# CREACION Y EJECUCIÓN DE SERVICIOS DE VALOR AGREGADO BAJO EL CONCEPTO SDP, CASO EMCALI.


Anexos

Gerardo Rojas Sierra

Director
Magíster. Francisco Martínez Pabón

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Línea de Investigación en Servicios Avanzados de Telecomunicaciones
Popayán, Enero de 2012

# Contenido

# Lista de Figuras

# Lista de Tablas

# Anexo A

# SDP comerciales y Organismos de Estandarización

El presente anexo expone de forma general las Plataformas de Despliegue de Servicios (SDP, Service Delivery Platfom) más significativas en el mercado para la creación, ejecución y despliegue de Servicios de Valor Agregado (VAS, Value Added Services) en Redes de Próxima Generación (NGN, Next Generation Network) haciendo énfasis en la plataforma del proveedor de Equipos de Telecomunicaciones Zhong Xing (ZTE, Zhong Xing Telecommunication Equipment), de Empresas Municipales de Cali (EMCALI EICE-ESP), y la arquitectura de aprovisionamiento de servicios de Opencloud, con RHINO como corazón para el desarrollo de servicios. Además, se realiza una descripción de los organismos de estandarización que conforman el núcleo del concepto de las SDP.

## A.1 Estructuras base para la SDP

Es este punto se describe de manera general las principales organizaciones de estandarización detrás de la construcción de la arquitectura de las SDP para el aprovisionamiento de VAS en las NGN

**OSE**

El Grupo Móvil Abierto (OMA, Open Mobile Alliance) está formado por varias empresas, incluyendo importantes operadores móviles, distribuidores de dispositivos e infraestructura de red, compañías de Tecnologías de la Información (IT, Information Technology) y proveedores de contenido y servicios, que tiene como objetivo consolidar en una sola organización todas las actividades de estandarización para el área de habilitadores de servicios móviles, incluyendo la arquitectura e interfaces abiertas e independientemente de las redes y plataformas de bajo nivel [1].

La especificación de un grupo de habilitadores, la definición de componentes estándar y las relaciones existentes entre estos elementos, en conjunto conforman el Ambiente de Servicios de OMA (OSE, OMA Service Enviroment), el cual representa, una arquitectura estándar, flexible, extensible e interoperable, en donde los servicios pueden ser desarrollados, integrados y desplegados con una complejidad menor [2].

La arquitectura OSE (ver Figura 1) se encuentra compuesta por los siguientes elementos [2]:

Habilitador: tecnología para el desarrollo, despliegue u operación de un servicio. Especifica una o más interfaces públicas.
Implementación de habilitador: puede ser visto como una plantilla que representa una implementación de cualquier habilitador OMA. Ofrece funciones estandarizadas de un recurso determinado y expone las interfaces de gestión del ciclo de vida que permiten usar las capacidades para la administración de componentes de los habilitadores.

Interfaz: límite común entre dos sistemas asociados.

Enlace de interfaz de habilitador: provee la sintaxis y los protocolos utilizados para tener acceso a los habilitadores de servicio.

Recurso: elemento que representa una capacidad en el dominio de un proveedor de servicios.

Ejecutor de políticas: elemento encargado de suministrar mecanismos de gestión basado en políticas para la protección de recursos.

Aplicación: implementación de un conjunto de funciones que habilitan uno o más servicios o que realizan algún otro tipo de trabajo útil.

Ambiente de ejecución o gestor del ciclo de vida de un habilitador: módulo en el cual se soportan varias funciones que permiten al dominio OSE controlar los habilitadores de servicio. Entre estas funcionalidades encontramos el proceso de monitoreo, el manejo del ciclo de vida del software, algunos soportes del sistema (manejo de hilos, balance de carga, transparencia, etc.), Operación Y Mantenimiento (O&M, Operation & Maintenance), entre otras.

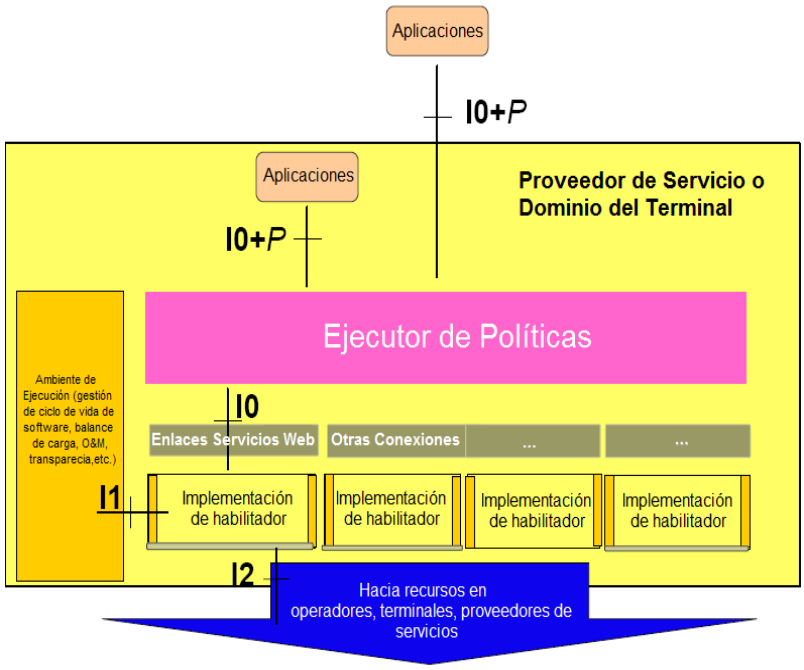



Figura 1 Elementos de la arquitectura OSE Fuente: OMA

**JAIN**

La iniciativa para Interfaces de Programación de Aplicaciones (API, Application Programming Interface) de Java para Redes Integradas (JAIN, Java API for Integrated Networks), representa a una comunidad de expertos en comunicaciones que define un conjunto de API Java para migrar las redes e interfaces de comunicaciones propietarias a estándares abiertos, permitiendo un rápido, simple y económico desarrollo, despliegue y mantenimiento de servicios y productos de próxima generación [3].

Dichas especificaciones definidas en la iniciativa JAIN son construidas basándose en tres objetivos principales heredados de la tecnología Java [3]:

Portabilidad de servicio: bajo la filosofía hecho una vez, corre en cualquier parte, define un conjunto de interfaces comunes estandarizadas Java que mapean aquellas soluciones que utilizan interfaces propietarias, asegurando una completa portabilidad.

Independencia de red: bajo la filosofía cualquier red, permite que las aplicaciones sean soportadas y accedidas por cualquier tecnología de red bien sea alambrada, inalámbricas o basada en el protocolo IP.

Desarrollo abierto: bajo la filosofía por cualquiera, define las API de comunicaciones de fácil uso permitiendo a diferentes desarrolladores crear nuevas aplicaciones que anteriormente eran de complejo desarrollo y despliegue.

Los anteriores objetivos actualmente se encuentran enmarcados en dos aéreas de desarrollo de especificaciones de API Java [3]:

Interfaces de contenedor que especifican las API del ambiente de ejecución bien sea orientado al dominio de las comunicaciones o al empresarial.

Interfaces de aplicación que especifican las API que permiten el desarrollo de servicios a través de redes de comunicaciones fijas, móviles y de datos basadas en el Protocolo Internet (IP, Internet Protocol).

En la Figura 2 se expone una topología Java típica y recomendada para las redes de comunicaciones, en donde se puede observar la forma como diferentes tecnologías e interfaces Java, tanto propietarias como desarrolladas por la misma comunidad, pueden complementar las API definidas en la iniciativa JAIN para ofrecer una solución mejorada y completa [3].

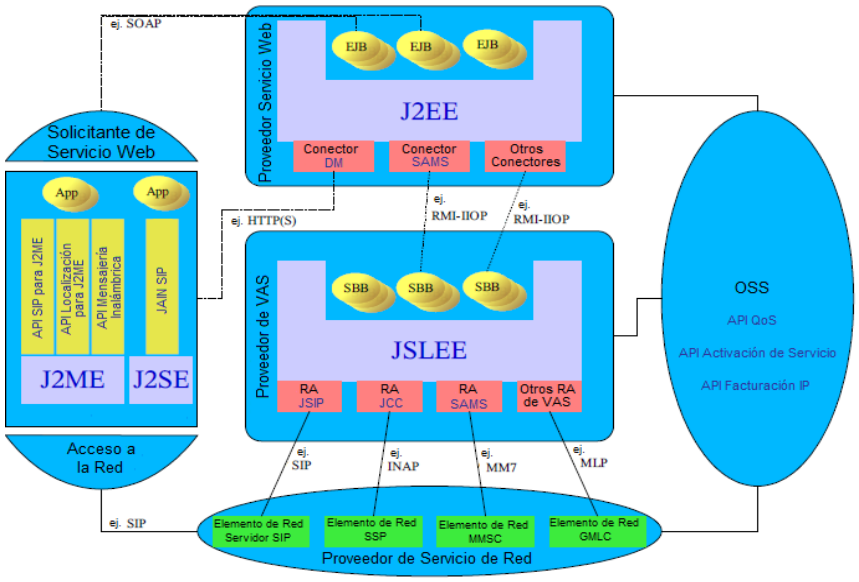



Figura 2 Java en la topología de red de comunicaciones Fuente: Sun Microsystems

**OSA/Parlay**

Las especificaciones de Acceso a Servicios Abierto (OSA, Open Service Access) definen una arquitectura que permite a los desarrolladores de aplicaciones implementar servicios que hagan uso de las funcionalidades de red a través de interfaces estandarizadas abiertas [4]. Estas funciones de red se definen en términos de un conjunto de Características de Capacidades de Servicio (SCF, Service Capability Features) que son accesibles desde dichas interfaces y son soportadas por diferentes Servidores de Capacidad de Servicio (SCS, Service Capability Servers) [5].

El objetivo de OSA es proveer una interfaz estándar, extensible y escalable, que permita la inclusión de nuevas funcionalidades y mejoras posteriores en la red, causando un mínimo impacto en las aplicaciones [5].

La arquitectura OSA (véase Figura 3) consta de tres partes, los cuales se describen a continuación:

Aplicaciones: componentes software que suministran servicios a los usuarios finales a través de la utilización de las SCF. Estas aplicaciones son implementadas en uno o más servidores de aplicación.

Framework: proporciona a las aplicaciones mecanismos básicos que les permiten hacer uso de las capacidades de servicio en una red.

Servidor(es) de capacidad de servicio: los SCS contienen las SCF que son ofrecidas a las aplicaciones.

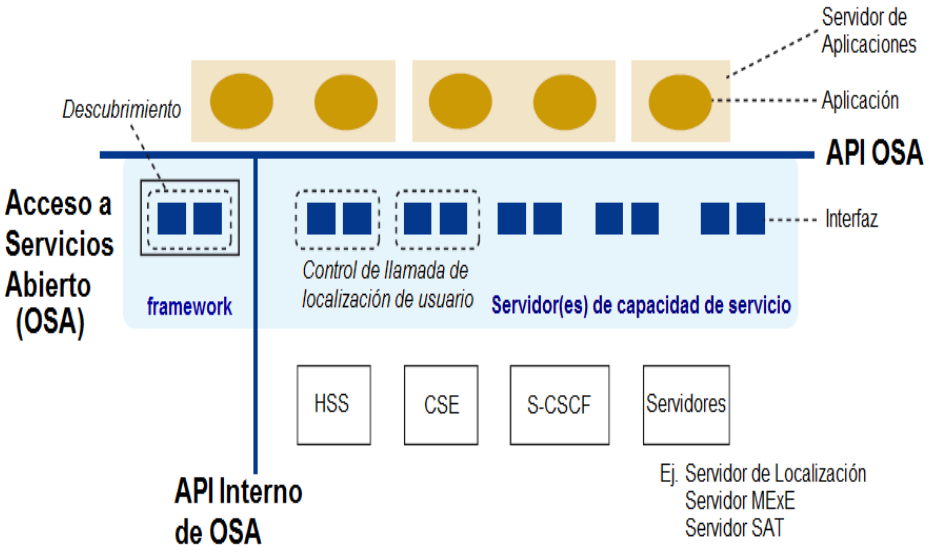


Figura 3 Visión general de OSA Fuente: ETSI

OSA propone un framework que contiene las diferentes interfaces que relacionan las partes que componen la arquitectura de la Figura 4 [6].

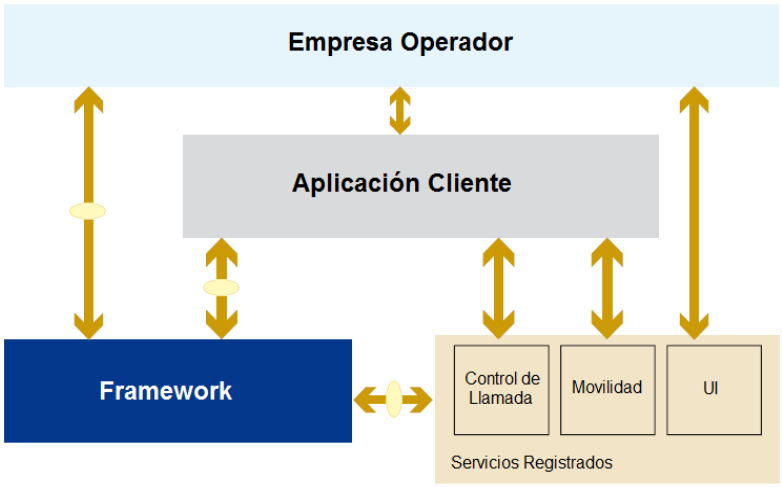


Figura 4 Relaciones del framework de OSA/Parlay. Fuente: ETSI

Interfaz framework-aplicación: proporciona a las aplicaciones los mecanismos necesarios que les permiten hacer uso de las capacidades de servicio en la red [4]. Dentro de estos mecanismos podemos encontrar autenticación, autorización, descubrimiento de capacidades, establecimiento de acuerdo de servicio y acceso a las capacidades de servicio [6].

