Table 2–1 Summary of Document Selector Elements for Sh Data Reference Requests

Data Reference Type	Required Document Selector Elements	Example Document Selector
RepositoryData	sh://uri/reference_type/Service-Indication	sh://sip:user@oracle.com/RepositoryData/Call Screening/
IMSPublicIdentity	sh://uri/reference_type/[Identity-Set] where Identity-Set is one of: ■ All-Identities ■ Registered-Identities ■ Implicit-Identities	sh://sip:user@oracle.com/IMSPublicIdentity/Registered-Identities
IMSUserState	sh://uri/reference_type	sh://sip:user@oracle.com/IMSUserState/
S-CSCFName	sh://uri/reference_type	sh://sip:user@oracle.com/S-CSCFName/
InitialFilterCriteria	sh://uri/reference_type/Server-Name	sh://sip:user@oracle.com/InitialFilterCriteria/www.oracle.com/
LocationInformation	sh://uri/reference_type/(CS-Domain | PS-Domain)	sh://sip:user@oracle.com/LocationInformation/CS-Domain/
UserState	sh://uri/reference_type/(CS-Domain | PS-Domain)	sh://sip:user@oracle.com/UserState/PS-Domain/
Charging information	sh://uri/reference_type	sh://sip:user@oracle.com/Charging information/
MSISDN	sh://uri/reference_type	sh://sip:user@oracle.com/MSISDN/

Converged Application Server provides a helper class, `com.bea.wcp.profile.ProfileService`, to help you easily retrieve a profile data document. The `getDocument()` method takes a constructed document key, and returns a read-only `org.w3c.dom.Document` object. To modify the document, you make and edit a copy, then send the modified document and key as arguments to the `putDocument()` method.

See "Using the Profile Service API (Diameter Sh Interface)" in *Converged Application Server Diameter Application Development Guide* for more information.

Monitoring Profile Data

The IMS Sh interface enables applications to receive automatic notifications when a subscriber's profile data changes. Converged Application Server provides an easy-to-use API for managing profile data subscriptions. A SIP Servlet registers to receive notifications by implementing the `com.bea.wcp.profile.ProfileListener` interface, which consists of a single update method that is automatically invoked when a change occurs to profile to which the Servlet is subscribed. Notifications are not sent if that same Servlet modifies the profile information (for example, if a user modifies their own profile data).

> **Note:** In a replicated environment, Diameter relay nodes always attempt to push notifications directly to the engine tier server that subscribed for profile updates. If that engine tier server is unavailable, another server in the engine tier cluster is chosen to receive the notification. This model succeeds because session information is stored in the SIP data tier, rather than the engine tier.

Actual subscriptions are managed using the subscribe method of the **com.bea.wcp.profile.ProfileService** helper class. The subscribe method requires that you supply the current **SipApplicationSession** and the key for the profile data document you want to monitor. See "Using Document Keys for Application-Managed Profile Data" for more information.

Applications can cancel subscriptions by calling ProfileSubscription.cancel(). Also, pending subscriptions for an application are automatically cancelled if the application session is terminated.

Example 2–5 shows sample code for a Servlet that implements the ProfileListener interface.

Example 2–5 Sample Servlet Implementing ProfileListener Interface

```
package demo;
    import com.bea.wcp.profile.*;
    import javax.servlet.sip.SipServletRequest;
    import javax.servlet.sip.SipServlet;
    import org.w3c.dom.Document;
    import java.io.IOException;
    public class MyServlet extends SipServlet implements ProfileListener {
      private ProfileService psvc;
      public void init() {
        psvc = (ProfileService)
getServletContext().getAttribute(ProfileService.PROFILE_SERVICE);
      }
      protected void doInvite(SipServletRequest req) throws IOException {
        String docSel = "sh://" + req.getTo() + "/IMSUserState/";
        // Subscribe to profile data.
        psvc.subscribe(req.getApplicationSession(), docSel, null);
}
      public void update(ProfileSubscription ps, Document document) {
        System.out.println("IMSUserState updated: " + ps.getDocumentSelector());
      }
    }
```

The ProfileListener interface is handled similar to the TimerService provided by JSR 116 for application timers. Multiple Servlets in an application may implement the ProfileListener interface, but only one Servlet may act as a listener. The SIP deployment descriptor for the application must designate the profile listener class in the set of listeners as shown in Example 2–6.

Example 2–6 Declaring a ProfileListener

```
<listener>
<listener-class>com.foo.MyProfileListener</listener-class>
Declaring a ProfileListener
</listener>
<listener>
  <listener-class>com.foo.MyProfileListener</listener-class>
</listener>
```

Converged Application Server Software Development Kit

The Converged Application Server SDK consists of the Converged Application Server executable and a selection of example applications available as source code and deployable binaries. The Converged Application Server SDK may be executed on any standard Windows or Linux workstation. When used in conjunction with an IDE, common tasks such as the writing and modification of Java code, setting of break-points, tracing and profiling are easily performed.

It is possible to use Converged Application Server in conjunction with virtually any of the popular development tools commonly used to develop Java and Java EE applications.

3

Converged Application Server in the Network

This chapter describes how Oracle Communications Converged Application Server functions in a service provider network:

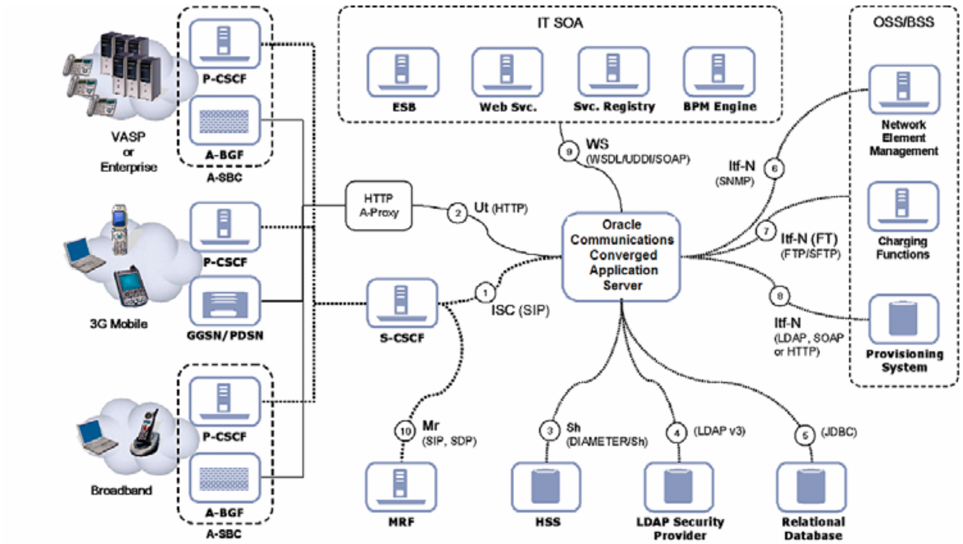
- Overview of Converged Application Server in a Typical Service Provider Network
- SIP and IMS Service Control
- HTTP User Interface
- Service and Subscriber Data and Authentication
- Proxy Registrar
- Management Interfaces
- Media Server Control
- Charging and Billing
- Security

Overview of Converged Application Server in a Typical Service Provider Network

Converged Application Server can be deployed in 3GPP R6 compliant IMS networks as well as in non-IMS networks. Converged Application Server can interoperate with a number of network functions regardless of which applications or functions it hosts.

"3GPP R6 Specification Conformance" on page 5-12 outlines the Converged Application Server's conformance to the requirements introduced in the 3GPP Release 6 specifications.

Figure 3–1 Converged Application Server deployed in a typical service provider network



SIP and IMS Service Control

The SIP interface between the Serving CSCF and the IMS SIP Application Server (AS) is defined as the IMS Service Control (ISC) reference point. Although ISC is generally compliant with the SIP protocol as defined by the IETF, it introduces several specific procedures and transport layer requirements. SIP usage is often described as the "3GPP SIP Profile."

The ISC reference point does not require that the AS or Serving CSCF add any particular attribute or value to a request or response beyond the standard behavior of a SIP protocol entity. There are, however, a number of SIP methods and headers that are relevant to many of the services that are deployed on the IMS (SIP) AS. In order for the IMS SIP AS to "fully" comply with all of the 3GPP R5 and R6 specifications, many IETF RFCs and drafts would have to be supported. However, it is not reasonable to characterize this as "ISC compliance" because ISC specifically addresses the relationship between the IMS (SIP) AS and the Serving CSCF. From this perspective, ISC compliance is relatively straightforward and is minimally reflected in "Procedures at the AS" defined in 3GPP TS 24.229: "IP Multimedia Call Control Protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3 (Release 6).

From the perspective of the Converged Application Server, the Serving CSCF is a SIP Proxy and/or User Agent (in the case of the Registration Event Package and third-party registration messages) and is the SIP Application Server's default gateway for SIP requests when the AS instantiates a User Agent Client.

ISC and the 3GPP SIP Profile

The 3GPP requires SIP to be used in a more restricted manner than the IETF specs allow, and also requires a number of additional SIP headers. This use of SIP is often referred to as the "3GPP SIP Profile."

The Converged Application Server SIP Servlet Container provides automated management of session objects, Servlet lifecycle, security, OAM and other functions that are not clearly within the scope of an application's business logic. The SIP Servlet Container allows applications to handle (send/receive) SIP messages with non-standard methods or headers—the container is concerned only with the validation of message syntax, and with the protocol transaction layer.

Converged Application Server uses certain p-headers directly. For example, p-asserted-identity is used as an assertion of identity within the Converged Application Server security framework. Other headers, like the 3GPP p-charging-vector or p-charging-function-address, are relevant only within the scope of the application and have no container-level implications.

Converged Application Server does not programmatically force applications to be compliant with the 3GPP SIP Profile, although applications deployed on Converged Application Server may comply with the SIP Profile as necessary.

AS Session Case Determination Requirement of ISC

When requests are sent to an IMS SIP Application Server by the S-CSCF, the SIP AS is generally required to determine the session case (originating, terminating, or terminating unregistered) of the request, either implicitly or explicitly.

Converged Application Server provides several ways of determining the session case for the request. There are three mechanisms described in the 3GPP standardization that an IMS (SIP) AS may use to make this determination:

- Session Case Specific Addresses (e.g. sip:sessioncase_as01.operator.net or sip:as01.operator.net:49494)
- Tokens in the “User Part” of the Request URI (e.g. sip:token@as01.operator.net)
- Request URI Parameters (e.g. sip:as01.operator.net;parameter)

See "3GPP TS 24.229: IP Multimedia Call Control Protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3 (Release 6)" for more information.

The choice of which mechanism to use is at the discretion of both the Communications Service Provider and the SIP Servlet application deployer. The SIP Servlet API relies on a deployment descriptor file that is packaged with the SIP Servlet Application archive file when it is created. The descriptor explicitly indicates the Service Trigger Points that will be used by the SIP Servlet Container to determine which SIP Servlets to invoke. These Service Trigger Points are sufficient to support any of the methods described above for determining the session case of the request.

See Appendix A, "SIP Servlet API Service Invocation" for a more detailed description of the Service Trigger Points supported by Converged Application Server.

Transport Layer Issues Related to ISC

The 3GPP Release 6 specifications mandate the use of IPv6 (see IETF RFC 2460: Internet Protocol, Version 6 (IPv6) Specification) for all interfaces, including ISC. Converged Application Server also supports IPv6.

When using TCP, Converged Application Server does not arbitrarily create new connections for each SIP Transaction or Dialog. By default, responses to SIP requests are returned using the connection on which the request was received. If a TCP connection fails, Converged Application Server establishes a new TCP connection to the target host. This may mean that responses to SIP requests are returned using TCP connections that are different from the connection over which the request was sent. Although this conforms to the current best practice and to "IETF RFC 3261: SIP: Session Initiation Protocol," Oracle has discovered that many SIP products on the market demonstrate non-compliant behaviors with regard to handling OSI layer 3 protocols.