Interfaz framewok-capacidad de servicio: proporciona el soporte multivendedor [4]. El mecanismo básico encontrado es el de registro de las capacidades de servicio de red, incluyendo las generadas recientemente debido a la instalación o actualización de un SCS [6].

Interfaz framework-operador: proporciona al operador los mecanismos básicos que le permiten administrar las cuentas de los clientes y gestionar tanto los perfiles como los contratos de servicio [4]. La función elemental es la de suscribir el servicio en donde se represente el acuerdo contractual entre el operador y el framework, de tal forma que el operador actúa como cliente/suscriptor del servicio y las aplicaciones cliente como usuarios o consumidores del servicio [6].

## A.2 SDP Comerciales

**Oracle**

La industria Oracle es una de las principales empresas que desarrollan sistemas de gestión de base de datos, software de mediador (fusion middleware), planificación de recursos empresariales (ERP, Enterprise Resource Planning) y gestión de relación cliente (CRM, Custumer Relationship Management), entre muchas otras tecnologías [7].

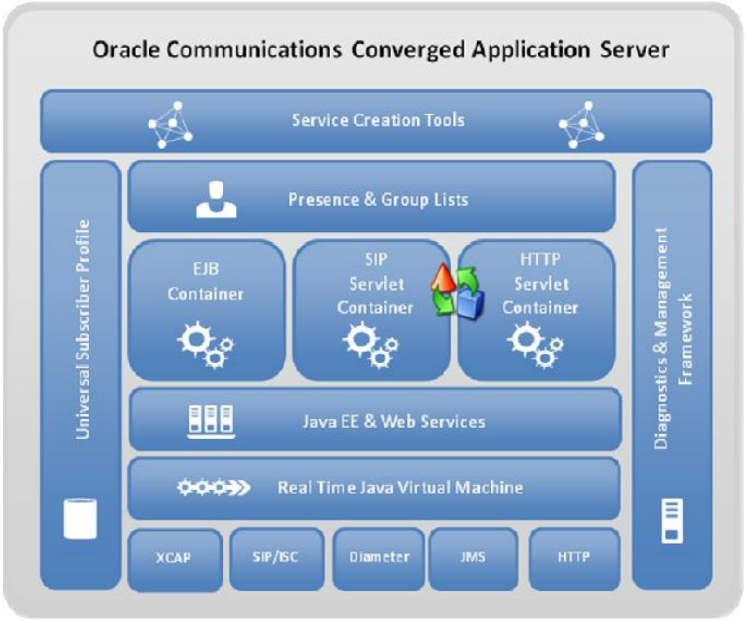



Figura 5 Servidor de aplicaciones de comunicaciones convergentes. Fuente: Oracle

El despliegue rápido de servicios de telecomunicaciones usando estándares abiertos es el concepto que ha venido tomando Oracle para la tecnología de las comunicaciones. La plataforma de despliegue de esta compañía, reúne la flexibilidad y madurez del mediador de fusión y de sus bases de datos, con una fuerte funcionalidad de telecomunicaciones. Esto permite a los operadores un rápido y eficiente despliegue de nuevos servicios de voz, datos y multimedia [7].

Esta SDP consiste de una familia de productos que permiten la creación, ejecución, exposición y gestión de servicios de telecomunicaciones. El servidor de aplicaciones convergentes y el

GateKeeper de servicios de comunicaciones son los dos productos clave de esta compañía para el desarrollo de VAS en NGN.
Servidor de aplicaciones de comunicaciones convergentes**:** este servidor de aplicación, basado en la tecnología de la Edición Empresarial de Java (JEE, Java Enterprise Edition) con soporte para el Protocolo de Inicio de Sesión (SIP, Session Initiation Protocol), para el Subsistema Multimedia IP (IMS, IP Multimedia Subsystem) y construido bajo el paradigma de Arquitectura Orientada a Servicios (SOA, Service Oriented Architecture), es una poderosa herramienta con altos índices de disponibilidad, fiabilidad y desempeño para suministrar una gran variedad de VAS en entornos de nueva generación. La
Figura 5 muestra la arquitectura del servidor Oracle para la creación, despliegue y ejecución de servicios convergentes de telecomunicaciones [8].

GateKeeper de comunicaciones convergentes: Este componente es una plataforma de exposición de servicios de telecomunicaciones convergentes basado en WEB-SOA, el cual permite a los operadores de red, potencializar sus servicios a través de la vinculación de nuevos actores que enriquezcan el contenido de las diferentes aplicaciones. Este módulo comprende todos los mecanismos de seguridad que requiere un operador al momento de exponer sus capacidades de red a terceros. Es aquí donde procesos como la Calidad de Servicio (QoS, Quality of Service) son ejecutados para cumplir con los altos requerimientos que los VAS demandan. En la
Figura 6 puede apreciarse la arquitectura funcional de este componente.

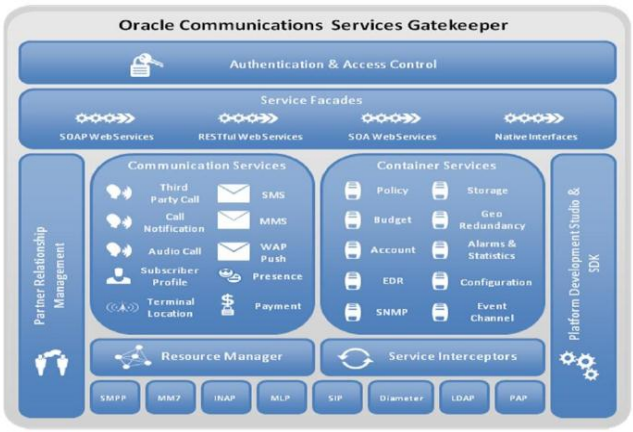



Figura 6 Vista funcional del GateKeeper de servicios de comunicaciones Fuente: Oracle

**IBM**

La industria IBM es sin duda la más grande y antigua compañía del mundo de la tecnología de la computación que junto con su grupo de soluciones en comunicaciones ha desarrollado un entorno de despliegue de servicios para que los operadores ofrezcan una gran variedad de servicios emergentes. Conocido como ambiente de despliegue de servicios del operador (SPDE, Service Provider Delivery Enviroment), el Framework de esta compañía suministra un completo mecanismo de seguridad, gestión, mantenimiento, creación y despliegue de servicios. Gracias a su orientación SOA, esta arquitectura flexible y escalable, permite una

rápida creación y bajos costos de implementación. Entre las características más relevantes encontramos:
Integración horizontal entre sus funciones y modelos de negocio Basado en estándares de IT (SOA, WEB 2.0, etc) y en la industria de la comunicaciones (eTOM, SID, NGOSS, IMS, SIP) Plataforma de servicios independiente de la red del operador Soporte de múltiples ambientes de creación de servicios Abstracción de red y exposición a terceros a través de las API y servicios web Bajo acoplamiento entre los componentes software Capacidad de adaptación y facilidad de crecimiento frente a la evolución tecnológica y la convergencia En la
Figura 7 puede apreciarse los diferentes módulos funcionales de la arquitectura SPDE de IBM

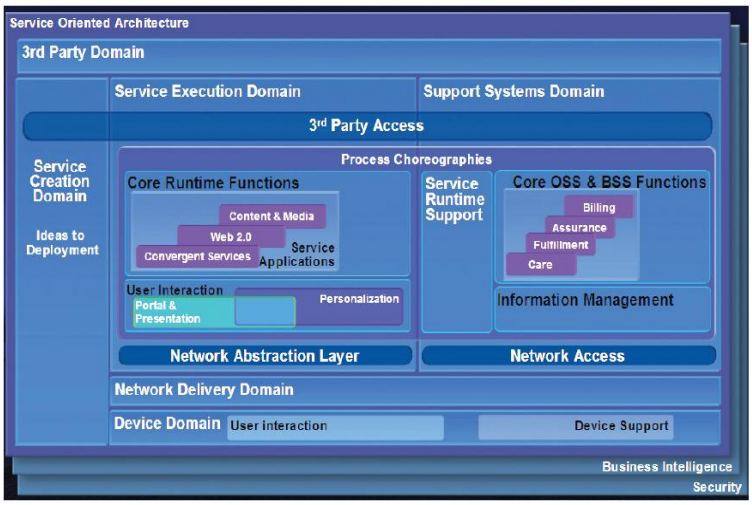



Figura 7 Ambiente de despliegue de servicios del Operador Fuente: IBM

**Microsoft**

La corporación Microsoft es una multinacional de la tecnología de la información, con un nivel impresionante de ventas anuales y más de 71 mil empleados en todo el mundo. Microsoft, conocido como uno de los principales impulsadores de la revolución digital, está adoptando un nuevo concepto en el desarrollo y entrega de servicios. Dentro de sus grandes variedades de productos de comunicaciones, se destaca el Framework diseñado para la capa de servicio donde es realizada la integración de las infraestructuras legadas de Sistemas de Soporte de Operaciones (OSS, Operations Support Systems)y Sistemas de Soporte de Negocio (BSS, Business Support System). Este entorno de servicios interconectados, denominado Framework de Servicios Conectados (CSF, Connected Services Framework) por la compañía, ofrece un ambiente para la creación, ejecución, despliegue y gestión de servicios. Bajo el paradigma SOA, la SDP de Microsoft, facilita a los operadores de telecomunicaciones la adaptación ante los cambios tecnológicos. La Figura 8 expone los diferentes módulos de la arquitectura CSF

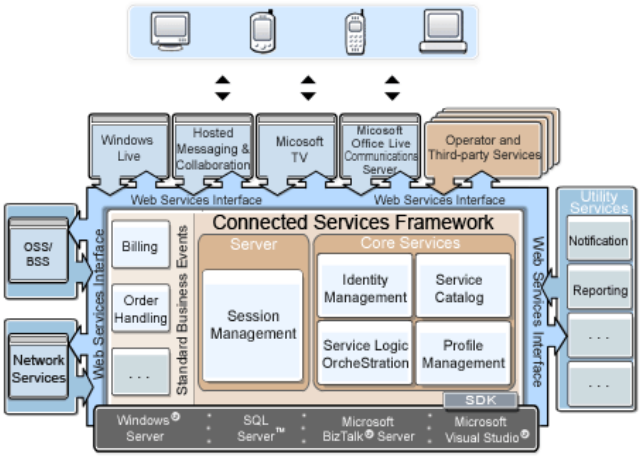



Figura 8 Arquitectura CSF. Fuente: Microsoft

**BEA**

Esta compañía líder a nivel mundial en el desarrollo de software para infraestructura de comunicaciones y entonarnos empresariales, trae al mercado su plataforma de comunicaciones WebLogic (WLCP, WebLogic Comunications Platforms), la cual habilita la convergencia del mundo telco con el IT. Esta plataforma, Figura 9, consiste de dos productos principales, un servidor SIP y un GateKeeper de red.

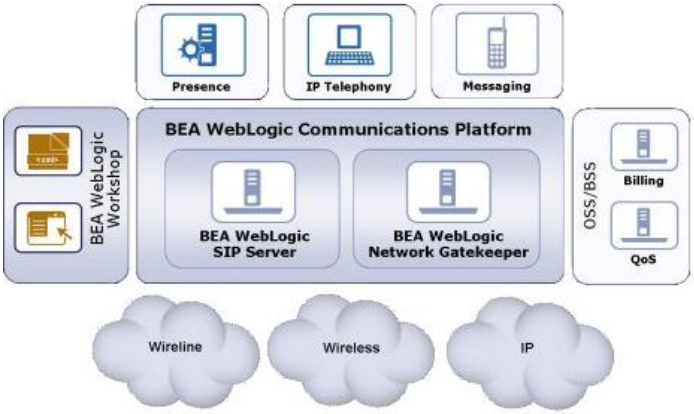



Figura 9 Plataforma de comunicaciones WebLogic Fuente: BEA

Servidor SIP WebLogic: Este componente es un servidor de aplicación SIP que junto con la tecnología JEE, permite el desarrollo de una gran variedad de servicios. La Figura 10 muestra la arquitectura de este producto.

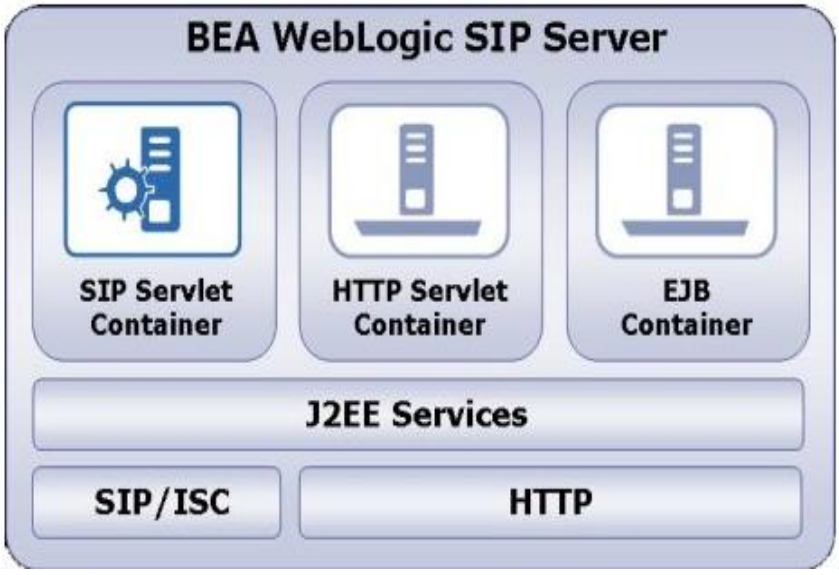



Figura 10 Arquitectura servidor SIP WebLogic. Fuente: BEA

GateKeeper de red: en este componente son aplicadas las políticas de seguridad para la manipulación de las capacidades de red del operador por parte de terceros. La selección de las capacidades se basa en el uso apropiado de las diferentes API que encapsulan las funcionalidades para el desarrollo de aplicaciones externas al dominio de confianza del operador, además, dentro de este componente, está la posibilidad de introducir nuevos componentes (SIP, Interfaces IMS, componentes redes inteligentes, etc) para ampliar el portafolio de VAS.

**Ericsson**

Esta compañía es una multinacional líder proveedora de equipos para telecomunicaciones y servicios para operadores de redes fijas y móviles. Dentro de sus tres aéreas de trabajo (redes, servicios y multimedia), la compañía ha establecido una SDP multimedia para el despliegue de servicios avanzados en redes móviles de telecomunicaciones. Dentro de las características más importantes se encuentran:

Soporte de funciones de entrega de servicios Orquestación y procesos de negocio Soporte de funciones comunes Ejecución de servicios Control de servicios

La Figura 11 presenta la arquitectura de la SDP de esta compañía

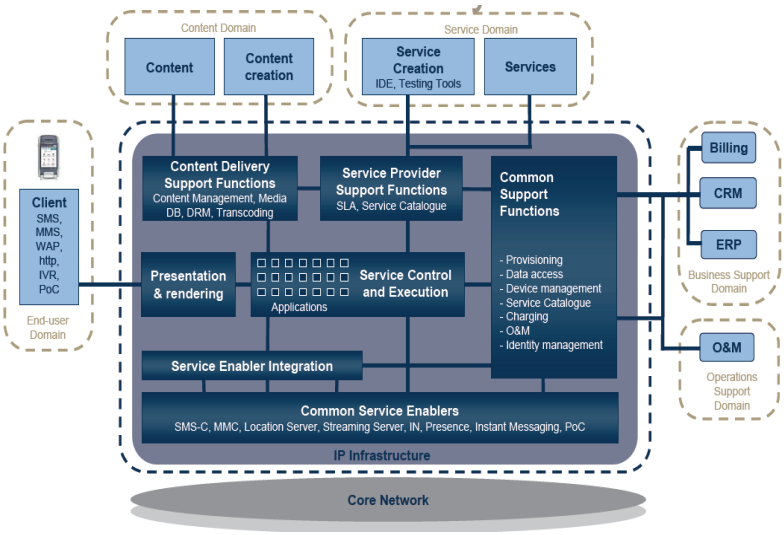



Figura 11 Arquitectura SDP Ericsson. Fuente: Ericsson

**Nokia Siemens**

Las SDP de esta compañía, conocida como Framework para el despliegue de servicios (SDF, Services Delivery Framework), reúne los conceptos adoptados por el grupo Moriana [9] y expone los siguientes beneficios [10]:

Implementación de nuevos servicios de manera rápida y a más bajo costo Reducción los costos de integración y desarrollo a través del reúso de funciones comunes entre servicios Decremento del costo de gestión de VAS Incremento en la calidad de aprovisionamiento de VAS Integración transparente con los BSS/OSS.

La
Figura 12 presenta la arquitectura propuesta por la compañía para una rápida creación, ejecución y despliegue de servicios avanzados de comunicaciones, y presenta los siguientes componentes.

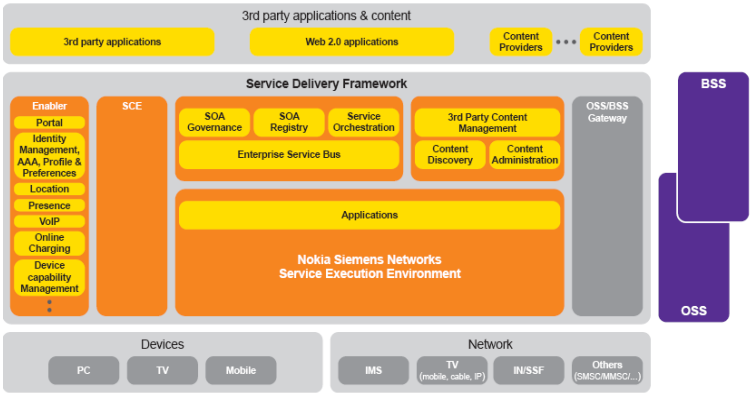



Figura 12 Arquitectura SDP Nokia Siemens Network. Fuente: Nokia Siemens Network

Ambiente de ejecución de servicio (SEE): suministra el entorno de ejecución para los servicios y aplicación.

Ambiente de creación de servicio (SCE): provee el conjunto de herramientas para construir todo tipo de servicios sobre la plataforma.

Gateways OSS/BSS: representa los componentes para la integración trasparente ente la SDF y los módulos OSS/BSS del proveedor de servicios de telecomunicaciones.

Ambiente SOA: contiene los módulos necesarios para la creación de servicios basados sobre el concepto SOA.

Gestión y entrega de contenido: proporciona el desarrollo y gestión de contenido por terceros.

Habilitadores: hace referencia a todo el conjunto de funciones comunes a los servicios.

**Red Hat**

Red Hat es la compañía responsable de la creación y mantenimiento de una distribución del sistema operativo GNU/Linux que lleva el mismo nombre: Red Hat Enterprise Linux, y de otros más. Fedora, así mismo, en el mundo del middleware patrocina jboss.org y distribuye la versión profesional bajo la marca JBoss Enterprise. Algunos años atrás, esta empresa adquirió completamente a Mobicents, una herramienta tecnológica en código abierto para la implementación de JAIN SLEE en un servidor Telco.

Con la combinación de Jboss[1] y el servidor Mobicents, Red Hat ha creado su plataforma de comunicaciones, una herramienta para la creación, ejecución y despliegue de VAS en la industria de las telecomunicaciones. La Figura 13 muestra la arquitectura para la SDP de Red Hat

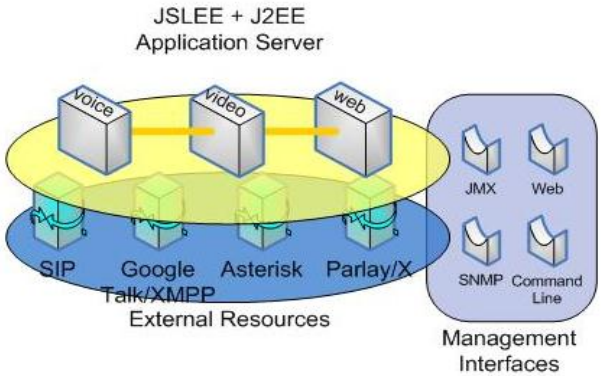



Figura 13 Plataforma de comunicaciones Jboss Fuente: Mobicents

[1] Servidor de aplicación JEE de código abierto e implementado en Java puro.

La ventaja de Mobicents recae en la implementación de la tecnología JAIN SLEE como motor de alto desempeño de una SDP. Las características más importantes de esta tecnología son tratadas en el Anexo B.

**ZTE**

La arquitectura de despliegue de servicios de la empresa ZTE se adapta al movimiento del mercado y a las nuevas tendencias tecnológicas. La plataforma contiene ambientes de ejecución, de creación de servicio y un nivel de abstracción de red. Algunas de las características más relevantes son mencionadas a continuación:

Plataforma que adapta tecnologías y servicios (GSM, WCDMA, CDMA2000, IMS, NGN).
Soporte al modelo de OSA. Interfaces abiertas multinivel como las API de Parlay y Servicios Web (WS, Web Services). Soporte de herramientas de desarrollo como el Lenguaje de Ejecucion de Proceso de Negocio(BPEL, Business Process Execution Language), SCE para Proveedor de Servicio(SPSCE, Service Provider SCE), SCE para Redes Inteligentes (INSCE, Intelligent Network SCE). Soporte de gestión para plataformas como Servicio de Mensajería Corta (SMS, Short Message Service), Servicio de Mensajería Multimedia (MMS, Multimedia Messaging System), Servicio Basado en Ubicación (LBS, Location Based Service), Java, respuesta de voz interactiva (IVR, Interactive, Voice, Response) , etc.

En la Figura 14 se observa la arquitectura funcional de referencia de la plataforma de despliegue de ZTE.

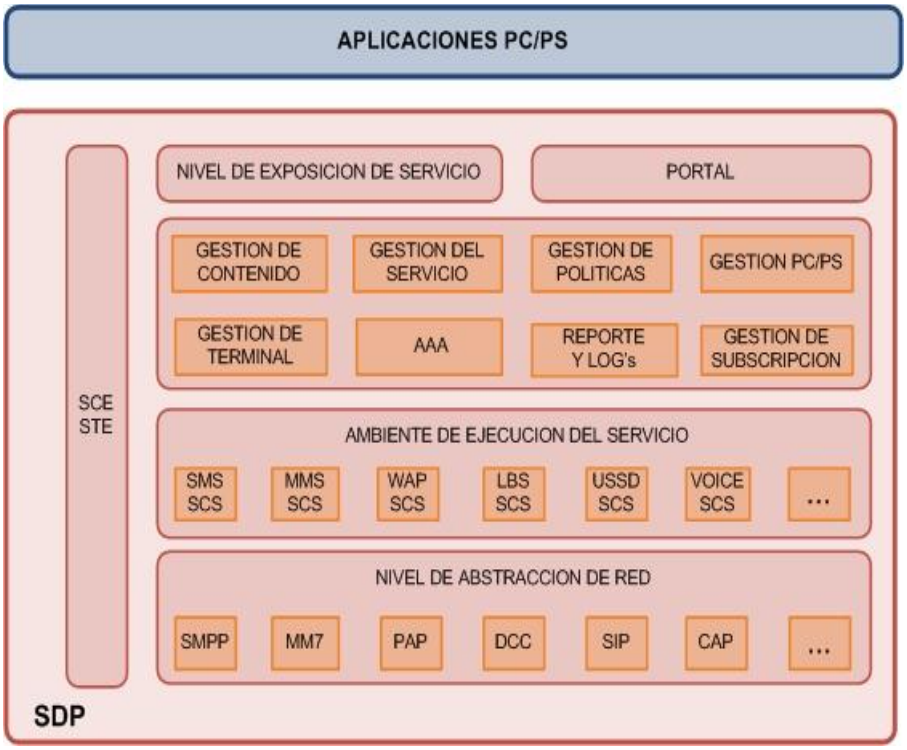


Figura 14 ZTE SDP. Fuente: ZTE

**SLEE ZTE**

Este componente es el centro de la arquitectura y realiza los procesos de gestión, configuración y composición con servicios básicos. Aquí la lógica es desarrollada y permite a diferentes parámetros como la definición de listas blancas[2] , listan negras[3] e implementación de los acuerdos de servicio ser configurados y ejecutados.

**SCE ZTE**

Este componente es una interfaz entre la red del operador y el diseñador del servicio. Es una entidad funcional usada para introducir rápidamente servicios de valor agregado en redes inteligentes. Este ambiente de creación, Figura 15, usa componentes básicos con el fin de construir nuevas aplicaciones las cuales serán finalmente probadas en herramientas como el Ambiente de Pruebas de Servicio (STE[4], Service Test Environment) para comprobar el buen funcionamiento del servicio dentro de la red de telecomunicaciones.

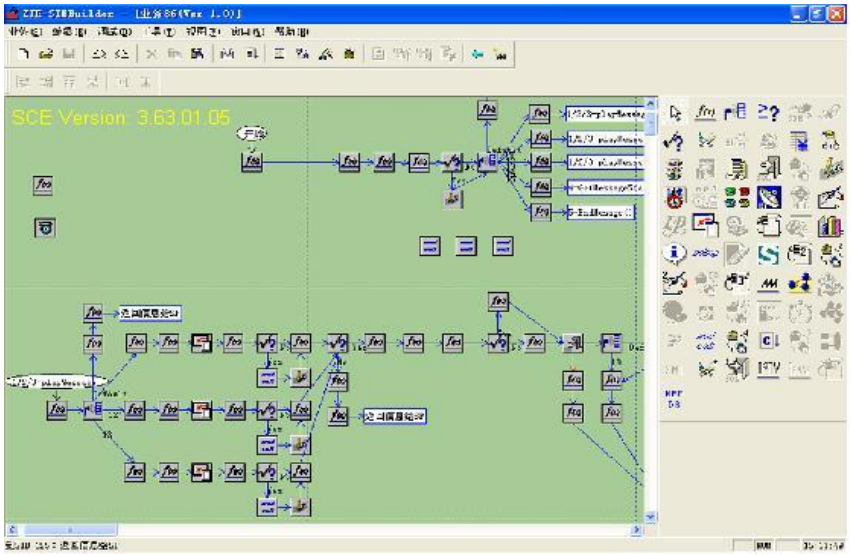



Figura 15 Parlay SCE- Fuente: ZTE

**Nivel de abstracción de red**

Este es el punto de acceso al centro de la SDP. Es el responsable para que los habilitadores del servicio accedan a los protocolos de red. Este nivel está compuesto de dos partes, (1) el módulo relacionado con interfaces de adaptación de servicio como mensajería corta, mensajería multimedia, localización, medios multimedia, presencia e IMS y (2) el módulo de interfaces de adaptación de voz como INAP, CAMEL, ISC, SIP y MAP.