Although it is not normally the case that Converged Application Server is deployed directly facing end-user SIP devices, it is important to understand the impact this behavior might have in such cases. When interacting with SIP endpoints on the public Internet, TCP connections are often kept alive indefinitely as a means of overcoming Network Address Translation (NAT) limitations in many typical broadband routers and residential gateways.

Converged Application Server does not provide an Application Layer Gateway (ALG) capability, and it is presumed that such capabilities are provided by a standard Session Border Control function.

HTTP User Interface

The 3GPP reference point associated with the HTTP interface provided by Converged Application Server is “Ut”. This interface is primarily used for three purposes:

- As a Web-based User Interface for customer self-care and service configuration, potentially using HTML, XHTML or other presentation technologies.
- To support content indirection.
- To support XML Configuration Access Protocol (XCAP), required by Presence and Conference Control Protocol.

Converged Application Server provides HTTP support through its HTTP Servlet Container. Application developers may implement applications or components that support any or all of the above use cases for the “Ut” reference point.

Service and Subscriber Data and Authentication

Converged Application Server supports the Sh reference point used to interact with the Home Subscriber Server (HSS) as the principal provider of IMS Profile data associated with the Public Identity of the network user or subscriber. In many cases, standard LDAP directory servers or relational databases are also used as supplementary resources for service or subscriber data. These may also be accessed via standard interfaces supported by Converged Application Server.

In many deployments, and for certain types of services such as Presence or media repositories, subscriber and service data can be accessed using other means. These include LDAP, HTTP, or access to relational databases.

In non-IMS deployments, the security provider may also be a standard directory accessed via Lightweight Directory access Protocol (LDAP) or access to a relational database using a database-specific interface. Most major commercial relational databases provide Java Database Connectivity (JDBC). A number of high-performance and fault-tolerant JDBC drivers are available commercially for use with Converged Application Server.

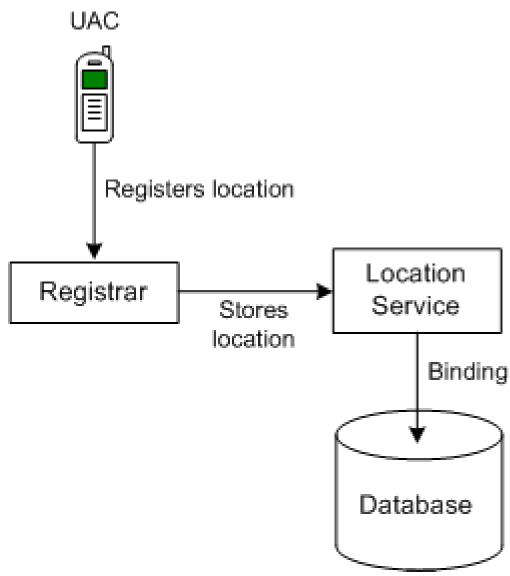
Proxy Registrar

The Converged Application Server Proxy Registrar implements the proxy and registrar functions described in RFC 3261. The Proxy Registrar combines the functionality of a SIP proxy server and registrar. Its main tasks include registering subscribers, looking up subscriber locations, and proxying requests onward. The Proxy Registrar is an optional component.

The Proxy Registrar's registrar function processes the REGISTER requests from user agent clients (UACs) and uses a location service to store a binding (that is, an association) between a user's address of record (AOR) and one or more contact addresses, typically the IP addresses of the UACs. The To header field of the REGISTER SIP message sent by a UA contains the address of record whose registration is to be created, queried, or modified and the CONTACT header field contains the corresponding contact addresses. The bindings between the AOR and the contact addresses are persistently stored in a database. The supported databases are Oracle 11g and MySQL 5.4.

Figure 3–2 illustrates the registration flow.

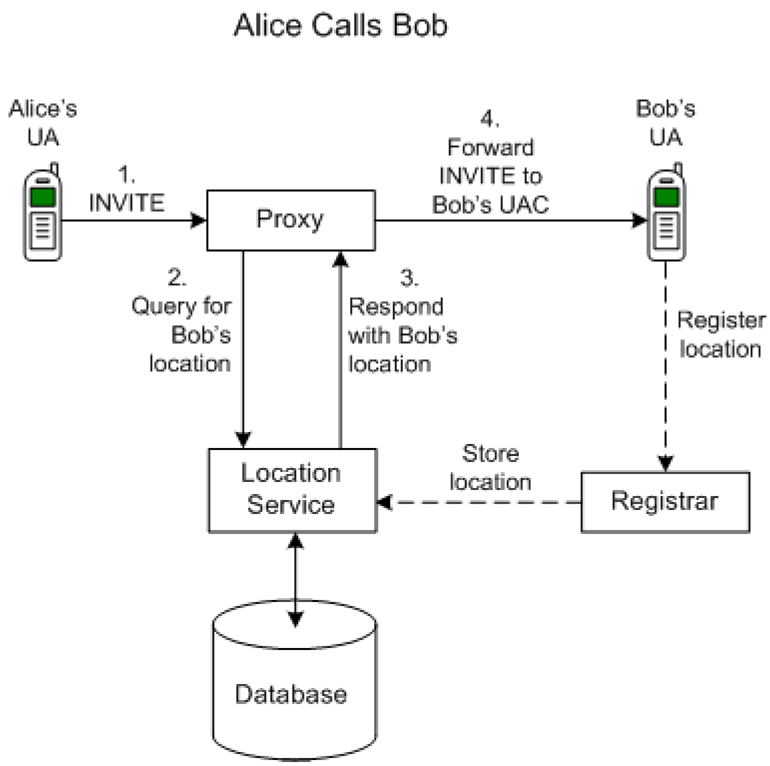
Figure 3–2 Registration flow



Upon receiving requests to the AOR, the proxy function locates the mapped URIs through a Location Service lookup and then proxies the request using the location information retrieved by this lookup.

Figure 3–3 illustrates a simplified view of the interaction between UAs when a subscriber, Alice, calls another subscriber, Bob, who is located in the same domain.

Figure 3–3 Interaction of UA Elements During a Call

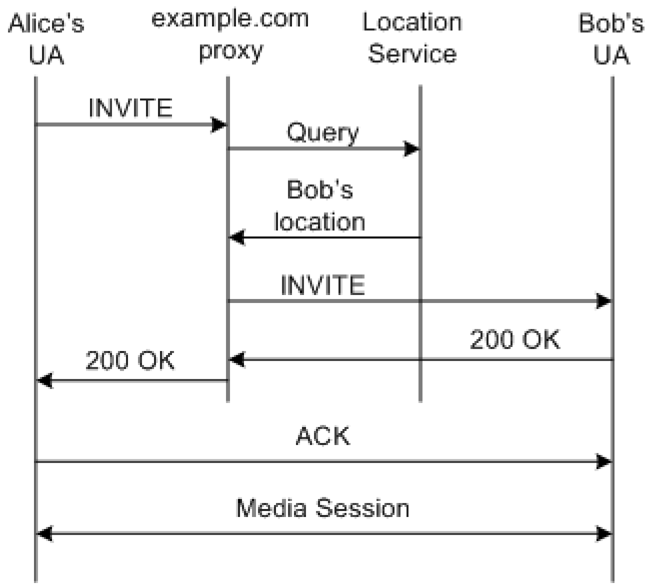


Bob may be registered from multiple user agents, such as personal phone, work phone, and computer. In this case, the query for Bob's location will return multiple bindings to the Proxy. The Proxy will then fork the call, either in parallel or sequentially, to the user agents that Bob is logged in to.

The Proxy is capable of proxying not only INVITE request, but other non-REGISTER requests such as MESSAGE, PUBLISH, SUBSCRIBE and so on.

When a caller and callee are in the same domain, the callee's location can be obtained by the outgoing proxy through the location service. A simplified example of the call flow for this scenario is shown in Figure 3–4. Note that this example does not include `100 Trying` and `180 Ringing` responses.

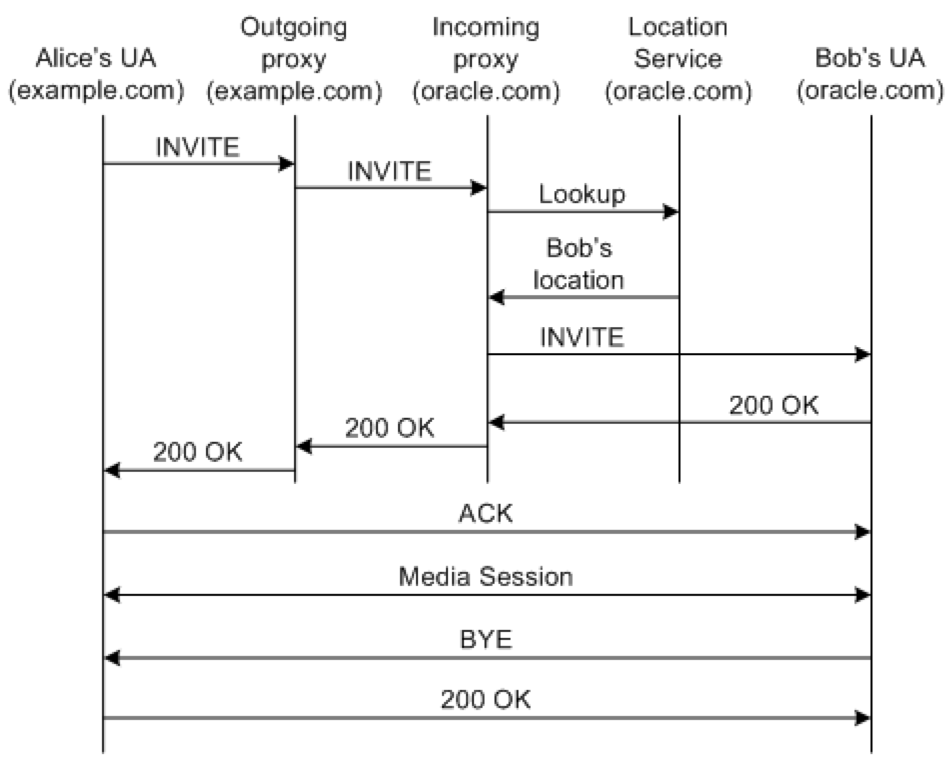
Figure 3–4 Call Establishment Flow Between Subscribers in a Single Domain



After the call is established, Alice and Bob's UAs can communicate directly, without using the Proxy. However, you can configure to route all subsequent traffic through the Proxy as well. This is the default and is useful if you want the ability to add additional services during the session.

If the caller and callee are in different domains, the outgoing proxy forwards the INVITE request to the callee's domain. The incoming proxy in the callee's domain performs the lookup and returns the callee's location, as illustrated in Figure 3–5.

Figure 3–5 Example of a Call Flow Between Two Domains



The Proxy can use ENUM lookup to resolve TEL URLs. The backend for the ENUM service is a DNS, which stores a mapping between TEL URLs and SIP URIs.

You configure the Proxy Registrar primarily through the Administration Console. You configure authentication for the Proxy Registrar by editing the `sip.xml` deployment descriptor packaged in the Proxy Registrar application. You can also edit advanced parameters by using WebLogic Scripting Tool For more information, see "Configuring the Proxy Registrar" in the chapter "Configuring Engine Tier Container Properties" in *Converged Application Server Administration Guide*.

Management Interfaces

Converged Application Server supports four primary management interfaces:

- JMX

  Converged Application Server interoperates with standard network element management systems via the Java Management eXtensions standard. Many common network management suites support JMX natively, which is the standard management technology for Java applications.