[2] Listas generadas por el usuario con el fin de configurar los permisos de aceptación de números para recibir llamada. Servicio ONLY de ZTE.
[3] Listas generadas por el usuario con el fin de configurar los permisos de negación de números para no recibir llamada, Servicio ONLY de ZTE.
[4] Ambiente de prueba del servicio proporcionado por ZTE en la SDP con el fin de suministrar al desarrollador un conjunto de herramientas para comprobar la integridad del servicio.

## Opencloud

La arquitectura de RHINO, distribución comercial de esta compañía, está construida con los componentes más importantes de una plataforma de despliegue de servicios. La capa de creación de servicios, la de ejecución, integración y nivel de atracción conforman su completa estructura. La Figura 16 muestra los componentes de la plataforma.

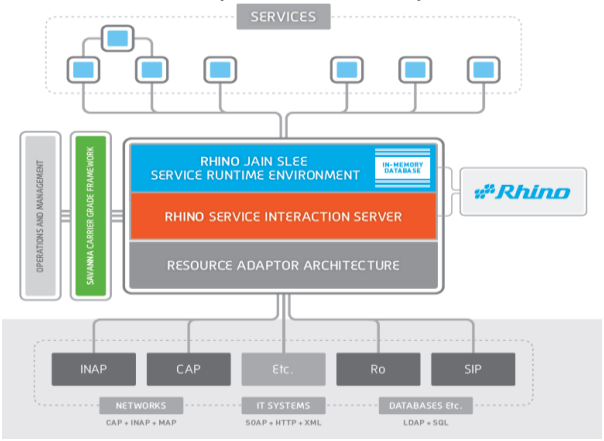



Figura 16 Plataforma de despliegue de RHINO. Fuente: OpenCloud

## Rhino JAIN SLEE

Es el corazón de la plataforma RHINO, el cual está constituido por un servidor de aplicaciones en tiempo real, diseñado y optimizado para aplicaciones de comunicaciones conducidas por eventos (modelo seguido en las aplicaciones del dominio telco). Este servidor soporta los requisitos de un operador en telecomunicaciones y es compatible totalmente con las especificaciones JAIN SLEE 1.0 (JSR 22) y JAIN SLEE 1.1 (JSR 240) [11].

Servidor de Interacción de Servicios RHINO

El Servidor de Interacción de Servicios RHINO (RHINO SIS[5], RHINO Service Interaction Server) es una poderosa, flexible y extensible plataforma de integración de servicios manejada por script[6] e interfaces de usuario. Este componente permite al desarrollador componer y gestionar nuevos servicios de redes SS7 e IMS. En la Figura 17 puede verse el contexto de implementación de servidor[7] de aplicaciones [12].

[5] Adopta la interacción de servicios para ampliar el concepto IMS SCIM incluyendo la composición de nuevos servicios como IN/SS7, IMS/IP, Web Services.
[6] Conjunto de instrucciones para automatizar tareas.
[7] RHINO SIS corre sobre el servidor de aplicaciones RHINO, RHINO JAIN SLEE.


Figura 17 RHINO SIS. Fuente: RHINO

Este servidor nace con la idea de extender la ventaja del Gestor de interacción de capacidades de Servicio (SCIM, Service Capability Interaction Manage. SCIM fue pensado con la intención de lograr la interacción entre las capacidades de una red IMS dentro de un servidor de aplicación SIP. Fue propuesto y desarrollado por el Proyecto Conjunto de Tercera Generación (3GPP, 3rd Generation Partnership Project) dentro de especificación 6 de su arquitectura de servicios. Por su parte, RHINO SIS acoge esta propuesta y lo extiende a tecnologías de redes inteligentes basadas en conmutación de circuitos. La
Tabla 1 resume las principales características de RHINO SIS.

| CARACTERÍSTICA | DESCRIPCIÓN |
|---|---|
| Manejo de interacción IN-SIS y SIP-SIS | SIP –SIS: interacción de servicio para servicios IMS<br>IN-SIS: Interacción de servicios para servicios conmutados IN |
| Combinación de servicios locales y/o externos | Con el RHINO SIS, los servicios pueden ser locales (desplegados en el SIS) o estar contenidos en plataformas externas. La ubicación física de los servicios es tranparente al SIS. Esto permite que nuevos servicios sean combinados y que los servicios contenidos en redes heredadas migren suavemente a la plataforma RHINO. |
| Gestión y operación manejables | El uso de RHINO SIS permite:<br>Reducir los costos de operación de la red debido a que usa los recurso de red existentes de manera más eficiente y que sean usados de múltiples maneras<br>Permite que nuevos servicios sean implementados como servicios puros JAIN SLEE o como combinaciones de servicios existentes. Esto suministra una mejor gestión de interacción y una migración a estructuras basadas en IMS. |

Tabla 1Características RHINO SIS Fuente: RHINO

### Arquitectura de adaptador de recursos

Este nivel se compone de la arquitectura de Adaptador de Recursos (RA, Resources Adapter) de JAIN SLEE. Esta suministra un conjunto de tecnologías que son introducidas[8] dentro de la plataforma con el fin de inter-operar diferentes sistemas. La Tabla 2 resume las tecnologías soportadas dentro de la plataforma RHINO las cuales son utilizadas para desarrollar servicios de próxima generación [11].

| RA | DESCRIPCIÓN |
|---|---|
| SIP | Protocolo de Inicio de Sesión |
| ISC | 3GPP/Soporte de Extensión |
| Diameter | Interfaz IMS |
| MM7 | Mensajería MMS |
| CAP | SS7 |
| INAP | SS7 |
| MAP | SS7 |
| HTPP | Integración Empresarial |
| SOAP | Servicios Web |
| JEE | Integración Empresarial |
| LDAP | Integración de servidor de directorio |
| JDBC | Integración a base de datos |

Tabla 2 RA de RHINO Fuente: RHINO

### Dominio del operador

La plataforma RHINO reúne los requerimientos[9] específicos al operador como desempeño, fiabilidad, disponibilidad, facilidad de crecimiento, contención contra fallas, redundancia, gestión y mantenimiento. Este conjunto de parámetros son tenidos en cuenta con el fin de crear servicios de valor agregado desde la perspectiva de un operador de telecomunicaciones y la unión de tecnologías IT. La Figura 18 muestra la relación entre IT y Telco.

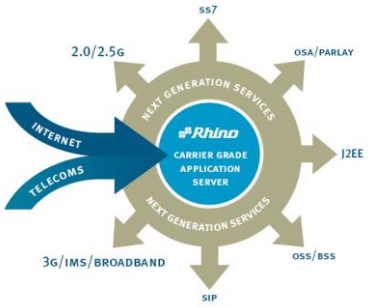



Figura 18 VAS en NGN Fuente: RHINO

[8] Arquitectura basada en “plugged-into”. introducción de módulos sobre cierta infraestructura, en este caso JAIN SLEE.
[9] Los requerimientos generalmente aceptados proceden de una serie de documentos de sistemas de conmutación de Bellcore, Investigación de comunicaciones BELL.

**Resumen**

| ÍTEM | ORACLE | MICROSOFT | ERICSSON | BEA | IMB | NOKIA SIEMENS | RED HAT |
|---|---|---|---|---|---|---|---|
| **Lenguaje base de programación** | Java | .NET | Java | Java | Java | Java | Java |
| **Servidores para el ambiente de ejecución** | SIP Servlet | SIP Servlet .NET | SIP Servlet | SIP Servlet | SIP Servlet | Rhino JAIN SLEE jNetX | Mobicents JAIN SLEE |
| **Políticas de Administración** | SI | SI | SI | SI | SI | SI | SI |
| **Servidores de preferencias/perfiles** | SI | SI | SI | SI | SI | SI | SI |
| **Soporte Parlay** | SI | NO | SI | SI | SI | SI | SI |
| **Localización / Presencia** | SI | SI | SI | SI | SI | SI | SI |
| **Herramientas /componente IT** | SI | SI | SI | SI | SI | SI | SI |
| **Compatibilidad con IMS** | SI | SI | SI | SI | SI | SI | SI |
| **Soporte para redes fijas y móviles** | SI | SI | SI | SI | SI | SI | SI |
| **Soporte OSS/BSS** | SI | SI | SI | SI | SI | SI | SI |
| **Servidor de ambiente de ejecución basado en modelo de composición y reúso** | NO | NO | NO | NO | NO | NO | SI |
| **Servidor de ambiente de ejecución con soporte de más de un protocolo** | SIP HTTP | SIP HTTP | SIP HTTP | SIP HTTP | SIP HTTP | SIP SS7 INAP CAP MAP Diameter Otros | SIP SS7 INAP CAP MAP Diameter Otros |

Tabla 3 Comparativa entre servidores de telecomunicaciones utilizados o proporcionados por diferentes compañías

Son muchas las SDP que existen el mercado, y muchas maneras de implementarlo. Debido a la no estandarización del concepto, las SDP terminan siendo ofrecidas como un producto específico a un proveedor para la creación, ejecución, desligue, facturación y gestión de servicios de telecomunicaciones, y no como un modelo general para el aprovisionamiento de

VAS en cualquier NGN. Sin embargo, todas estas SDP introducen cuatro componentes clave en la oferta de servicios: Nivel de ejecución, Nivel de creación, Nivel de adaptación de red y acceso a terceros. Todos ellos analizados en detalle dentro de la monografía

A manera de resumen, se presenta en la Tabla 3 una comparación entre las SDP comerciales según las características más importantes de toda plataforma para la creación, ejecución y despliegue de VAS en NGN.

# Anexo B

# Plataforma Multiservicios de EMCALI

El presente anexo muestra una descripción general de la NGN del operador de telecomunicaciones EMCALI EICE-ESP, de su componente principal para el aprovisionamiento de VAS y se presenta la lista de servicios desplegados dentro de la plataforma mutiservicios.

## B.1 NGN EMCALI

La red de EMCALI es el claro ejemplo de la migración tecnológica que demanda el mercado de los nuevos servicios de telecomunicaciones, pasando de una red tradicional conformada por estructuras verticales, la clásica Red telefónica Publica Conmutada (PSTN, Public Switched Telephone Network), hasta modelos horizontales como las NGN. En la medida que el mercado de las comunicaciones ha evolucionado, EMCALI tuvo la necesidad de adaptarse a las nuevas tendencias tecnológicas con el fin de permanecer en el negocio. Motivo por el cual, la empresa se adentra a las redes de nueva generación[10] migrando sus estructuras tradicionales a un mundo basado en IP. EL camino es largo para este operador principalmente porque su modelo de ofrecimiento de servicios de nueva generación pareciera aun no estar concebido en plenitud y peor aún, su modelo de negocio sigue sin ser orientado al mundo NGN manteniendo el implementado en su red PSTN.

La Figura 19 muestra la arquitectura de referencia de red de próxima generación de EMCALI.

**Nivel de acceso**

Este nivel lo componen tecnologías heredadas como PSTN[11], ATM, RDSI; móviles como GSM/GPRS 2G[12], inalámbricas como WIMAX[13], y telefonía IP sobre dispositivos SIP y Softphones [14].

En relación a los dispositivos de interconexión entre las tecnologías presentes y el nivel de transporte de la NGN de EMCALI, ZTE provee los siguientes equipos de acceso:

[10] la red NGN lleva en operación 3 años
[11] En este punto, se realiza la interconexión con las empresas de telefonía móvil celular. La conexión se lleva a cabo entre los equipos de conmutación de esta red y de de la red del operador móvil.
[12] para trabajadores de la empresa y telefonía rural
[13] está implementando en cierta porción no muy amplia y solo para la parte urbana de Cali
[14] Para usuarios NGN de EMCALI

Media Gateway ZXMSG 9000: es un equipo de transporte del flujo de datos y señalización. Según la configuración establecida, estos dispositivos actúan como Gateway de Troncal (TG, Trunk Gateway), Gateway de señalización (SG, Signaling Gateway) y Gateway de acceso (AG, Access Gateway).

ZXMSG 9000 como AG: cuando es utilizado este modo, el equipo se ubica en el nivel de acceso de la red de paquetes y permite a suscriptores PSTN, RDSI, V5 y DSL, acceder a la red IP.

ZXMSG 9000 como TG: cando se utiliza con este fin, el dispositivo es ubicado en el nivel central de la red de paquetes. Esta configuración permite el acceso tanto a los suscriptores de troncal[15] número 7 como a los suscriptores PRI[16].

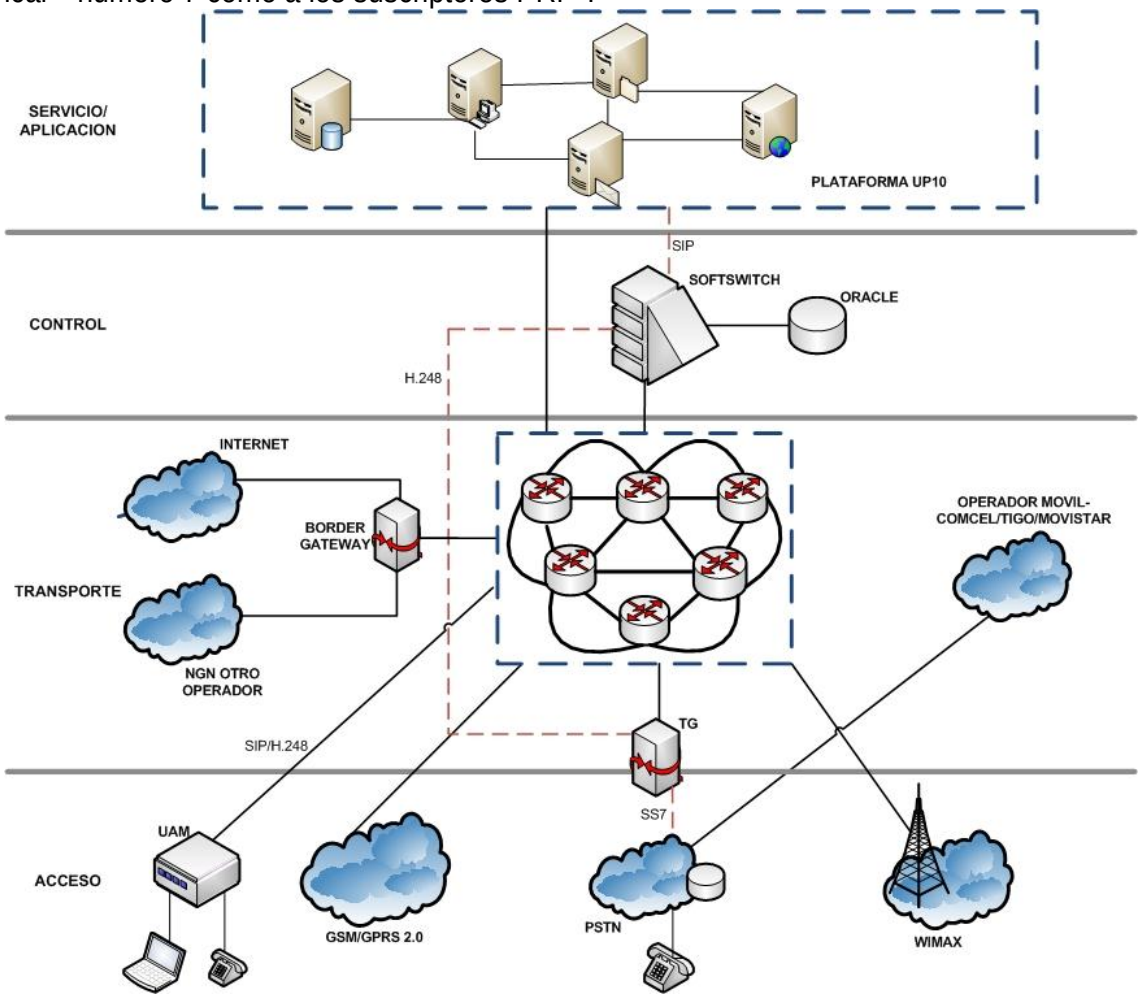



Figura 19 NGN de EMCALI.

[15] Altos volúmenes de tráfico sobre troncales que manejan señalización número 7.
[16] Canal primario definido en RDSI para altos volúmenes de tráfico

ZXMSG 9000 como SG: cuando el equipo es implementado en este modo, se ubica en el núcleo de la red de paquetes y realiza la interoperabilidad entre los mensajes SS7 de la PSTN y los mensajes de la familia de protocolos SIGTRAN[17] de la red IP.

Border Gateway ZXSS10 B100: es un dispositivo de frontera que enlaza la red pública de un proveedor de servicios a la red privada del proveedor. Este direccionamiento de redes públicas y privadas es obtenido por la traslación de dirección de red (NAT, Network Adress Translation) o vía HTTP.

Dispositivo de acceso integrado (IAD, Integrated Access Device): dispositivo de acceso diseñado para usuarios individuales y pequeñas organizaciones empresariales. Este permite a dispositivos como PC, teléfonos y máquinas de fax acceder la red de conmutación de paquetes.

**Nivel de Transporte**

Este nivel está conformado por el núcleo de Conmutación de Etiqueta Multiprotocolo (MPLS[18], Multiprotocol Label Switching) con elementos de red de capa 3. Los seis anillos (formados por los nodos Colón, Guabito, San Fernando, Versalles, Poblado y Parcelaciones) dan completo soporte a toda la red multiservicio. En este nivel se realiza la conexión con la red de Internet y con redes NGN de otro operador. Los enrutadores están sujetos a un diseño de redundancia 1 a 1 y diseñados para soportar una gran variedad de protocolos. La Tabla 4, resume los protocolos manejados por estos dispositivos.

| TIPO DE PROTOCOLO | PROTOCOLOS |
|---|---|
| De nivel de enlace | PPP, MPPP |
| Nivel de Red | IP, ICMP, ARP, V-SWITCH, SMARTGROUP |
| Nivel de Transporte | TCP y UDP |
| De Enrutamiento | RIP v1/v2, OSPF v2, BGP4, IS-IS, IPv6, RIPng, OSPF v3, BGP4+, IS-IS6, MPLS/VPN, VPWS, VPLS, QoS, RSVP TE, políticas de enrutamiento y funciones de carga compartida |
| Tunel | GRE |
| Nivel de aplicación | Telnet, FTP y TFTP |
| Control de red y aplicación | NAT, ACL y URPF |
| Protocolos NM | SNMP v1/v2/v3, RMON v1 y NTP |

Tabla 4 Protocolos soportados por los dispositivos del nivel de transporte de la NGN

[17] Grupo de trabajo del IETF que ha desarrollado una serie de protocolos para transportar SS7 por redes IP.

[18] Es un mecanismo de transporte de datos estándar creado por la IETF y definido en el RFC 3031. Opera entre la capa de enlace de datos y la capa de red del modelo OSI. Fue diseñado para unificar el servicio de transporte de datos para las redes basadas en circuitos y las basadas en paquetes. Puede ser utilizado para transportar diferentes tipos de tráfico, incluyendo tráfico de voz y de paquetes IP.

El border Gateway conecta los clientes NGN que usan herramientas software para el establecimiento de llamadas como el Xlite y el EMVOZ[19] a través de la red de internet.

**Nivel de Control**

Este nivel es el más importante de toda red NGN. En este punto se realiza el control sobre las llamadas, señalización, control de flujos de información, funciones AAA[20], entre muchas otras. El componente esencial en esta arquitectura es el Softswitch, encargado de realizar todas funciones de procesamiento de control integrado como procesamiento de llamadas, adaptación de los protocolos de acceso, autenticación, enrutamiento, asignación de recursos, almacenamiento de registro detallado de llamada (CDR, Call Detail Record) y la interconexión e inter-funcionamiento de todos los nodos de la plataforma multiservicio de EMCALI, de igual manera, el Softswitch suministra todos los servicio básicos de llamada, servicios suplementarios, servicios de valor agregado basados en web y servicios multimedia punto a punto de la red PSTN.

En cuanto a su arquitectura, el nivel físico está constituido por un componente de procesamiento en tiempo real, un servidor de base de datos, un componente de OSS y un sistema de conmutación de red que interconecta todos los módulos internos. La capa lógica esa dividida en un de un nivel de servicio, un nivel de control y un nivel de adaptación.

Nivel de servicio: en este nivel se lleva a cabo el enlace con los niveles superiores y a los sistemas de información empresarial por medio del gestor del servicio y del gestor de datos respectivamente. El gestor del servicio sirve como punto de control para la interacción entre el Softswitch y el Punto de Control de Servicio (SCP, Service Control Point) del servidor de aplicaciones, y el gestor de datos suministra una interfaz unificada a la base de datos de EMCALI.

Nivel de control: este permite el control sobre la funciones de procesamiento de llamada, de acceso a protocolos, interconexión, funcionamiento y soporte a toda la infraestructura de EMCALI. Este lo componen los módulos de gestor de recursos, control de llamada y conexión con otros nodos de control. el primero asigna los recursos más adecuados para manejar la solicitud de algún servicio en particular, el segundo, se encarga de realizar un control de llamada unificado y finalmente, el modulo SIP/SIP-T soporta la interconexión entre diferentes Softswitch.

Nivel de adaptación: son todos aquellos protocolos manejados por el nodo que permiten la conexión y señalización con toda la infraestructura NGN del operador (SIP, H.323, SS7-INAP/CAP/MAP, H.248. entre muchos otros).

La Figura 20 muestra la distribución de los bloques lógicos de la arquitectura del Softswitch

[19] Programa desarrollado por ZTE con el fin de suministrar mensajería multimedia para los clientes de la empresa. Este programa aparte de la facilidades de mensajería permite hacer llamadas locales ilimitadas
[20] Siglas de Autenticación, Autorización, Facturación (en inglés Accounting)


Figura 20 Componentes lógicos del Softswitch Fuente: ZTE

Este componente en conjunto con los servidores de aplicación, proveen a la NGN de EMCALI, una variedad de servicios avanzados de telecomunicaciones, desde servicios tradicionales inteligentes hasta servicios mejorados en redes IP.

**Nivel de aplicación y servicio**

Este nivel está conformado por la plataforma de despliegue UP10. Esta plataforma la componen un número de servidores y módulos de control para la prestación de servicios. A continuación se detalla más a fondo este nivel en razón que el análisis del presente proyecto se enfoca en la creación, ejecución y despliegue de servicios de valor agregado soportados sobre el nivel de aplicación y servicio.

## B.2 Plataforma UP10

Esta plataforma, conocida a nivel productivo como ZXUP10, cumple con las especificaciones de la ITU, ETSI y el IETF. La arquitectura, Figura 21, se basa principalmente en la especificación de OSA/Parlay versión 3 (ETSI ES 201 915) y se encuentra dividida en cuatro bloques funcionales, (1) el nivel de aplicación, (2) el nivel de control, (3) el nivel de adaptación y (4) el nivel de recursos.

Architecture of the third generation unified platform

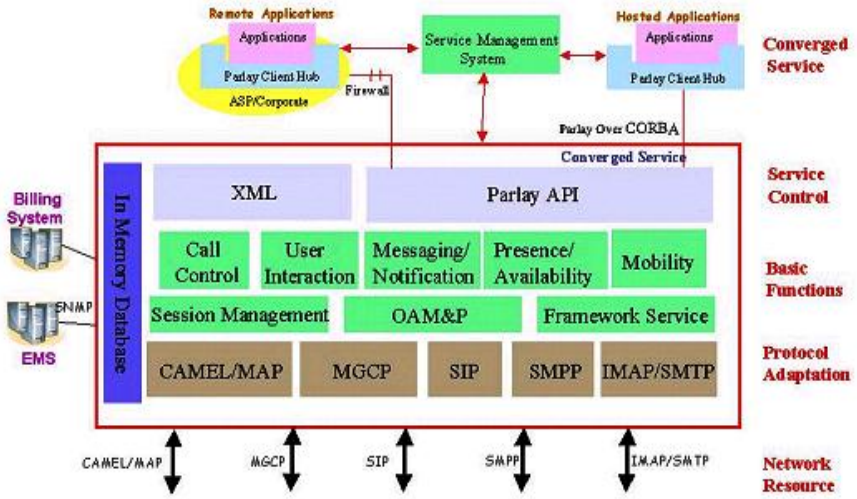



Figura 21 Arquitectura de referencia ZXUP10. Fuente: ZTE

**Componentes lógicos**

**Nivel de aplicación**

En este nivel residen las aplicaciones que usa una o más SCF[21]. Estas aplicaciones son registradas y autenticadas dentro del Framework API y luego (véase sección 2.1.3.5), la aplicación tiene la autorización para el uso de las SCF sobre el servidor Parlay. Este mecanismo de seguridad implementado por la plataforma a través del cliente parlay (PCH, Parlay Client Hub), permite a terceros hacer uso de los recursos y capacidades del operador para construir nuevos servicios. Sin embargo, aunque se cuente con dicho componente, el acceso está totalmente cerrado en gran parte por problemas contractuales presentes entre el operador EMCALI y uno de sus proveedores de equipos de telecomunicaciones, ZTE. Este inconveniente ha traído todo tipo de restricciones en cuanto al uso y accedo de la tecnología OSA/Parlay para desarrollo de terceros, motivo por el cual, esta tecnología no ha podido ser implementada dentro de trabajos externos a los desarrollados y ejecutados por la compañía ZTE.

**Nivel de control de servicio.**

Este nivel de control implementa la API de Parlay y registra los módulos SCF con el Framework de forma tal, que una aplicación sabe si puede obtener cierto servicio del servidor Parlay.