- SNMP

  Converged Application Server interoperates with standard network element management systems via use of the Simple Network Management Protocol, V2. The Converged Application Server SNMP MIB complies with MIB II. Converged Application Server also enables developers to send SNMP traps from within application code, as described in Generating SNMP Traps from Application Code in Developing SIP Applications.

  Converged Application Server also uses the SNMP features available in Oracle WebLogic Server 11g, such as SNMP proxying. See the WebLogic SNMP Management Guide in the Oracle WebLogic Server documentation for more information.

- Administration Console (GUI)

  Converged Application Server provides an extensive Web-based GUI that supports all configuration management, including deployment of applications, configuration of connectivity, and other common tasks. This interface offers secure, role-based administration of servers from any terminal that has access to the Administration Server and supports a standard HTML Web browser.

- Command-line Interface

  Converged Application Server provides a Command Line Interface (CLI) for manual runtime configuration from any network terminal with secure access to the Administration Server.

Adminstration Console

The Converged Application Server Web Administration Console is used for the following tasks:

- Configuring attributes of resources
- Deploying applications or components
- Monitoring resource usage
- Displaying log messages

- Starting and stopping servers

Figure 3–6 Converged Application Server Administration Console

Cluster-Wide Traffic Monitoring via the Administration Console

The Converged Application Server Administration console provides a convenient interface for observing current usage metrics as shown in Figure 3–7, "Cluster-Wide Session Metrics" and Figure 3–8, "SIP Data Tier Statistics".

Figure 3–7 Cluster-Wide Session Metrics

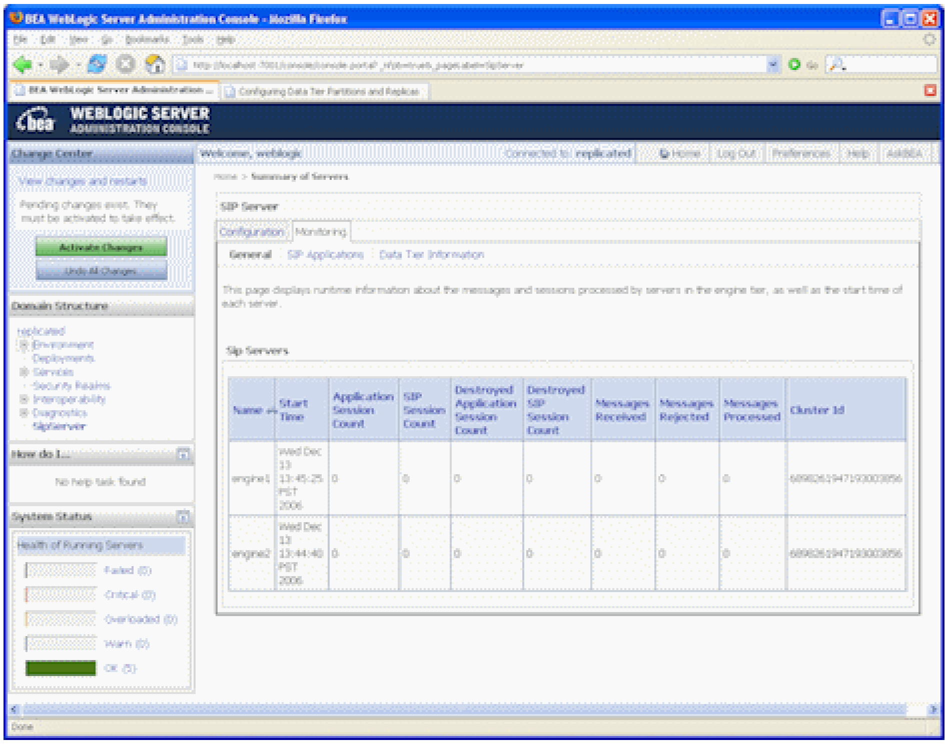
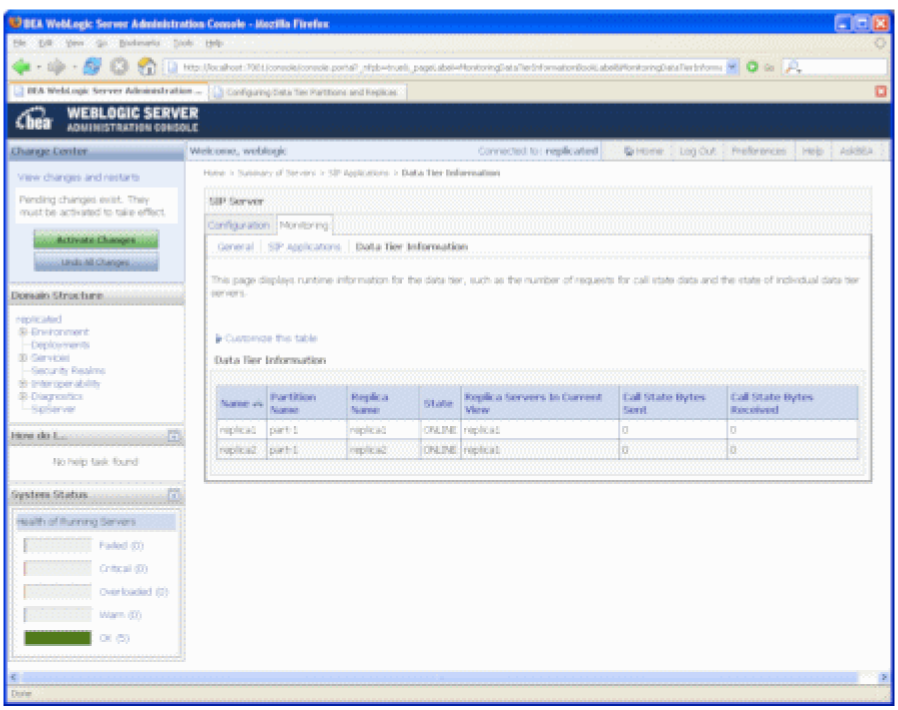


Figure 3–8 SIP Data Tier Statistics



Media Server Control

Converged Application Server enables control of media servers using the Media Server Control API based on JSR309, a standard Java interface. JSR309 (also referred to as JSR 309 and the JSR 309 API) defines an abstract Java interface for the manipulation of audio and video streams and conferences. Vendors of IP media servers provide JSR 309 based driver implementations that work with their IP media servers.

The JSR309 architecture assumes a distributed or IMS-like model where the Converged Application Server and media server reside on separate physical servers. User Agents (such as soft phones) interact with the applications deployed on Converged Application Server using the SIP protocol. Converged Application Server implements the SIP Servlets specification (JSR289) which allows application developers to develop converged SIP-J2EE applications. With the availability of JSR 309 based drivers in OCCAS, application developers can develop media-rich applications such as Conferencing, Ring-back tone, or IVR applications easily using the JSR 309 API.

For developers, the Media Server Control API provides a standard API for developing and deploying media rich, JSR-based applications for the Java platform without having prior knowledge of the underlying Media Server Control protocols. Moreover, a Java application that uses the Media Server Control API can use any JSR309-based implementation with any compatible media server.

JSR309 makes the use and adoption of media servers significantly easier for Enterprise Java developers; delivering the control they need within a familiar environment to reduce development time.

Converged Application Server supports media server control with JSR309 by implementing the Service Provider Interface (SPI) that allows a JSR309-based driver to be installed and registered with the server. For more information, see "Configuring the Media Server Control Factory" in the chapter "Configuring Engine Tier Container Properties" in *Converged Application Server Administration Guide*.

Charging and Billing

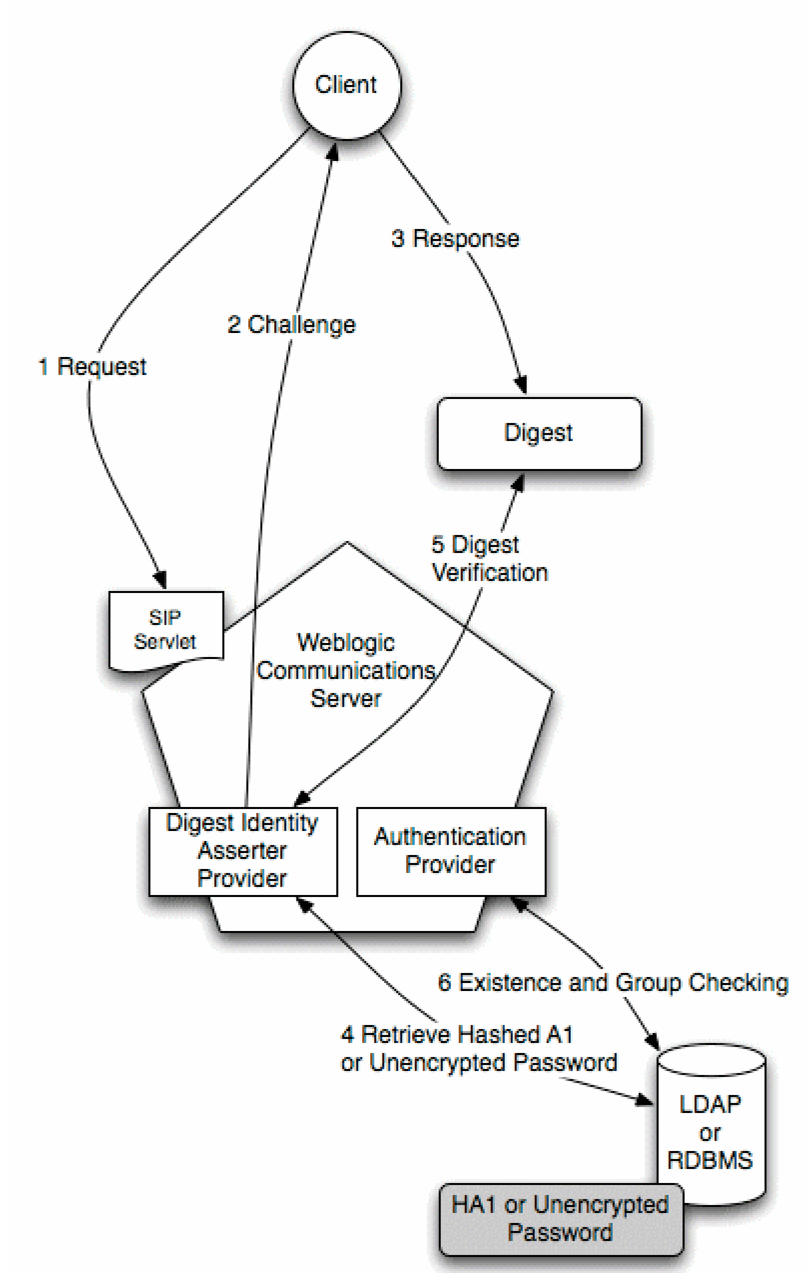
Converged Application Server provides both a Diameter Rf interface application and a Diameter Ro interface application to facilitate offline and online charging in IMS networks. See "Using the Diameter Rf Interface Application for Offline Charging" and "Using the Diameter Ro Interface Application for Online Charging" in *Converged Application Server Diameter Application Development Guide* for information about how to access and use these Diameter applications in your own SIP Servlets.

Security

Converged Application Server users must be authenticated when they request access to a protected resource, such as a protected method in a deployed SIP Servlet. Converged Application Server enables you to perform SIP Servlet authentication using any of the following techniques:

- **DIGEST authentication** uses a simple challenge-response mechanism to verify the identity of a user over SIP or HTTP. See "Configuring Digest Authentication" in *Converged Application Server Administration Guide*.