[21] control de llamada, interacción de usuario, gestión de movilidad, mensajería, facturación, gestión de conectividad, framework y operación y mantenimiento [16].

**Nivel de adaptación.**

Este nivel mapea los diferentes protocolos de red[22] a una interfaz unificada suministrada por la API Parlay con el fin de ser usada por cualquier modulo SCF.

**Nivel de recursos.**

Conformado por el Softwicth, centro de conmutación móvil, servidor de medios, servidor de correo y centro de mensajería corta.

Todo este conjunto de niveles se encuentran encapsulados en componentes físicos dentro de la plataforma. A continuación se describe la arquitectura física de la plataforma ZXUP10.

**Componentes físicos**

La plataforma ZXUP10 es un infraestructura integrada conformada por elementos como el Parlay Gateway, servidor de medios, servidor de aplicaciones, servidor Texto a Habla (TTS, Text to Speech), Gateway de señalización y consola de operación, administración y mantenimiento.

**Parlay Gateway[23]**

Este es el componente más importante de la plataforma. De él depende la adaptación de red, el suministro de la API de acuerdo al protocolo y el acceso a terceros para el desarrollo de nuevos servicios. Lo conforman tres módulos funcionales, (1) el adaptador de protocolo, (2) el bloque de funciones básicas y (3) el bloque de control de servicio. Sus diferentes niveles se aprecian en la Figura 22.

Adaptador de protocolo: corresponde al nivel de adaptación de red. Este punto permite encapsular los diferentes protocolos que serán usados para la construcción de nuevos servicios.

Módulo de Función Básica: este componente se encarga de la gestión y control del Parlay Gateway. También contiene las capacidades básicas para la construcción de servicio (SCF) y el Framework según lo establecido por la especificación de OSA/Parlay.

Framework: este sub-modulo ofrece funciones de gestión orientadas al usuario y orientada a la aplicación. La primera es en función de la confiabilidad y seguridad, y la segunda, en relación al registro, autenticación y autorización.

Call Control: sub-modulo encargado de suministrar las capacidades básicas de control de llamada y del establecimiento de sesiones.

[22] soporta protocolos como CAMEL, SIP, MGCP,IMAP, SMTP, SMPP [16].
[23] reúne los niveles de control y adaptación de la plataforma ZXUP10

Interacción de usuario: parte que suministra la comunicación de información entre usuarios.

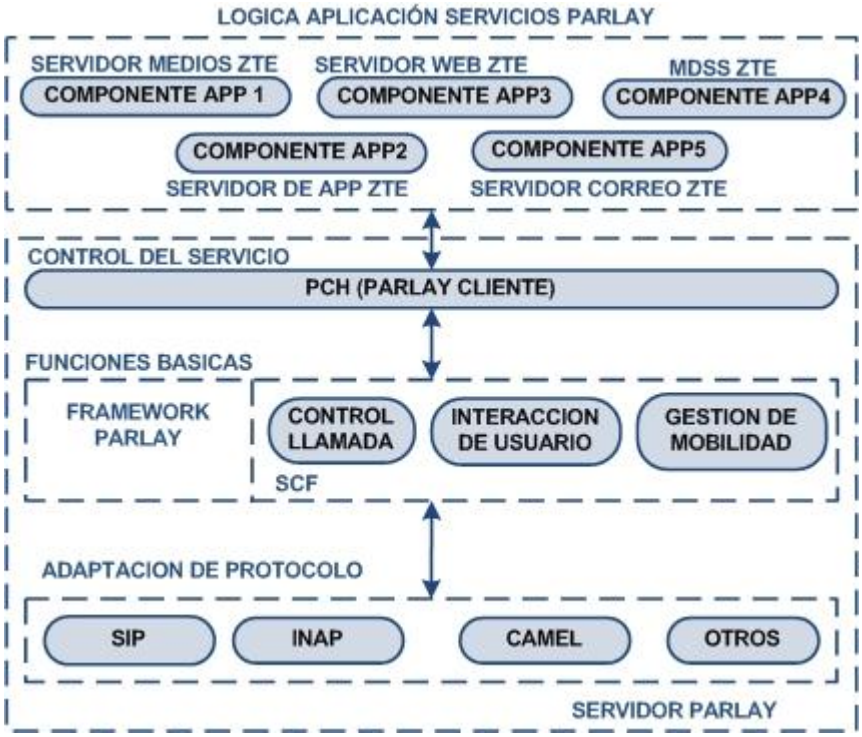


Figura 22 Módulos del Parlay Gateway. Fuente: ZTE

Gestión de movilidad: provee el servicio de ubicación geográfica y estado de actividad del usuario.

Mensaje general: representa el sub-modulo para enviar, almacenar y recibir mensajes de correo de voz o electrónicos.

Charging: modulo que genera y gestiona las sesiones de cobro de los servicios. En este punto se realiza la generación de CDR.

Gestión de conectividad: sub-modulo usado para gestionar la conectividad entre la red de la empresa y la red del proveedor del servicio. En este punto se configurar los parámetros de QoS para enviar los paquetes de información ente las diferentes redes.

OA&M: sub-modulo encargado de realizar funciones generales de gestión, administración y mantenimiento.

Módulo de control de Servicio: este componente es el encargado de implementar la API de Parlay, lo que hace posible, registrar las SCF al framework de forma que cualquier aplicación haga uso de ellas. Este módulo implementa la tecnología PCH con el fin de que terceros accedan a las SCF para desarrollar una gran variedad de servicios sin la necesidad de conocer los protocolos específicos de red. La Figura 23 representa la manera como las aplicaciones convergentes a través del API Parlay usan las SCF para suministrar servicios.

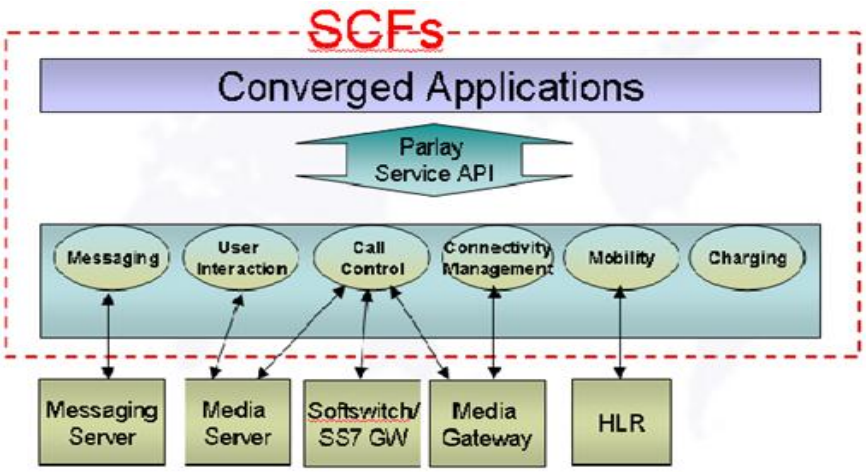



Figura 23 Interacción de las aplicaciones con las SCF Fuente: ZTE

**Servidor de aplicaciones**

En este componente se ejecuta la lógica de las aplicaciones. Los servidores aquí contenidos hacen uso de las capacidades del Gateway habiendo primero pasado por los mecanismos de registro, autenticación y autorización. Los servidores de aplicaciones pueden ser implementados en máquinas diferentes o integrados dentro el mismo servidor Parlay [24] pero generalmente, estos elementos son desplegados por terceros en el modelo OSA.

**Servidor de medios.**

Brinda el soporte para las funciones especializadas que requieren servicios de alto nivel como conferencia, gestión, mantenimiento, conversión de diferentes algoritmos de codificación, etc.

**Servidor TTS.**

Es un servidor que convierte un conjunto de palabras en flujo de media la cual es transmitida al servidor de medios para ser reproducido a los usuarios.

**Gateway de señalización.**

Componente encargado de conectar los protocolos de señalización SS7 (INAP, CAP, MAP) con los protocolos de señalización IP (SIP, MGCP).

**Consola de operación, mantenimiento y gestión.**

Este componente permite el manejo y mantenimiento del sistema ZXUP10. Implementa funciones como configuración y modificación de datos, señalización de ruta, mantenimiento de alarmas, solicitud de información de llamada, gestión del operador, entre otras.

[24] para la implementación de servicio, RHINO JAIN SLEE estaría dentro de otra máquina y actuaria como tercero

La Figura 24 muestra la como se realiza la conexión del Parlay Gateway con el Softswitch de ECMALI.

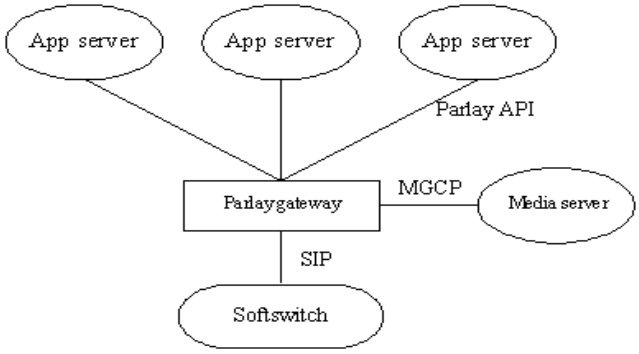



Figura 24 Conexión entre Parlay Gateway y Softswitch. Fuente: ZTE

**Servicios soportados por la plataforma ZXUP10**

Esta plataforma suministra una variedad de servicios personalizables sobre NGN como llamada web, conferencia web, web 800, correo de voz, Un Numero de enlace (ONLY, One Numer Link You) y servicio de mensajería unificada (UMS, Unified Messaging Service). A continuación se realiza una breve descripción de cada uno de ellos.

Correo de voz: un suscriptor de este servicio puede escuchar a través de correo (acceso vía web) o por llamada telefónica convencional, el mensaje que cualquier usuario deja por medio del lanzamiento de la respuesta de voz interactiva (IVR, Interactive, Voice, Response) en el buzón de voz del suscriptor del servicio. Normalmente, este servicio es usado en conjunto con el servicio ONLY y UMS.

Conferencia Web: Este servicio está desarrollado para múltiples redes y consiste principalmente en el establecimiento de llamadas para conferencia basada en la web. Un usuario puede atender una conferencia a través del terminal soft, conjunto de teléfonos normales, terminales SIP o teléfonos móviles. Toda la información es desplegada en páginas web en tiempo real. Es posible también crear sub-conferencias a través de estas páginas.

Web 800: este servicio es similar al servicio prestado sobre la red inteligente en relación a la llamada 01 8000 con la diferencia que se realiza sobre la web. Entre las compañías que pueden incluir este servicio se destacan las agencias de viajes, compañías de aerolíneas, empresas de transporte e instituciones educacionales.

UMS: este servicio es pensado para usuarios individuales donde se conjuga una serie de características para brindar acceso unificado a varios servicios. Entre ellos, acceso a terminales de usuarios (teléfonos móviles, teléfonos SIP, softphones), e integración de servicios de voz, video, datos y fax. La información almacenada en un punto central del sistema, es accedida por cualquiera de los siguientes dispositivos: PC (vía correo electrónico),

fax, telefonía (vía servicio correo de voz), y aplicaciones de mensajería (MSN, Xlite, EMVOZ, SMS)

ONLY (): este es un servicio de movilidad en cual el suscriptor usa un único número personal de telecomunicaciones (PTN, Personal Telecommunicatin Number) a fin de acceder a cualquier red y recibir llamadas a través de ellas. Según la configuración del usuario, la llamada entrante al PTN, puede ser direccionada a números diferentes. Esto permite al usuario estar siempre en contacto y disponible, bien sea en su teléfono SIP, en su teléfono móvil, en la oficina, en la casa, etc. La configuración de los números telefónicos se hace vía web por el suscriptor del servicio

Llamada Web: este servicio permite al usuario originar llamadas con solo hacer click sobre alguno de los números desplegados en una página web. La conexión RTP se realiza entre dispositivos SIP, teléfonos soft, teléfonos convencionales PSTN, telefonía IAD y dispositivos móviles.

SoftDA: este servicio conocido comercialmente como EMVOZ[25], presta características semejantes al MSN de la compañía Microsoft. Entre sus funciones se encuentran: mensajería, sesiones de llamada, sesiones de videoconferencia, transferencia de archivos, video llamada, entre muchas otras. La característica diferencial en relación a otros clientes de mensajera comerciales, radica en el establecimiento de llamadas locales bien sea a usuarios NGN o usuarios PSTN.

[25] Actualmente no está comercializado por EMCALI.

# Anexo C

# MOM en ESB

El presente anexo muestra las características del concepto de Mediación Orientada al Mensaje, núcleo principal en la mensajería empresarial de la arquitectura de servicios empresariales y que contribuye a la estrategia de integración empresarial [13].

**Message Oriented Middleware:**

MOM es un concepto trabajados por más de dos décadas. Con los años, las nuevas ideas se han introducido, nuevos proveedores se han movido en el espacio de MOM, y las nuevas normas continúan evolucionando relacionadas a la mensajería y los conceptos de MOM. Mientras que la tecnología en mensajería continúa mejorando, también hay una serie de conceptos básicos que siguen siendo similares a través de muchas implementaciones de MOM. Cuando una aplicación se comunica de este modo, se está actuando ya sea como productor o un consumidor. Remitentes (productores) producen mensajes y receptores (consumidores) consumen mensajes. Una aplicación puede ser tanto un productor y un consumidor al mismo tiempo.