Figure 3–9 Digest Authentication Handling in Converged Application Server



- **CLIENT-CERT authentication** uses an X509 certificate chain passed to the SIP application to authenticate a user. The X509 certificate chain can be provided in a number of different ways. In the most common case, two-way SSL handshake is performed before transmitting the chain to ensure secure communication between the client and server.
- **BASIC authentication** uses the Authorization SIP header to transmit the username and password to SIP Servlets. BASIC authentication is not recommended for production systems unless you can ensure that all connections between clients and the Converged Application Server instance are secure.

Different SIP Servlets deployed on Converged Application Server can use different authentication mechanisms as necessary. The required authentication mechanism is specified in the auth-method element of the SIP Servlet Application's deployment descriptor. The deployment descriptor may also define resources that are to be protected, listing the specific role names that are required for access.

Authentication Providers

The Converged Application Server authentication services are implemented using one or more authentication providers. An authentication provider performs the work of proving the identity of a user or system process, and then transmitting the identity information to other components of the system.

Converged Application Server may be configured to use multiple authentication providers via different authentication methods. For example, when using Digest authentication an administrator may configure both a Digest Identity Asserter provider to assert the validity of a digest, and a second LDAP or RDBMS authentication provider that determines the group membership of a validated user.

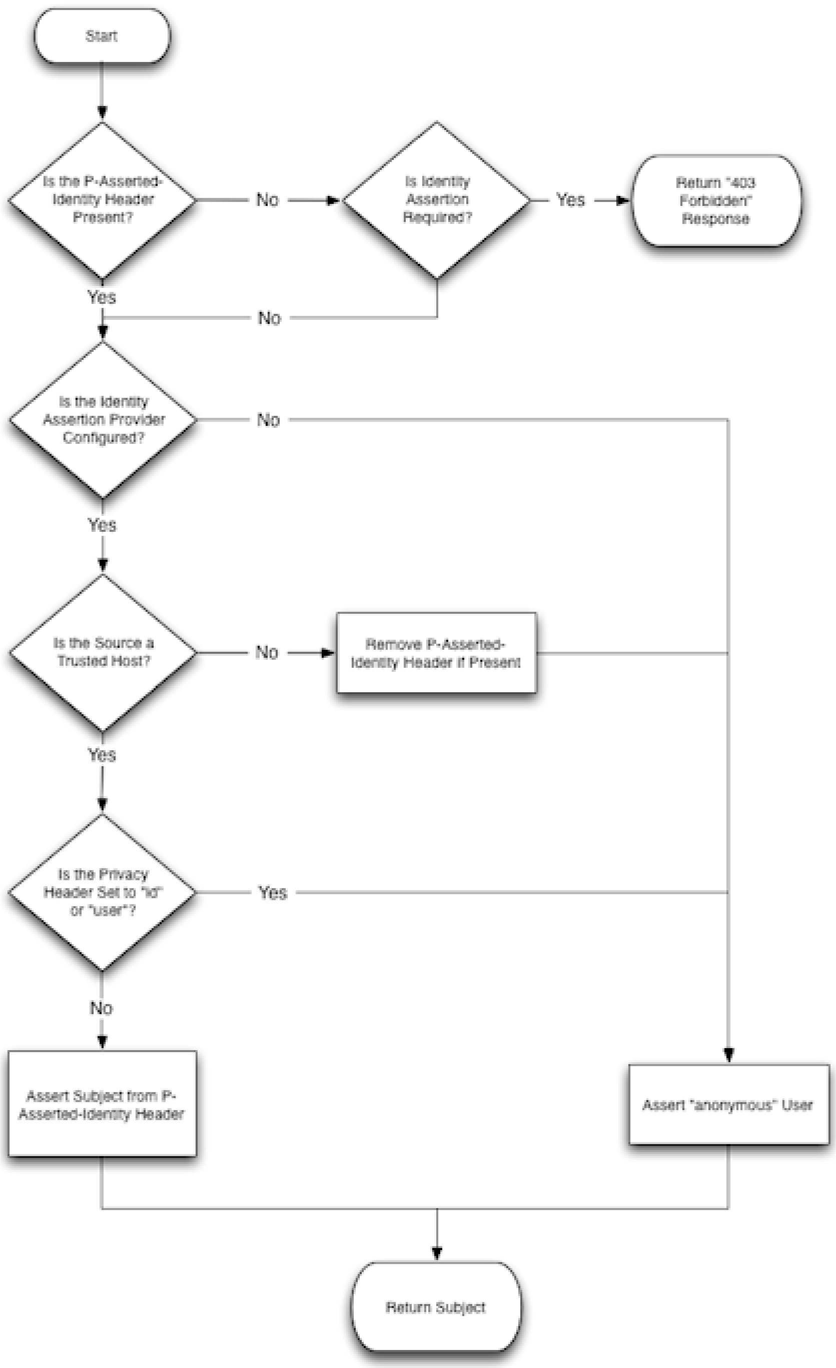
Trusted Host Authentication

Converged Application Server is designed for deployment scenarios where it is adjacent to trusted hosts and it is not required to fulfill the role of an application layer security boundary between the trusted and untrusted domains.

Converged Application Server enables administrators to designate network hosts that are considered to be "trusted." Trusted hosts are hosts for which Converged Application Server performs no authentication. If the server receives a SIP message having a destination address that matches a configured trusted hostname, the message is delivered without Authentication.

Converged Application Server supports the P-Asserted-Identity SIP header as described in IETF RFC 3325: Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks. This functionality automatically logs in using credentials specified in the P-Asserted-Identity header when they are received from a trusted host. When combined with the privacy header, P-Asserted-Identity also determines whether the message can be forwarded to trusted and non-trusted hosts.

Figure 3–10 Asserted Identity Handling in Converged Application Server



It is also possible to use Converged Application Server in scenarios that do not involve trusted hosts. See Chapter 5, "Standards Alignment," for a more detailed description of Converged Application Server standards compliance.

Declarative Security

The SIP Servlet API specification defines a set of deployment descriptor elements that can be used for providing declarative and programmatic security for SIP Servlets. The primary method for declaring security constraints is to define one or more security-constraint elements and role definitions in the sip.xml deployment descriptor. Converged Application Server adds additional deployment descriptor elements to help developers easily map SIP Servlet roles to actual principals and/or roles configured by the Converged Application Server administrator.

Protecting the Converged Application Server Domain with a Session Border Controller

A Session Border Controller (SBC) is a device used in VoIP networks to exert control over the signaling (and usually also the media streams) involved in setting up, conducting, and tearing down interactive media communications. SBCs are typically used to secure and protect the network and other devices in the operator's network from denial of service (DOS) attacks. Besides security, SBCs also perform functions such as QoS guarantees, regulatory compliances (lawful intercept), statistics collection, and so on. Services developed and deployed on Converged Application Server are most commonly hosted inside trusted networks. It is recommended to protect the network which hosts such services deployed on Converged Application Server with a Session Border Controller.

4

Converged Application Server Cluster Architecture

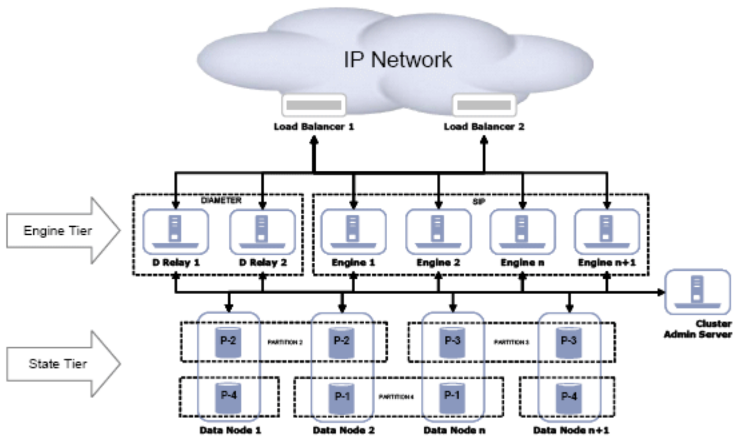
This chapter describes the Oracle Communications Converged Application Server cluster architecture.

Overview of the Cluster Architecture

Converged Application Server provides a multi-tier cluster architecture in which a stateless “Engine Tier” processes all traffic and distributes all transaction and session state to a “SIP Data Tier.” The SIP data tier is comprised of one or more partitions, labeled as “Data Nodes” in Figure 4–1 below. Each partition may contain one or more replicas of all state assigned to it and may be distributed across multiple physical servers or server blades. A standard load balancing appliance is used to distribute traffic across the Engines in the cluster. It is not necessary that the load balancer be SIP-aware; there is no requirement that the load balancer support affinity between Engines and SIP dialogs or transactions. However, SIP-aware load balancers can provide higher performance by maintaining a client’s affinity to a particular engine tier server.

Figure 4–1 shows an example Converged Application Server cluster.

Figure 4–1 Example Converged Application Server Cluster



In some cases, it is advantageous to have SIP data tier instances and engine tier instances running on the same physical host. This is particularly true when the physical servers or server blades in the cluster are based on Symmetrical Multi-Processing (SMP) architectures, as is now common for platforms such as Advance Telecom Computing Architecture (ATCA). This is not arbitrarily required, however, and it is entirely possible to physically distribute SIP data tier and Engine instances each to a different physical server or server blade.

There is no arbitrary limit to the number of engines, partitions or physical servers within a cluster, and there is no fixed ratio of engines to partitions. When dimensioning the cluster, however, a number of factors should be considered, such as the typical amount of memory required to store the state for a given session and the increasing overhead of having more than two replicas within a partition.

Converged Application Server Cluster Linear Scalability

Converged Application Server has demonstrated linear scalability from 2 to 16 hosts (up to 32 CPUs) in both laboratory and field tests and in commercial deployments. This characteristic is likely to be evident in larger clusters as well, up to the ability of the cluster interconnect (or alternatively the load balancer) to support the total traffic volume. Gigabit Ethernet is recommended as a minimum for the cluster interconnect.

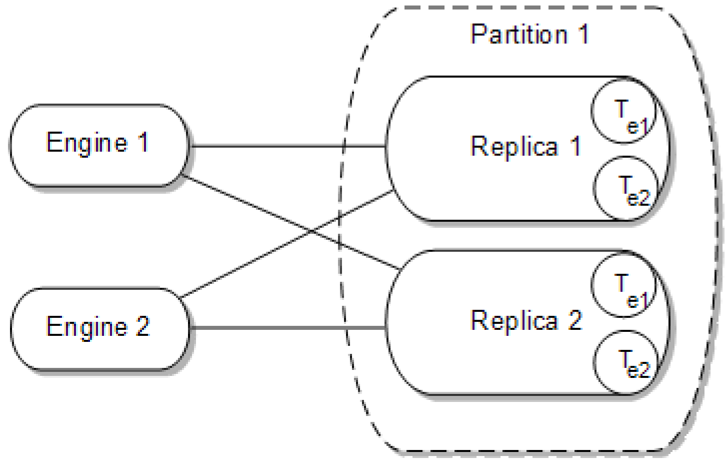
Converged Application Server State Replication

The Converged Application Server SIP data tier is an in-memory, peer-replicated store. The store also functions as a lock manager, whereby call state access follows a simple "library book" model (a call state can only be checked out by one SIP engine at a time).

The nodes in the SIP data tier are called *replicas*. To increase the capacity of the SIP data tier, the data is split evenly across a set of partitions. Each partition has a set of 1-8 replicas which maintain a consistent state. (Oracle recommends using no more than 3 replicas per partition.) The number of replicas in the partition is the *replication factor*.

Replicas can join and leave the partition. Any given replica serves in exactly one partition at a time. The total available call state storage capacity of the cluster is determined by the capacity of each partition.

Figure 4–2 Converged Application Server State Replication



The call state store is peer-replicated. This means that clients perform all operations (reads and writes) to all replicas in a partition. Peer replication stands in contrast to the more common primary-secondary replication architecture, wherein one node acts as a primary and the all other nodes act as secondaries. With primary-secondary replication, clients only talk directly to the current primary node. Peer-replication is roughly equivalent to the synchronous primary-secondary architecture with respect to failover characteristics, peer replication has lower latency during normal operations on average. Lower latency is achieved because the system does not have to wait for the synchronous 2nd hop incurred with primary-secondary replication.

Peer replication also provides better failover characteristics than asynchronous primary-secondary systems because there is no change propagation delay.

The operations supported by all replicas for normal operations are: "lock and get call state," "put and unlock call state," and "lock and get call states with expired timers." The typical message processing flow is simple:

1. Lock and get the call state.
2. Process the message.
3. Put and unlock the call state.

Additional management functions deal with bootstrapping, registration, and failure cases.

Partition Views

The current set of replicas in a partition is referred to as the partition view. The view contains an increasing ID number. A view change signals that either a new replica has joined the partition, or that a replica has left the partition. View changes are submitted to engines when they perform and operation against the SIP data tier.

When faced with a view change, engine nodes performing a lock/get operation must immediately retry their operations with the new view. Each SIP engine schedules a 10ms interval for retrying the lock/get operation against the new view. In the case of a view change on a put request, the new view is inspected for added replicas (in the case that the view change derives from a replica join operation instead of replica failure or shutdown). If there is an added replica, that replica also gets the put request to ensure consistency.

Timer Processing

An additional function of the SIP data tier is timer processing. The replicas set timers for the call states when call states perform put operations. Engines then poll for and “check out” timers for processing. Should an engine fail at this point, this failure is detected by the replica and the set of checked-out timers is forcefully checked back in and rescheduled so that another engine may check them out and process them.

As an optimization, if a given call state contains only timers required for cleaning up the call state, the SIP data tier itself expires the timers. In this special case, the call state is not returned to an engine tier for further processing, because the operation can be completed wholly within the SIP data tier.

Replica Failure

The SIP engine node clients perform failure detection for replicas, or for failed network connections to replicas.

During the course of message processing, an engine communicates with each replica in the current partition view. Normally all operations succeed, but occasionally a failure (a dropped socket or an invocation timeout) is detected. When a failure is detected the engine sends a “replica died” message to any of the remaining live replicas in the partition. (If there is no remaining live replica, the partition is declared “dead” and the engines cannot process calls hashing to that partition until the partition is restored). The replica that receives the failed replica notification proposes a new partition view that excludes the reportedly dead replica. All clients will then receive notification of the view change (see "Partition Views" for more information).

To handle partitioned network scenarios where one client cannot talk to the supposedly failed replica but another replica can, the “good” replica removes the reportedly failed replica offline, ensuring safe operation in the face of network partition.

Engine Failure

The major concerns with engine failure are:

- They are in the middle of a lock/get or put/unlock operation during failure
- They fail to unlock call states for messages they are currently processing
- They abandon the set of timers that they are currently processing

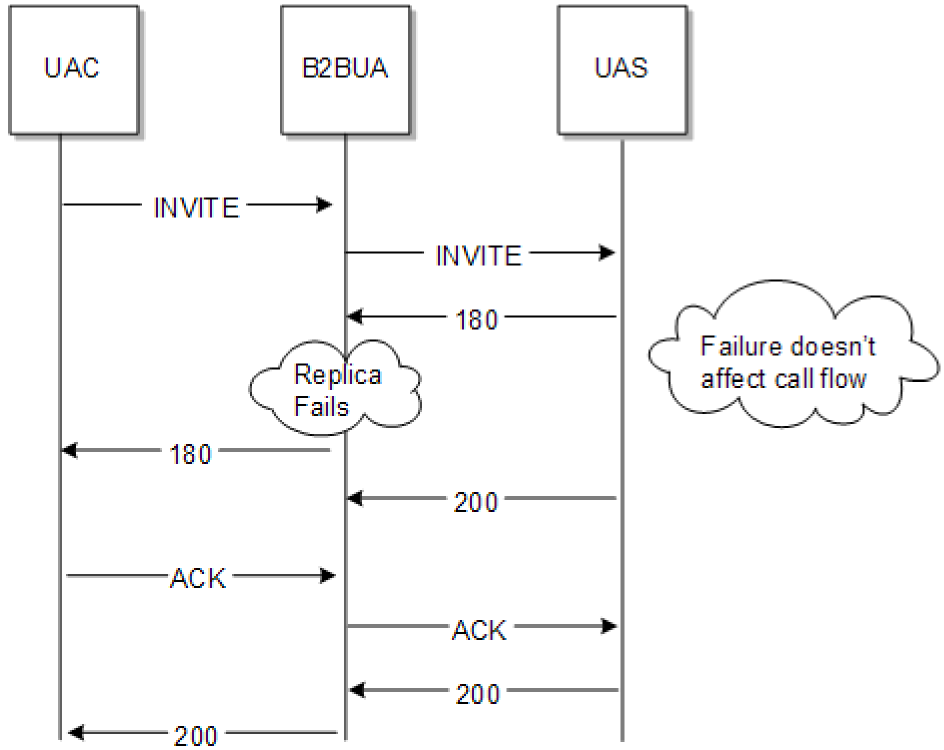
Replicas are responsible for detecting engine failure. In the case of failures during lock/get and put/unlock operations, there is risk of lock state and data inconsistency between the replicas (data inconsistency in the case of put/unlock only). To handle this, the replicas break locks for call states if they are requested by another engine and the current lock owner is deemed dead. This allows progress with that call state.

Additionally, to deal with possible data inconsistency in scenarios where locks had to be broken, the call state is marked as “possibly stale”. When an engine evaluates the response of a lock/get operation, it wants to choose the best data. If any one replica reports that it has non-stale data, that data is used. Otherwise, the “possibly stale” data is used (it is only actually stale in the case that the single replica that had the non-stale version died in the intervening period).

Effects of Failures on Call Flows

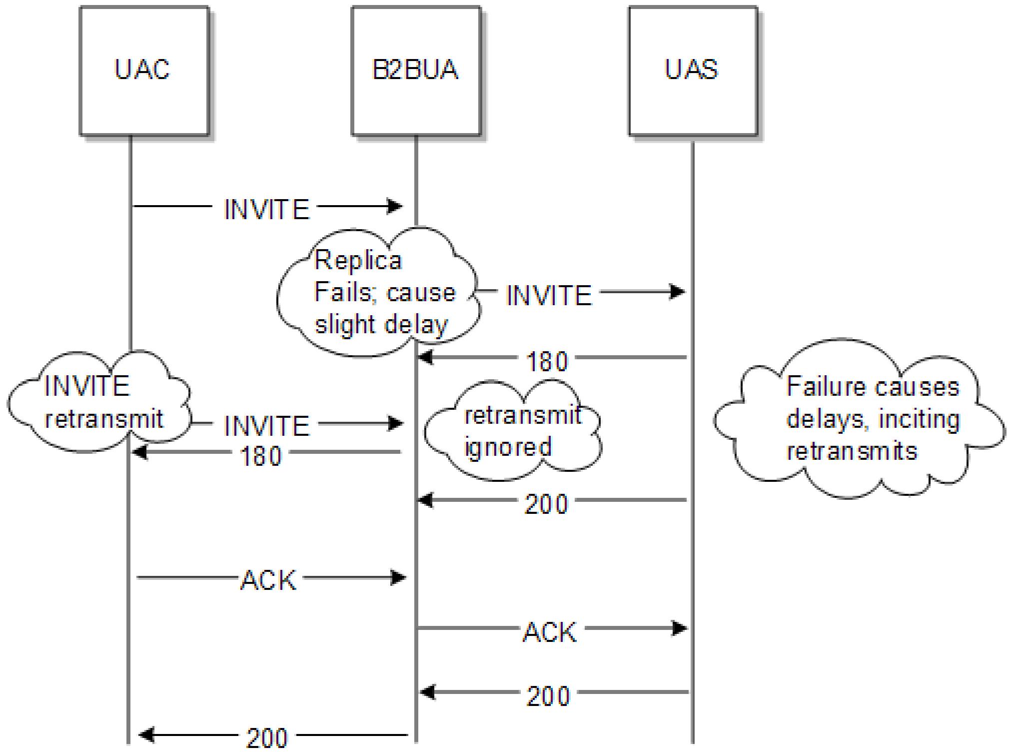
Because of the automatic failure recovery of the replicated store design, failures don’t affect call flow unless the failure is of a certain duration or magnitude.

Figure 4–3 Replication Example Call Flow 1



In some cases, failure recovery causes “blips” in the system where the engine’s coping with view changes causes message processing to temporarily back-up. This is usually not dangerous, but may cause UAC or UAS re-transmits if the backlog created is substantial.

Figure 4–4 Replication Example Call Flow 2



Catastrophic failure of a partition (whereby no replica is remaining) causes a fraction of the cluster to be unable to process messages. If there are four partitions, and one is lost, 25% of messages will be rejected. This situation will resolve once any of the replicas are put back in the service of that partition.

Diameter Protocol Handling

A Converged Application Server domain may optionally deploy support for the Diameter base protocol and IMS Sh interface provider on engine tier servers, which then act as Diameter Sh client nodes. SIP Servlets deployed on the engines can use the profile service API to initiate requests for user profile data, or to subscribe to and receive notification of profile data changes. The Sh interface is also used to communicate between multiple IMS Application Servers.

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

The relay agents included in Converged Application Server perform only stateless proxying of Diameter messages; messages are not cached or otherwise processed before delivery to the HSS.

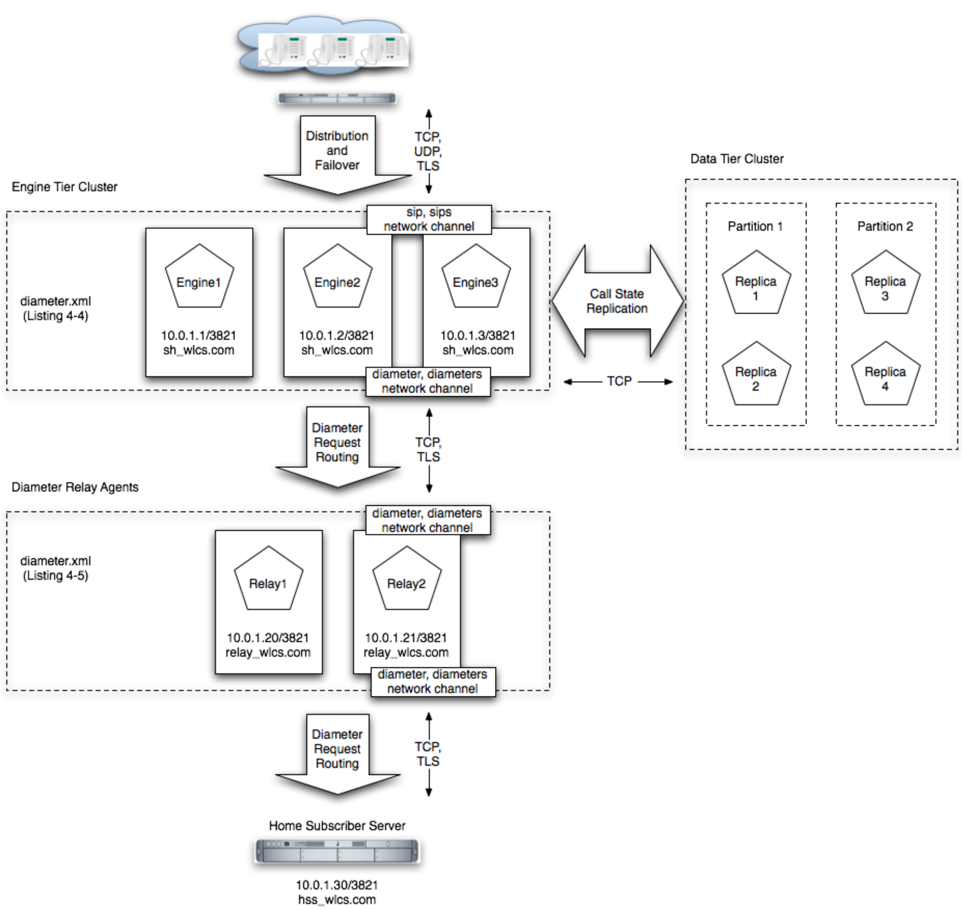
Note: In order to support multiple HSSs, the 3GPP defines the Dh interface to look up the correct HSS. Converged Application Server does not provide a Dh interface application, and can be configured only with a single HSS.