**Desacoplamiento abstracto:**

Los productores y los consumidores están débilmente acoplados y no se encuentran directamente vinculados entre sí, en cambio, son abstractamente conectados entre sí a través de canales virtuales llamados publicación y suscripción de canales o canales punto a punto.

El productor no tiene por qué saber qué aplicaciones están recibiendo un mensaje, o en algunos casos incluso cuántos lo están recibiendo. Del mismo modo, el consumidor no tiene por qué saber qué aplicaciones están enviando los datos, sólo sabe que recibe un mensaje y actúa sobre él. Si se requiere una respuesta, este hecho está codificado en el mensaje a través de la presencia de un "ReplyTo" destino para identificar el canal en el que enviara la respuesta.

Dependiendo de la aplicación, las definiciones de canal virtual están o codificadas en la solicitud, o se describen y se configuran administrativamente con una herramienta, con los detalles de la configuración almacenada en un directorio de servicio. Incluso si los datos de configuración de canales se almacenan en un directorio de servicio, la creación de los canales de mensajes debe ser codificados en la aplicación cuando se utiliza una base de MOM. Con un ESB, los detalles de la creación y gestión de los canales de mensajería se encapsulan en un entorno gestionado por el contenedor que está configurado en lugar de ser escrito en código de aplicación.

**Modelo de mensajería: Publicar/Suscribir y punto a punto:**

En el modelo publicar/suscribir, varios consumidores pueden registrarse con un interés, o suscribirse a un tema. El productor envía un mensaje en ese canal mediante la publicación en ese tema. Cada suscriptor recibe una copia de ese mensaje.

En el modelo de punto a punto, sólo un consumidor puede recibir un mensaje que se envía a una cola. Como se muestra en la Figura 25, una cola de punto a punto puede tener varios consumidores escuchar a los efectos de equilibrio de carga o "copia de seguridad en caliente", sin embargo, sólo un receptor puede consumir cada mensaje individual. También puede ser que el receptor no escuche el mensaje, en cuyo caso el mensaje se queda en la cola hasta que un receptor se une a la cola para recuperar los mensajes. En el modelo publicación y suscripción, mensajes no marcados como confiables pueden ser descartados si el suscriptor no ha registrado para recibirlos en el momento de su publicación.

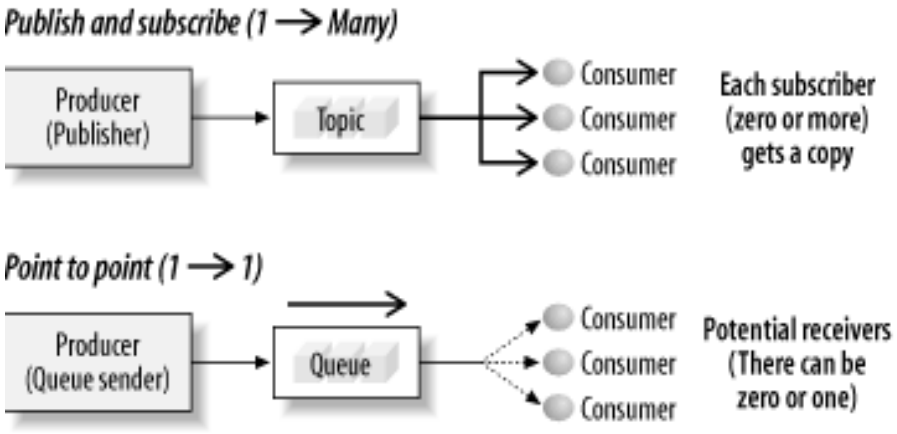



Figura 25 Modelo común de mensajes incluyendo publicar/suscribir y punto a punto.
Figura x.

La elección del modelo a utilizar en gran medida puede depender de cuántos consumidores necesitan ver las copias duplicadas de un mismo mensaje. Por ejemplo, en una cadena de suministro, un proveedor que desee transmitir las reducciones de precios especiales a sus compradores en una manera de uno a muchos con el modelo de mensajería publicación/suscripción, todos recibirían la misma información. Sin embargo, si un comprador particular, quiere responder a ese precio inferior al comprar los productos publicitados, requerirá una conversación punto a punto que involucra sólo a las dos partes: el comprador y el vendedor.

**Listas de Control de Acceso:**

Mecanismos de Listas de Control de Acceso (Access Control List (ACL)) puede permitir a los administradores dar acceso a diferentes niveles de la jerarquía y especificar las rutas o temas. Dominios de seguridad se pueden implementar de manera que los individuos o grupos pueden acceder con los debidos permisos, ya sea positiva o negativamente, a las vías o a los tópicos

individuales dentro de un espacio de nombres basado en jerarquías en el destino de la mensajería. Por ejemplo, usted puede especificar administrativamente: "Conceder este grupo de usuarios de todo el acceso a la gráfica de los temas que coinciden con la expresión. #. Doméstica. #, con exclusión de estos tres usuarios." Si el espacio de nombres de cola de punto a punto tiene una forma similar, el acceso es permitido.

En un ESB, esta capacidad contribuye a la autonomía federada entre los departamentos y unidades de negocio. Cada segmento de ESB puede residir en una unidad de negocios individuales y tienen su propio control local sobre las ACL.

**Fiabilidad Asincrónica.**

Con asincronía viene la necesidad de fiabilidad. Cuando una aplicación envía un mensaje asincrónico, a menudo tiene que ser una especie de garantía de que el mensaje va a llegar a donde necesita ir. Hay tres partes fundamentales de mensajería fiable: la autonomía de mensajes, almacenaje y envío, y la semántica del mensaje subyacente acuse de recibo.

**Mensaje de Autonomía:**

Los mensajes son autónomos, entidades autónomas, que representan una transacción comercial. Una vez que un productor de mensajes envía un mensaje, su papel se ha completado. Si el mensaje está bien designado para la entrega fiable, las garantías del sistema de mensajería que se recibe de todos los interesados, y la ESB asegura que llega a la meta deseada formato de datos. Este contrato entre el cliente que envía y el sistema de mensajería es muy similar a la del contrato entre un cliente JDBC y la base de datos. Una vez que los datos son fiables si se entregan al servidor de mensajería, se considera "seguro" y de las manos del cliente.

**Almacenamiento y Envío:**

MOM proporciona colas de mensajes y la semántica de entrega garantizada, que aseguran que aplicaciones "no disponible " se obtienen los datos en cola y se entregan en un momento posterior.

La semántica de la entrega de mensajes cubre una gama de opciones de entrega, de exactamente una vez la entrega a por lo menos-de una sola entrega en más de una sola entrega. En el modo de entrega del exactamente una vez (también conocido como una vez y sólo una vez), las garantías del sistema de mensajería que el mensaje llegue al destino previsto, no importa qué, y nunca será enviado a más de una vez. Incluso en el pub / sub-modelo, donde varios receptores pueden consumir una copia de un mensaje transmitido, las normas siguen siendo válidas en el punto de vista relativo de cada consumidor. Exactamente una vez garantiza la entrega se lleva a cabo en parte por el uso de una técnica conocida como almacenar y enviar.

**Mensaje de Persistencia:**

Cuando los mensajes son marcados como persistentes, es responsabilidad del sistema de mensajería utilizar un mecanismo de almacenamiento y reenvío para cumplir su contrato con el remitente. El mecanismo de almacenamiento se utiliza para conservar los mensajes en el disco para asegurarse de que se puede recuperar si se produce algún fallo de cualquiera de los sistemas de mensajería del cliente consumidor. El mecanismo de transmisión es el responsable de la recuperación de mensajes del almacén, y posteriormente de enrutamiento y la entrega de ellos.

**Reconocimiento de mensajes:**

Mensaje de confirmación a nivel de protocolo de conexión es un factor clave en la mensajería garantizada. El protocolo de reconocimiento permite que el sistema de mensajería pueda supervisar el progreso de un mensaje y sepa si el mensaje fue producido con éxito y consumido. Con este conocimiento, el sistema de mensajería puede gestionar la distribución de mensajes y garantizar su entrega.

En el entorno de correspondencia imprecisa asincrónica de MOM, los remitentes y receptores (productores y consumidores) son intencionalmente desconectados unos de otros. Por lo tanto, las funciones y responsabilidades se reparten entre el productor del mensaje, el servidor de mensajes, y el consumidor de mensajes.

**Modelo de mensajería confiable**.

Un error común acerca de publicación y suscripción en comparación con colas de mensajería punto a punto es que publicar/ suscribir es para la mensajería más ligero de peso y no fiable y en cola de punto a punto más pesado y fiable. Eso puede haber sido cierto hace una década, cuando en realidad sólo estaban dos proveedores de mensajería: uno que soportaba publicar/subscribir ligero y otro que soportaba las colas punto a punto pesadas y fiables. Desde entonces, sin embargo, muchos proveedores de mensajería nuevos han aparecido en el mercado de apoyo a modelos de mensajería, cada uno con su propia gama completa de opciones de calidad de servicio. Además, estas técnicas fiables de mensajería puede ser utilizado a través de algunas topologías de implementación potencialmente sofisticados.

**Publicación/suscripción confiable:**

Un mensaje publicar/suscribir se puede enviar tan confiable como un mensaje de punto a punto. Un mensaje enviado en una cola de punto a punto puede ser entregado con una sobrecarga adicional si no se marca como persistente. Una confiable publicación/suscripción se entrega con una combinación de mensajes persistentes y suscripciones duraderas (un término común en el lenguaje JMS). Cuando una aplicación registra su interés en recibir mensajes sobre un tema particular, se puede especificar que la suscripción es durable. Una suscripción duradera sobrevivirá a fallas de la suscripción del cliente. Esto significa que si el receptor previsto de un mensaje no está disponible por cualquier razón, el servidor de mensajes continuará por almacenar los mensajes en nombre del receptor hasta que el

receptor vuelva a estar disponible. Cuando el receptor restablece su suscripción con el sistema de mensajería, los mensajes almacenados se entregarán.

El suscriptor se suscribe e indica que la suscripción es durable. El abonado se desconecta del servidor de mensajes, ya sea a través de un cierre correcto o a través de algún tipo de fallo. El editor envía el mensaje utilizando un método publish (). El método publish () se bloqueará y esperar hasta que reciba una confirmación del servidor de mensajes. El servidor de mensajes escribe el mensaje a un dispositivo confiable, el almacenamiento persistente. El mensaje se lleva a cabo de forma fiable en el disco. Se envía una confirmación al remitente indicando que el mensaje es seguro en las manos del servidor de mensajes. La publicación () devuelve el método. El suscriptor vuelve a conectar y restablece la suscripción. El mensaje se recupera del almacén persistente. El mensaje se envía al suscriptor. El suscriptor reconoce que el servidor de mensajes que ha recibido el mensaje correctamente. El servidor de mensajes elimina el mensaje del almacén persistente.

**Las colas fiables punto a punto:**

Para las colas de punto a punto, los mensajes están marcados por el productor, ya sea persistente o no persistente. Si son persistentes, son escritas en el disco y sujetas a las mismas normad de reconocimiento, condiciones de fallo, y recuperación como mensajes persistentes en el modelo publicación/suscripción.

Desde la perspectiva del receptor, las reglas son un tanto más sencillas que con las suscripciones duraderas porque el receptor no tiene ningún papel en el aspecto de la durabilidad. Con colas de punto a punto, porque sólo un consumidor puede recibir un caso particular de un mensaje, está en la cola, ya sea como un mensaje persistente o un mensaje no persistente. Un mensaje persistente permanece en la cola y en el disco hasta que se entrega a un consumidor o este caduca. Un mensaje no persistente también se mantiene en la cola hasta que se entrega o caduca, pero no está garantizado para sobrevivir a un fallo y recuperación del servidor de mensajería. La recepción del mensaje también está sujeto a las mismas normas de reconocimiento y la recuperación de condiciones de emergencia como en el modelo publicación/suscripción de una vez un mensaje persistente se entrega a la aplicación que se consume y se envía una confirmación de vuelta al servidor de mensajes, el mensaje puede se saquen del almacén persistente. Si algo falla durante ese proceso, el mensaje será entregado de nuevo durante la recuperación.

**Almacenamiento y envío a través de múltiples servidores:**

En un núcleo de ESB de MOM, el concepto de almacenamiento y de avance debe ser capaz de ser repetido a través de múltiples servidores de mensajes que se encadenan. En este escenario, cada servidor de mensajes utiliza almacenamiento y reenvío de mensajes y reconocimiento para hacer llegar el mensaje al siguiente servidor de la cadena.

Cada transferencia de servidor a servidor mantiene la fiabilidad y calidad de servicio mínimo que se han especificado por el remitente. Todo esto se hace de una manera que sea transparente tanto para el emisor y el receptor.

La cadena de servidores de mensajes no siempre se puede formar una línea recta. Dependiendo de la aplicación, sofisticado y dinámico de encaminamiento puede ser posible que dirige cada mensaje a lugares remotos a través de un camino diferente de servidores basados en la información de enrutamiento que está asociado con el destino.