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

Converged Application Server also provides a simple HSS simulator that you can use for testing Sh client applications. You can configure a Converged Application Server instance to function as an HSS simulator by deploying the appropriate application.

Figure 4–5 shows diameter protocol handling within a Converged Application Server Diameter Domain.

Figure 4–5 Converged Application Server Diameter Domain



Deployment of Converged Application Server in Non-clustered configurations

Converged Application Server may be deployed in non-clustered configurations where session retention is not a relevant capability. The SIP signaling throughput of individual Converged Application Server instances will be substantially higher due to the elimination of the computing overhead of the clustering mechanism. Non-clustered configurations are appropriate for development environments or for cases where all deployed services are stateless and/or session retention is not considered important to the user experience (where users are not disturbed by failure of established sessions).

"Zero Downtime" Application Upgrades

With Converged Application Server, you can upgrade a deployed SIP application to a newer version without losing existing calls being processed by the application. This type of application upgrade is accomplished by deploying the newer application version alongside the older version. Converged Application Server automatically manages the SIP Servlet mapping so that new requests are directed to the new version. Subsequent messages for older, established dialogs are directed to the older application version until the calls complete. After all of the older dialogs have completed and the earlier version of the application is no longer processing calls, you can safely un-deploy it.

Converged Application Server's upgrade feature ensures that no calls are dropped while during the upgrade of a production application. The upgrade process also enables you to revert or rollback the process of upgrading an application. If, for example, you determine that there is a problem with the newer version of the deployed application, you can simply un-deploy the newer version. Converged Application Server then automatically directs all new requests to the older application version.

Requirements and Restrictions for Upgrading Deployed Applications

To use the application upgrade functionality of Converged Application Server:

- You must assign version information to your updated application in order to distinguish it from the older application version. Note that only the newer version of a deployed application requires version information; if the currently-deployed application contains no version designation, Converged Application Server automatically treats this application as the “older” version.
- Both the deployed application and the updated application must provide only SIP protocol functionality. You cannot upgrade converged HTTP/SIP applications using these procedures.
- A maximum of two different versions of the same application can be deployed at one time.
- If your application hard-codes the use of an application name (for example, in composed applications where multiple SIP Servlets process a given call), you must replace the application name with calls to a helper method that obtains the base application name. Converged Application Server provides SipApplicationRuntimeMBean methods for obtaining the base application name and version identifier, as well as determining whether the current application version is active or retiring.
- When applications take part in a composed application (using application composition techniques), Converged Application Server always uses the latest version of an application when only the base name is supplied.

Converged Application Server also provides the ability for Administrators to upgrade the SIP Servlet container, JVM, or application on a cluster-wide basis without affecting existing SIP traffic. This is accomplished by creating multiple clusters and having

Converged Application Server automatically forward requests during the upgrade process. See the discussion on upgrading the Oracle Communications Converged Application server software in *Converged Application Server Administration Guide* for more information on upgrading the SIP Servlet container, JVM, or applications.

5 Standards Alignment

Th chapter describes how Oracle Communications Converged Application Server complies with various specifications and RFCs.

- Overview of Converged Application Server Standards Alignment
- Java Sun Recommendation (JSR) Standards Compliance
- IETF RFC Compliance
- 3GPP R6 Specification Conformance

Overview of Converged Application Server Standards Alignment

Converged Application Server is developed with special attention to Internet Engineering Task Force(IETF) and 3rd Generation Partnership Project (3GPP) specifications. Feature development is prioritized according to general market trends, both observed and predicted. In cases where certain specifications are obsolete or where Internet drafts are formalized as 'Request For Comments' standards, Converged Application Server places priority on compliance with those specifications. In cases where specifications are part of a larger release plan, as with the 3GPP, Oracle prioritizes compliance with the latest ratified release (in this case, Release 6). This should not be presumed to mean that the product is not compliant with subsequent versions of component specifications, although this document does not summarize compliance with those specifications.

Java Sun Recommendation (JSR) Standards Compliance

Converged Application Server is compliant with Java EE version 5.0 and the corresponding Java EE component specifications.

Converged Application Server is further enhanced by the addition of a SIP Servlet container defined by JSR 289: "SIP Servlet API."

Converged Application Server has executed all related Test Compatibility Kits (TCKs) and has met the formal requirements established by Sun Microsystems for formal public statements of compliance.

IETF RFC Compliance

The following table lists the Converged Application Server level of compliance to common IETF Requests for Comment (RFCs) and Internet drafts. The level of compliance is defined as follows:

- **Yes**—Indicates that Converged Application Server directly supports the feature or specification.
- **Yes (Platform)**—Indicates Converged Application Server can host applications or components that implement the RFC. However, the RFC or feature has no impact on the transaction layer of the protocol or on the behavior of the SIP Servlet container.

Table 5–1 Converged Application Server IETF Compliance

RFC or Specification Number	Title	Compliant?	Additional Information
761	DoD Standard Transmission Control Protocol	Yes	See http://www.ietf.org/rfc/rfc761.txt
768	User Datagram Protocol	Yes	See http://www.ietf.org/rfc/rfc768.txt
1157	A Simple Network Management Protocol (SNMP)	Yes	Converged Application Server supports SNMP V2c traps. See http://www.ietf.org/rfc/rfc1157.txt
1847	Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted	Yes (Platform)	Converged Application Server supports applications that consume or generate signed or encrypted multipart MIME objects. See http://www.ietf.org/rfc/rfc1847.txt
1901	Introduction to Community-based SNMPv2	Yes	Converged Application Server supports SNMP V2c traps. See http://www.ietf.org/rfc/rfc1901.txt
1905	Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)	Yes	Converged Application Server supports SNMP V2c traps. See http://www.ietf.org/rfc/rfc1905.txt
1906	Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)	Yes	Converged Application Server supports SNMP over both TCP and UDP. See http://www.ietf.org/rfc/rfc1906.txt
1907	Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)	Yes (Platform)	See http://www.ietf.org/rfc/rfc1907.txt
2183	Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc2183.txt

Table 5–1 (Cont.) Converged Application Server IETF Compliance

RFC or Specification Number	Title	Compliant?	Additional Information
2246	The TLS Protocol Version 1.0	Yes	Converged Application Server supports TLS. See http://www.ietf.org/rfc/rfc2246.txt
2460	Internet Protocol, Version 6 (IPv6) Specification	Yes	See http://www.ietf.org/rfc/rfc2460.txt
2543	SIP: Session Initiation Protocol (v1)	Yes	Converged Application Server supports backward compatibility as described in this specification. See http://www.ietf.org/rfc/rfc2543.txt
2571	An Architecture for Describing SNMP Management Frameworks	Yes	See http://www.ietf.org/rfc/rfc2571.txt
2572	Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)	Yes	See http://www.ietf.org/rfc/rfc2572.txt
2573	SNMP Applications	Yes	See http://www.ietf.org/rfc/rfc2573.txt
2574	User-based Security Model (USM) for version 3 of theSimple Network Management Protocol (SNMPv3)	Yes	See http://www.ietf.org/rfc/rfc2574.txt
2575	View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)	Yes	See http://www.ietf.org/rfc/rfc2575.txt
2576	Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework	Yes	See http://www.ietf.org/rfc/rfc2576.txt
2616	Hypertext Transfer Protocol -- HTTP 1.1	Yes	See http://www.ietf.org/rfc/rfc2616.txt
2617	HTTP Authentication: Basic and Digest Access Authentication	Yes	See http://www.ietf.org/rfc/rfc2617.txt

Table 5–1 (Cont.) Converged Application Server IETF Compliance

RFC or Specification Number	Title	Compliant?	Additional Information
2782	A DNS RR for specifying the location of services (DNS SRV)	Yes	See http://www.ietf.org/rfc/rfc2782.txt
2786	Diffie-Helman USM Key Management Information Base and Textual Convention	Yes	See http://www.ietf.org/rfc/rfc2786.txt
2806	URLs for Telephone Calls	Yes	See http://www.ietf.org/rfc/rfc2806.txt
2848	The PINT Service Protocol: Extensions to SIP and SDP for IP Access to Telephone Call Services	Yes (Platform)	Note that implementing PINT services implies a pre-IMS architecture. Although Oracle favors the 3GPP/TISPAN architecture and approach to class 4/5 Service Emulation and does not advocate PINT, it is possible to implement PINT service elements using Converged Application Server. See http://www.ietf.org/rfc/rfc2848.txt
2960	Stream Control Transmission Protocol	Yes	SCTP supported only for Diameter traffic. See http://www.ietf.org/rfc/rfc2960.txt
2976	The SIP INFO Method	Yes	See http://www.ietf.org/rfc/rfc2976.txt
3204	MIME media types for ISUP and QSIG Objects	Yes (Platform)	Converged Application Server does not directly consume or generate ISUP and QSIG objects, but it supports applications that consume or generate these objects. See http://www.ietf.org/rfc/rfc3204.txt
3261	SIP: Session Initiation Protocol	Yes	See http://www.ietf.org/rfc/rfc3261.txt
3262	Reliability of Provisional Responses in the Session Initiation Protocol (SIP)	Yes	See http://www.ietf.org/rfc/rfc3262.txt
3263	Session Initiation Protocol (SIP): Locating SIP Servers	Yes	See http://www.ietf.org/rfc/rfc3263.txt
3264	An Offer/Answer Model with Session Description Protocol (SDP)	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3264.txt
3265	Session Initiation Protocol (SIP)-Specific Event Notification	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3265.txt

Table 5–1 (Cont.) Converged Application Server IETF Compliance

RFC or Specification Number	Title	Compliant?	Additional Information
3268	Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)	Yes (Platform)	Converged Application Server supports cryptographic services, but specific algorithms that are used are subject to local availability and export control. See `http://www.ietf.org/rfc/rfc3268.txt`
3311	The Session Initiation Protocol (SIP) UPDATE Method	Yes	Converged Application Server supports applications that conform to this specification. See `http://www.ietf.org/rfc/rfc3311.txt`
3312	Integration of Resource Management and Session Initiation Protocol (SIP).	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See `http://www.ietf.org/rfc/rfc3312.txt`
3313	Private Session Initiation Protocol (SIP) Extensions for Media Authorization	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See `http://www.ietf.org/rfc/rfc3313.txt`
3323	A Privacy Mechanism for the Session Initiation Protocol (SIP)	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See `http://www.ietf.org/rfc/rfc3323.txt`
3325	Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks	Yes	See `http://www.ietf.org/rfc/rfc3325.txt`
3326	The Reason Header Field for the Session Initiation Protocol (SIP)	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See `http://www.ietf.org/rfc/rfc3326.txt`
3327	Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts.	Yes (Platform)	See `http://www.ietf.org/rfc/rfc3327.txt`
3351	User Requirements for the Session Initiation Protocol (SIP) in Support of Deaf, Hard of Hearing and Speech-impaired Individuals	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See `http://www.ietf.org/rfc/rfc3351.txt`
3372	Session Initiation Protocol for Telephones (SIP-T): Context and Architectures	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See `http://www.ietf.org/rfc/rfc3372.txt`

Table 5–1 (Cont.) Converged Application Server IETF Compliance

RFC or Specification Number	Title	Compliant?	Additional Information
3420	Internet Media Type message/sipfrag		See http://www.ietf.org/rfc/rfc3420.txt
3428	Session Initiation Protocol (SIP) Extension for Instant Messaging	Yes	See http://www.ietf.org/rfc/rfc3428.txt
3455	Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP)	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3455.txt
3515	The Session Initiation Protocol (SIP) Refer Method.	Yes	See http://www.ietf.org/rfc/rfc3515.txt
3524	Mapping of Media Streams to Resource Reservation Flows	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3524.txt
3556	Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3556.txt
3578	Mapping of Integrated Services Digital Network (ISDN) User Part (ISUP) Overlap Signalling to the Session Initiation Protocol (SIP)	Yes (Platform)	Converged Application Server supports applications that conform to this specification, but it does not provide an ISUP interface. See http://www.ietf.org/rfc/rfc3578.txt
3581	An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing	Yes	See http://www.ietf.org/rfc/rfc3581.txt
3589	Diameter Command Codes for Third Generation Partnership Project (3GPP) Release 5	Yes	See http://www.ietf.org/rfc/rfc3589.txt
3588	Diameter Base Protocol	Yes	See http://www.ietf.org/rfc/rfc3588.txt

Table 5–1 (Cont.) Converged Application Server IETF Compliance

RFC or Specification Number	Title	Compliant?	Additional Information
3605	Real Time Control Protocol (RTCP) attribute in Session Description Protocol ((SDP)	Yes (Platform)	See http://www.ietf.org/rfc/rfc3605.txt
3608	Session Initiation Protocol (SIP) Extension Header Field for Service Route Discovery During Registration.	Yes (Platform)	Converged Application Server supports applications that conform to this specification, but it does not provide a means of storing the ServiceRoute established during registration. This functionality can be implemented as part of the application. See http://www.ietf.org/rfc/rfc3608.txt
3665	Session Initiation Protocol (SIP) Basic Call Flow Examples.	Yes	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3665.txt
3666	Session Initiation Protocol (SIP) Public Switched Telephone Network (PSTN) Call Flows	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3666.txt
3680	A Session Initiation Protocol (SIP) Event Package for Registrations	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3680.txt
3689	General Requirements for Emergency Telecommunication Service (ETS)	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3689.txt
3690	IP Telephony Requirements for Emergency Telecommunication Service (ETS)	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3690.txt
3702	Authentication, Authorization, and Accounting Requirements for the Session Initiation Protocol (SIP)	Yes	Converged Application Server version supports JDBC and LDAP. See http://www.ietf.org/rfc/rfc3702.txt
3725	Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)	Yes	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3725.txt

Table 5–1 (Cont.) Converged Application Server IETF Compliance

RFC or Specification Number	Title	Compliant?	Additional Information
3761	The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)	Yes	See `http://www.ietf.org/rfc/rfc3761.txt`
3764	Enumservice Registration for Session Initiation Protocol (SIP) Addresses-of-Record	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See `http://www.ietf.org/rfc/rfc3764.txt`
3824	Using E.164 numbers with the Session Initiation Protocol (SIP)	Yes	See `http://www.ietf.org/rfc/rfc3824.txt`
3826	The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model	Yes	See `http://www.ietf.org/rfc/rfc3826.txt`
3840	Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See `http://www.ietf.org/rfc/rfc3840.txt`
3841	Caller Preferences for the Session Initiation Protocol (SIP)	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See `http://www.ietf.org/rfc/rfc3841.txt`
3853	S/MIME Advanced Encryption Standard (AES) Requirement for the Session Initiation Protocol (SIP)	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See `http://www.ietf.org/rfc/rfc3853.txt`
3891	The Session Initiation Protocol (SIP) 'Replaces' Header	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See `http://www.ietf.org/rfc/rfc3891.txt`
3892	The Session Initiation Protocol (SIP) Referred-By Mechanism	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See `http://www.ietf.org/rfc/rfc3892.txt`

Table 5–1 (Cont.) Converged Application Server IETF Compliance

RFC or Specification Number	Title	Compliant?	Additional Information
3893	Session Initiation Protocol (SIP) Authenticated Identity Body (AIB) Format	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3893.txt
3903	Session Initiation Protocol (SIP) Extension for Event State Publication	Yes	See http://www.ietf.org/rfc/rfc3903.txt
3911	The Session Initiation Protocol (SIP) "Join" Header	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3911.txt
3959	The Early Session Disposition Type for the Session Initiation Protocol (SIP)		See http://www.ietf.org/rfc/rfc3959.txt
3960	Early Media and Ringing Tone Generation in the Session Initiation Protocol (SIP)	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3960.txt
3966	The tel URI for Telephone Numbers	Yes	See http://www.ietf.org/rfc/rfc3966.txt
4028	Session Timers in the Session Initiation Protocol (SIP)	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc4028.txt
4032	Update to the Session Initiation Protocol (SIP) Preconditions Framework	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc4032.txt
4244	An Extension to the Session Initiation Protocol (SIP) for Request History Information	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc4244.txt
4320	Actions Addressing Identified Issues with the Session Initiation Protocol's (SIP) Non-INVITE Transaction	Yes	See http://www.ietf.org/rfc/rfc4320.txt
4321	Problems Identified Associated with the Session Initiation Protocol's (SIP) Non_INVITE Transaction	Yes	See http://www.ietf.org/rfc/rfc4321.txt

Table 5–1 (Cont.) Converged Application Server IETF Compliance

RFC or Specification Number	Title	Compliant?	Additional Information
4474	Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)	Yes	See http://www.ietf.org/rfc/rfc4474.txt.
4483	A Mechanism for Content Indirection in Session Initiation Protocol (SIP) Messages	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc4483.txt.
4566	SDP: Session Description Protocol	Yes	Converged Application Server supports applications that consume or generate SDP. See http://www.ietf.org/rfc/rfc4566.txt
4825	The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc4825.txt
5888	The Session Description Protocol (SDP) Grouping Framework	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc5888.txt
5806	Diversion Indication in SIP	Yes	Converged Application Server supports applications that conform to this specification. See https://datatracker.ietf.org/doc/rfc5806/
draft-donovan-mmusic-183-00	SIP 183 Session Progress Message Draft	Yes (Platform)	Converged Application Server supports applications that conform to this specification. See http://tools.ietf.org/html/draft-donovan-mmusic-183-00
draft-ietf-sip-gruu-15	Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)	Yes	See http://tools.ietf.org/id/draft-ietf-sip-gruu-15.txt
draft-reeder-snmpv3-usm-3desede-00	Extension to the User-Based Security Model (USM) to Support Triple-DES EDE in "Outside" CBC Mode	Yes	See http://tools.ietf.org/id/draft-reeder-snmpv3-usm-3desede-00.txt

3GPP R6 Specification Conformance

Table 5–2 summarizes the ability of Converged Application Server to support implementation of the enablers or application functions identified by each applicable 3GPP Release 6 specification.

Other than the exceptions noted, Converged Application Server does not impose any restrictions on implementing applications or functions that are compliant with those associated with the Application Server entity described in the specification. In some cases, applications must implement support for SIP methods or headers. The default behavior of the Converged Application Server Sip Servlet Container is to pass unrecognized headers, request methods and payloads to SIP Servlets using normal SIP Servlet API procedures.

Table 5–2 3GPP R6 Specification Conformance

Specification	Comments
3GPP TS 23.228: "IP Multimedia Subsystem (IMS); Stage 2 (Release 6)"	No comments.
3GPP TS 24.229: "IP Multimedia Call Control Protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3 (Release 6)"	Converged Application Server does not enforce the requirement that only one p-charging-function-address header per SIP request as described in sub-section 5.7.1.2. Converged Application Server does enforce uniqueness. Converged Application Server does not provide privacy support as described in sub-section 5.7.3.
3GPP TS 23.141: "Presence Service; Architecture and Functional description (Release 6)"	Converged Application Server does not natively support the Ph (MAP), Pl (LIF-MLP), Px (DIAMETER Cx/Dx), Pg (phase 4, 3GPP Release 5),Pc (CAMEL phase 4, 3GPP Release 5) or Pr, Pk and Pp (RADIUS) reference points. Converged Application Server does not support IPv6 as required for the Presence User Agent (Peu) reference point as required in sub-section 4.3.1.
3GPP TS 23.218: "IP Multimedia (IM) session handling; IM call model; Stage 2 (Release 6)"	No comments.
3GPP TS 24.247 "Messaging using the IP Multimedia (IM) Core Network (CN) subsystem; Stage 3 (Release 6)"	Converged Application Server does not provide support for the Message Session Relay Protocol (MSRP), although it is presumed that an MSRP relay will typically be implemented as a Media Resource Function in the IMS architecture.
3GPP TS 24.841: "Presence service based on Session Initiation Protocol (SIP); Functional models, information flows and protocol details (Release 6)"	Converged Application Server does not provide support for IETF RFC 3310: "Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA)".
3GPP TS 24.109: "Bootstrapping interface (Ub) and Network application function interface (Ua); Protocol details (Release 6)"	Converged Application Server does not provide support for IETF RFC 3310: "Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA)". Converged Application Server supports the 'X-3GPP-Asserted-Identity extension-header' for use in applying access control and authorization constraints within the integrated security framework.

Table 5–2 (Cont.) 3GPP R6 Specification Conformance

Specification	Comments
3GPP TS 29.328: "IP Multimedia Subsystem (IMS) Sh interface; Signalling flows and message contents"	No comments.
3GPP TS 29.329: "Sh interface based on the Diameter protocol; Protocol details"	No comments.
3GPP TS 32.299: "Telecommunication management; Charging management; Diameter charging applications"	No comments.
3GPP TS 33.222: "Generic Authentication Architecture (GAA); Access to network application functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS) (Release 6)"	Converged Application Server supports the Application Server role in the GAA.

A

SIP Servlet API Service Invocation

This appendix describes the Service invocation method of the SIP Servlet API (JSR 116).

- SIP Servlet API Overview
- Servlet Mapping Rules: Objects, Properties and Conditions

SIP Servlet API Overview

The SIP Servlet API provides a model for application composition and interaction Service Interaction which is analogous with a simplistic implementation of the Service Capability Interaction Manager (SCIM) alluded to by the 3GPP. Handling of all incoming requests is governed by the Converged Application Server SIP Servlet Container in accordance with the SIP Servlet API specification.

The Oracle communications Converged Application Server SIP Servlet Container filters received Initial SIP requests and applies a set of defined rules (Servlet Mapping Rules) to determine which SIP Servlets within the deployed applications shall be invoked to service that particular request. This order is always sequential and is defined in a configuration file built up through successive deployments of SIP applications.

Within the deployment descriptor for each SIP Application that is deployed, a sequence of conditions, called Servlet Mapping Rules, is defined. These rules determine which Servlets will handle any initial request. As the request object is “routed” between Servlets, the path from Servlet to Servlet is recorded in a fashion equivalent to the “record-route” and “via” headers in SIP requests. This route is stored as part of the SIP application session and is appended to subsequent requests within the same dialogue in either “forward” or “reverse” order depending on the orientation of the “From” and “To” tags for the request. This internal “route” is stripped from the request object before a SIP request leaves Converged Application Server and is not visible to external SIP servers. It is again added whenever a new request within an existing dialog is received.

The SIP Servlets (SIP/HTTP application) that are invoked in this manner are unaware that any other SIP/HTTP application exists. This is one of the fundamental characteristics of the SIP Servlet programming model. Making maximal use of this model requires that the Servlet container be treated by the developer as if it is a logical sub-network, with the container effectively acting as an intermediary proxy. In many ways, the SIP Servlet Container may be compared with the Serving CSCF function in an IMS architecture.

Servlet Mapping Rules: Objects, Properties and Conditions

Servlet mapping rules are defined by the service developer and are detailed in the Deployment Descriptor for the application. The deployment descriptor is a document that is contained within the SAR archive file that is deployed on Converged Application Server. There may be more than one Servlet mapping rule defined within the Deployment Descriptor for the application (SIP/HTTP application). In this case, these rules must be applied in the order in which they are defined in the Deployment Descriptor.

The following figure provides an example of a simple Servlet mapping rule found in a typical Deployment Descriptor.

Note: Servlet mapping rules are entirely concerned with the content of the SIP message being processed. It is not possible to use information regarding the actual IP address and port number on which the request was received as service trigger points unless this information matches the request URI of the Sip message.

The Servlet mapping rule shown in Example A–1 illustrates the following Boolean expression:

(Method="INVITE" OR Method = "MESSAGE" OR Method="SUBSCRIBE") AND

(Method="INVITE" OR Method = "MESSAGE" OR (NOT Header = "from" Match = "Bob"))

Note: This is the same logical condition used in the Initial filter Criteria example provided in 3GPP TS 29.228 Annex C expressed as a Servlet Mapping Rule.

Example A–1 Example Servlet Mapping Rule

```
<servlet-mapping>
<servlet-name>servlet1</servlet-name>
<pattern>
    <and>
    <or>
        <equal>
        <var>request.method</var>
        <value>INVITE</value>
        </equal>
            <equal>
        <var>request.method</var>
        <value>MESSAGE</value>
            </equal>
        <equal>
        <var>request.method</var>
        <value>SUBSCRIBE</value>
            </equal>
    </or>
    <or>
        <equal>
        <var>request.method</var>
        <value>INVITE</value>
        </equal>
```

```
            <equal>
        <var>request.method</var>
        <value>MESSAGE</value>
            </equal>
        <not>
            <equal>
            <var>request.from.display-name</var>
            <value>Bob</value>
            </equal>
        </not>
    </or>
    </and>
</pattern>
</servlet-mapping>
```

Supported Service Trigger Points

Service Point Triggers are the attributes of a SIP request that may be evaluated by Servlet Mapping Rules. See "Section 11.1: Triggering Rules" in the JSR 116 specification for more information.

Request Object

The Request Object is a Java representation of a SIP request.

- **method**: the request method, a string
- **uri**: the request URI; for example a SipURI or a TelURL
- **from**: an Address representing the value of the From header
- **to**: an Address representing the value of the To header

URI

- **scheme**: the URI scheme

SipURI (extends URI)

- **scheme**: a literal string – either “sip” or “sips”
- **user**: the “user” part of the SIP/SIPS URI
- **host**: the “host” part of the SIP/SIPS URI. This may be a domain name or a dotted decimal IP address.
- **port**: the URI port number in decimal format; if absent the default value is used (5060 for UDP and TCP, 5061 for TLS).
- **tel**: if the “user” parameter is not “phone”, this variable is undefined. Otherwise, its value is the telephone number contained in the “user” part of the SIP/SIPS URI with visual separators stripped. This variable is always matched case insensitively (the telephone numbers may contain the symbols ‘A’, ‘B’, ‘C’ and ‘D’).
- **param.name**: value of the named parameter within a SIP/SIPS URI; name must be a valid SIP/SIPS URI parameter name.

TelURL (extends URI)

- **scheme**: always the literal string “tel”
- **tel**: the tel URL subscriber name with visual separators stripped off

- **param.name**: value of the named parameter within a tel URL; name must be a valid tel URL parameter name

Address

- **uri**: the URI object; see URI, SipURI, TelURL types above
- **display-name**: the display-name portion of the From or To header

Conditions and Logical Connectors

- **equal**: compares the value of a variable with a literal value and evaluates to true if the variable is defined and its value equals that of the literal. Otherwise, the result is false.
- **exists**: takes a variable name and evaluates to true if the variable is defined, and false otherwise.
- **contains**: evaluates to true if the value of the variable specified as the first argument contains the literal string specified as the second argument.
- **subdomain-of**: given a variable denoting a domain name (SIP/SIPS URI host) or telephone subscriber (tel property of SIP or Tel URLs), and a literal value, this operator returns true if the variable denotes a subdomain of the domain given by the literal value. Domain names are matched according to the DNS definition of what constitutes a subdomain; for example, the domain names “example.com” and “research.example.com“are both subdomains of “example.com”. IP addresses may be given as arguments to this operator; however, they only match exactly. In the case of the tel variables, the subdomain-of operator evaluates to true if the telephone number denoted by the first argument has a prefix that matches the literal value given in the second argument; for example, the telephone number “1 212 555 1212" would be considered a subdomain of “1212555”.
- **and**: contains a number of conditions and evaluates to true if and only if all contained conditions evaluate to true
- **or**: contains a number of conditions and evaluates to true if and only if at least one contained condition evaluates to true
- **not**: negates the value of the contained condition.

The equal and contains operators optionally ignore character case when making comparisons. The default is case-sensitive matching.

B

Acronyms

This appendix describes the acronyms used in the Converged Application Server documentation.

Acronyms

- 3GPP—3rd Generation Partnership Project
- API—Application Program Interface
- CSP—Communications Service Provider
- HSS—Home Subscriber Server
- HTTP—Hypertext Transport Protocol
- IDE—Integrated Development Environment
- IETF—Internet Engineering Task Force
- IMS—IP Multimedia Subsystem
- IP—Internet Protocol
- ISC—IMS Service Control
- ITU—International Telecommunication Union
- Java EE—Java Platform, Enterprise Edition
- OAM—Operation, Administration and Maintenance
- PoC—Push to Talk over Cellular
- RAM—Random Access Memory
- RDBMS—Relational Database Management System
- SCE—Service Creation Environment
- S-CSCF—Serving Call Session Control Function
- SDK—Software Development Kit
- SIP—Session Initiation Protocol
- SPA—Service Provider Administrator
- TCK—Technology Compatibility Kit
- TCP—Transport Control Protocol
- UDP—User Datagram Protocol

C
References

This appendix provides references to the industry specifications that are mentioned in the Oracle Communications Converged Application Server documentation.

References

1. 3GPP TS 22.250: "IP Multimedia Subsystem (IMS) group management; Stage 1 (Release 6)"
2. 3GPP TS 22.340: "IMS Messaging; Stage 1 (Release 6)"
3. 3GPP TS 23.002: "Network architecture (Release 6)"
4. 3GPP TS 23.141: "Presence Service; Architecture and Functional description (Release 6)"
5. 3GPP TS 23.218: "IP Multimedia (IM) session handling; IM call model; Stage 2 (Release 6)"
6. 3GPP TS 23.228: "IP Multimedia Subsystem (IMS); Stage 2 (Release 6)"
7. 3GPP TS 24.109: "Bootstrapping interface (Ub) and Network application function interface (Ua); Protocol details (Release 6)"
8. 3GPP TS 24.141: "Presence service using the IP Multimedia (IM); Core Network (CN) subsystem; Stage 3 (Release 6)"
9. 3GPP TS 24.147: "Conferencing using the IP Multimedia (IM) Core Network (CN) Subsystem; Stage 3 (Release 6)"
10. 3GPP TS 24.228: "Signaling flows for the IMS call control based on SIP and SDP; Stage 3 (Release 6)"
11. 3GPP TS 24.229: "IP Multimedia Call Control Protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3 (Release 6)"
12. 3GPP TS 24.147: "Conferencing using the IP Multimedia (IM) Core Network (CN) subsystem; Stage 3 (Release 6)"
13. 3GPP TS 24.247 "Messaging using the IP Multimedia (IM) Core Network (CN) subsystem; Stage 3 (Release 6)"
14. 3GPP TS 24.841: "Presence service based on Session Initiation Protocol (SIP); Functional models, information flows and protocol details (Release 6)"
15. 3GPP TS 26.141: "IP Multimedia System (IMS) Messaging and Presence; Media formats and Codecs (Release 6)"
16. 3GPP TS 24.109: "Bootstrapping interface (Ub) and network application function interface (Ua); Protocol details (Release 6)"

17. 3GPP TS 29.328: "IP Multimedia Subsystem (IMS) Sh interface; Signalling flows and message contents"

18. 3GPP TS 29.329: "Sh interface based on the Diameter protocol; Protocol details (Release 6)"

19. 3GPP TS 33.222: "Generic Authentication Architecture (GAA); Access to network application functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS) (Release 6)"

20. draft-burger-sipping-netann-11: "Basic Network Media Services with SIP"

21. draft-ietf-simple-presence-rules-02: "Presence Authorization Rules."

22. draft-ietf-simple-xcap-03 (July 2004): "The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)".

23. draft-ietf-simple-xcap-list-usage-02 (February 2004): "An Extensible Markup Language (XML) Format for Representing Resource Lists".

24. draft-ietf-simple-xcap-package-03: "An Extensible Markup Language (XML) Document Format for Indicating Changes in XML Configuration Access Protocol (XCAP) Resources."

25. draft-ietf-sip-connect-reuse-03.txt (April 2005): "Connection Reuse in the Session Initiation Protocol (SIP)"

26. draft-ietf-sip-content-indirect-mech-03 (June 2003): "A Mechanism for Content Indirection in Session Initiation Protocol (SIP) Messages".

27. draft-ietf-sipping-cc-conferencing-01 (June 2003): "Session Initiation Protocol Call Control - Conferencing for User Agents".

28. draft-ietf-sipping-conference-package-03 (February 2004): "A Session Initiation Protocol (SIP) Event Package for Conference State".

29. draft-ietf-sipping-conferencing-framework-01 (October 2003): "A Framework for Conferencing with the Session Initiation Protocol".

30. draft-ietf-xcon-cpcp-xcap-00 (April 2004): "The Conference Policy Control Protocol (CPCP)".

31. draft-isomaki-simple-xcap-pidf-manipulation-usage-00 (February 2004): "An Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Usage for Manipulating Presence Document Contents".

32. draft-jennings-sipping-outbound-00 (April 2005): "SIP Conventions for Connection Usage"

33. IETF RFC 2251: "Lightweight Directory Access Protocol (v3)"

34. IETF RFC 2327: "SDP: Session Description Protocol"

35. IETF RFC 2460: "Internet Protocol, Version 6 (IPv6) Specification"

36. IETF RFC 2486: "The Network Access Identifier"

37. IETF RFC 2617: "HTTP Authentication: Basic and Digest Access Authentication".

38. IETF RFC 3261: "SIP: Session Initiation Protocol"

39. IETF RFC 3263: "Session Initiation Protocol (SIP): Locating SIP Servers"

40. IETF RFC 3310: "Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA)"

41. IETF RFC 3325: “Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Network”

42. IETF RFC 3412: “Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)”

43. IETF RFC 3418: “Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)”

44. IETF RFC 3428: “Session Initiation Protocol (SIP) Extension for Instant Messaging”

45. IETF RFC 3512: “Configuring Networks and Devices with Simple Network Management Protocol (SNMP)”

46. IETF RFC 3761: “The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)”

47. IETF RFC 3840: “Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)”

48. IETF RFC 3966: “The tel URI for Telephone Numbers”

49. RFC 3261: “Caller Preferences for the Session Initiation Protocol (SIP)”

50. RFC 3329 (January 2003): “Security Mechanism Agreement for the Session Initiation Protocol (SIP)”.

51. LIF TS 101: “Mobile Location Protocol Specification” (Location Interoperability Forum 2001)

52. JSR 116: “SIP Servlet API”

53. JSR 58: “JavaTM 2 Platform, Enterprise Edition 1.3 Specification”