**Transacción de mensajes.**

Tener todos los productores y todos los consumidores de los mensajes de participar en una transacción global acabaría con el propósito de un acoplamiento flexible en un entorno de mensajería asincrónica. En un ambiente de acoplamiento flexible, las aplicaciones necesitan la mano de sus mensajes al entorno de mensajería con operaciones locales, y dedicarse a sus negocios. Ellos no necesitan preocuparse acerca de si todas las partes interesadas están disponibles para participar simultáneamente en una transacción global de la pieza de datos que el mensaje representa.

**Operaciones locales:**

En el contexto de un remitente o un receptor individual, muchas implementaciones de MOM tienen un modelo para agrupar varias operaciones en una sola transacción que es de ámbito local a un remitente o receptor. Uno de estos casos es la agrupación de varios mensajes de una manera "todo o nada". Las operaciones siguen la convención de separar las de envío de las de recepción. Desde la perspectiva del emisor, los mensajes se llevan a cabo por el servidor de mensajes hasta que un comando commit se emite. Si se produce un error o si un comando de reversión se emite, los mensajes son descartados. Mensajes enviados al servidor de mensajes en una transacción no se transmiten a los consumidores hasta los productores confirma la transacción.

El servidor de mensajes no empezará a entregar los mensajes a sus consumidores hasta que se produzca una confirmación de comando commit en la sesión. Una transacción local basada en mensajes puede incluir cualquier número de mensajes.

También existe la noción de una transaccional recibir, en el que se recibió a un grupo de mensajes de transacción por parte del consumidor sobre la base de todo o nada (Figura 5-15). Desde la perspectiva del receptor, los mensajes se entregan con la mayor rapidez posible, pero se mantienen en un mecanismo de almacenamiento de reembolso por el servidor de mensajes hasta que se produzca una confirmación de comando commit. Si se produce un fallo o una orden emitida reversión (rollback), el sistema de mensajería intentará reenviar todos los mensajes.

Las transacciones locales también pueden ser utilizadas para agrupar las operaciones de recibir y enviar. En este caso, la aplicación cliente es a la vez consumidor y productor. Se puede recibir un mensaje, modificar su contenido, y enviar el mensaje nuevo junto a otro destino. Mediante el uso de la transacción local para agrupar de envío y recepción, una sola operación de confirmación se eliminará el mensaje de almacenamiento persistente del servidor y enviarlo a su próximo destino.

**Operaciones con múltiples recursos:**

A veces es necesario para coordinar el envío o recepción de una transacción local con la actualización de otro recurso transaccional, como una base de datos o una transacción de entidad EJB. Esto suele implicar un administrador de transacciones subyacente que se encarga de coordinar la preparación, confirmar o deshacer las operaciones de cada recurso que participan en la transacción. Una implementación de MOM puede proporcionar interfaces de transacción para lograr esto, lo que permite un productor de mensajes o al consumidor a participar en una transacción con cualquier otro recurso que es compatible con la XOpen / XA en dos fases-protocolo de confirmación de la transacción.

Técnicamente, se trata de una "distribución" de la transacción. Sin embargo, desde una perspectiva de MOM, se trata simplemente de vincular las interacciones servidor de mensajes con el procesamiento de un mensaje que implica el uso de otro recurso transaccional. Varios clientes de mensajes no participaría en la misma transacción global de forma distribuida, ya que esto acabaría con el propósito de un entorno de correspondencia imprecisa asincrónica. En un ESB, cada participante en un intercambio de mensajes tiene que ser capaz de confiar en que la integridad transaccional con su interacción con el autobús, y no con las otras aplicaciones que se conectan al sistema.

**Un ESB elimina las complejidades de bajo nivel:**

La forma habitual de utilizar una de MOM, ya sea basada en JMS, SOAP basados en, o algo más, es escribir código en una aplicación cliente que maneja las cosas como el establecimiento de una conexión con el servidor de mensajería, la creación de los editores, los suscriptores, la cola remitentes y receptores de la cola, y la gestión de la demarcación de transacciones y la recuperación del fracaso. Un ESB elimina esta complejidad, delegando esa responsabilidad al contenedor de servicios ESB.

Un arquitecto de integración puede utilizar una herramienta administrativa para configurar un servicio de ESB para enviar y recibir mensajes mediante el pub / sub, las colas de punto a punto, u otros tipos de mecanismos de transporte, simplemente mediante la configuración de las entradas y salidas del servicio. El contenedor se encarga del resto. Incluso es posible que un canal de salida para utilizar pub / sub, que es el tiempo asignado en una invocación de servicios web externos mediante SOAP. Más detalles sobre los contenedores de servicio y su uso de los canales de entrada y de salida se puede encontrar en el capítulo 6, capítulo 7, y en el capítulo 8.

**El Patrón de mensajería petición/respuesta.**

Comunicaciones a través de un ESB o MOM un son en gran medida la intención de ser asincrónica en la naturaleza. Aplicaciones y servicios de envío de mensajes en un "fuego y olvidar", lo que permite que una aplicación de ir sobre su negocio una vez que un mensaje es entregado de forma asincrónica. Esto no excluye necesariamente la necesidad de realizar peticiones y las operaciones de respuesta. A veces es necesario realizar una solicitud sincrónica / operación de respuesta, como cuando usted está tratando de integrarse con un

cliente de servicios web, que bloquea y espera una respuesta sincrónica regresar a él. A menudo el remitente (solicitante) puede esperar por la respuesta a suceder en un momento posterior (de forma asíncrona).

Petición / respuesta de mensajería patrones se pueden construir en la parte superior de un MOM para ejecutar una solicitud síncrono / asíncrono respuesta o la solicitud / respuesta. Un ESB además puede automatizar este proceso mediante la gestión de los detalles de la solicitud / respuesta en el modelo de invocación ESB contenedor. El modelo de solicitud / respuesta se introduce ahora, en el capítulo de mensajería, como nos referiremos a ella en otros debates en todo el resto del libro. En el capítulo 8, por ejemplo, vamos a ver cómo una solicitud sincrónica o servicio web, la respuesta se puede asignar a un ESB por el simple hecho de cómo los extremos se crean.

La clave de este patrón es el uso de una propiedad ReplyTo que se deja llevar con el mensaje de solicitud. El solicitante tiene que escuchar en una "respuesta" del canal que acepte el mensaje de respuesta. El mensaje de solicitud que se envía en el canal de solicitud debe contener el identificador del canal de respuesta, junto con un identificador de correlación que se utiliza para correlacionar el mensaje de solicitud con el mensaje de respuesta. Esto suele hacerse con una propiedad de mensaje, si el sistema de mensajería compatible con la noción de propiedades.

Una vez recibido el mensaje de solicitud, el replier extrae el identificador del canal de respuesta de la propiedad ReplyTo, y la utiliza para identificar el canal en el que para enviar la respuesta. También los extractos de la propiedad CorrelationID y lugares que en el mensaje de respuesta. En un entorno asincrónico, varias solicitudes y las respuestas pueden ocurrir simultáneamente. El solicitante utiliza el identificador de correlación para que coincida con la respuesta a la solicitud correspondiente.

**El patrón reenviar respuesta:**

A veces, la respuesta no tiene por qué ser devueltos a la empresa misma que inició la petición (Figura 5-19). Un ESB utiliza una solicitud más elaboradas / modelo de respuesta en la que una respuesta de la invocación de un servicio puede ser otro destino, o una dirección de envío de dónde enviar el siguiente mensaje. Esto se conoce como el patrón de respuesta hacia adelante.

**Estándares de mensajería.**

Una serie de normas que dejan de evolucionar en el área de MOM. Una norma establecida es el Java Message Service, y otros que se están desarrollando son los basados en SOAP y servicios web.

**El servicio de mensaje Java (JMS):**

El Java Message Service (JMS) es una especificación de mensajería que ha contado con la adopción generalizada de la industria desde su introducción en 1998. JMS proporciona un API

y un conjunto de normas que rigen la semántica de entrega de mensajes en un entorno de MOM para mensajería confiable y no confiable. La especificación JMS define las reglas para el comportamiento operacional de publicación y suscripción y gestión de colas de punto a punto. También hay definiciones muy rico y flexible de lo que un mensaje se compone de, y las normas estrictas que rigen almacenar y reenviar mensajes, entrega garantizada, acuses de recibo de mensajes, transacciones, y la recuperación de los fracasos. JMS también proporciona el modelo de transacción fines múltiples descritos en este capítulo. La ventaja de haber especificado las normas de comportamiento es que usted puede confiar en un conjunto básico de comportamiento en el entorno de mensajería, independientemente de la aplicación de proveedores.

Si bien gran parte pretende ser una forma asincrónica de comunicación entre aplicaciones, JMS también tiene un conjunto de patrones de mensajería y las interfaces de ayuda para sincrónica y asincrónica apoyar el modelo de solicitud / respuesta utilizando el pub / sub y modelos de mensajería punto a punto.

JMS proporciona una interfaz opcional para la integración con un servidor de aplicaciones. Esto permite que un proveedor de JMS de un proveedor para integrarse con un servidor de aplicaciones de otro proveedor. Un proveedor de ESB que soporta una interfaz JMS debería apoyar el servidor de aplicaciones interfaces también.

**Que pasa con la "J" en JMS?**

JMS definitivamente tiene sus raíces en Java. Sin embargo, usted no tiene que ser un programador de Java a utilizar un sistema de mensajería JMS compatible si está utilizando un ESB.

Mientras que en teoría podría implementar un ESB con una mamá que no es compatible con JMS, que es una idea mejor no, por varias razones. El uso de un ESB que soporta JMS semántica de la entrega de mensajes significa que usted puede contar con ciertas reglas de comportamiento. Independientemente de la aplicación de proveedor, usted puede confiar en la existencia de capacidades tales como suscripciones duraderas para mensajería pub / sub, y la recuperación de transacciones de la falta de uno u otro modelo de mensajería.

Dicho esto, incluso si no son particularmente apegado a Java, puede tomar ventaja de JMS si está utilizando un ESB. Una aplicación que soporta JMS ESB también pueden proporcionar tipos de clientes no-Java para C + +, VB o C #. JMS define una API que puede ser igualmente implementado en Java, C + + o C # si usted está codificando los clientes de mensajería telefónica. Si está diseñando una estrategia de integración ESB que sólo requiere enchufar juntos adaptadores de aplicaciones y servicios de integración sin ningún tipo de cliente específico de codificación, JMS puede agregar valor al proporcionar un conjunto coherente de normas de comportamiento para el sistema de mensajería subyacente, sin importar el proveedor que ofrezca la aplicación.

**Mensajería de confianza con SOAP.**

Un enfoque para lograr la confiabilidad asincrónica entre los servicios web, es a través de un protocolo fiable a nivel SOAP. En este enfoque, reconocimientos y recibos de entrega de mensajes se codifican en el encabezado predefinido sobre SOAP construcciones. Ejemplos de esto que se están desarrollando hoy en día son WS-ReliableMessaging y WS en Confiabilidad. ESB son muy adecuadas para la adopción de estos protocolos a medida que maduran y son ejecutadas por más vendedores. Un ESB debe ser capaz de soportar cualquier número de transportes de mensajería confiable y protocolos.

**Eventos y Notificaciones servicios web:**

Dos especificaciones naciente, WS-Eventing y WS-notificación, se describe cómo publicar y suscripción deben trabajar a través de servicios web, interfaces y protocolos. Un comité técnico ha sido formado bajo OASIS bajo el auspicio de WS-Notificación a formalizar una familia de especificaciones que detalla una publicación y suscripción del modelo de notificación de eventos mediante los servicios Web y SOAP.

**Invocaciones de servicios ESB, ruteo y SOA**

En este capítulo vamos a aprender acerca del modelo de invocación de servicios, es decir, el marco básico que proporciona la SOA en un ESB. Vamos a discutir las múltiples formas de proceso de enrutamiento, incluyendo el concepto de itinerario y el enrutamiento basado en el papel que desempeña en permitir una SOA altamente distribuidos a través de servicios de manera independiente desplegado.

También vamos a discutir algunos servicios fundamentales ESB, tales como enrutamiento basado en contenido y la transformación XSLT, y examinar cómo los tipos de servicio se puede definir y luego volver a utilizar para diferentes propósitos con opciones configurables que se especifican de forma declarativa y no codificado.

A continuación exploraremos algunos servicios avanzados de ESB, como un servicio de persistencia XML y un servicio de orquestación.

**Buscar, enlazar e Invocar.**

Uno de los lugares donde la ESB añade valor incremental en un SOA típico es en el descubrimiento / bind / invocar las operaciones que normalmente se producen. Hasta la fecha, SOA han sido típicamente usando un modelo cliente-servidor. En una arquitectura SOA que sigue el modelo cliente-servidor, ya sea a través de servicios web o un predecesor, los clientes de servicios en contacto con otros servicios a través de un proceso conocido como encontrar, se unen, e invocar. El hallazgo / bind / invocar modelo asume que existe un registro o un directorio que almacena la ubicación y, posiblemente, otros metadatos de una aplicación de servicio. En el "buscar" la operación, el servicio al cliente realiza una búsqueda en el registro para el servicio, usando ciertos criterios que pueden incluir los nombres o características tales

como "color de doble cara." El siguiente paso es la operación "bind", que para una solicitud de servicios Web podría significar simplemente haciendo una conexión HTTP y, por último, el "invocar" la operación, lo que significa el envío de un mensaje o invocar un procedimiento remoto. Con un enfoque de servicios web a SOA, se unen y se invocan en la misma operación.

El problema con el hallazgo / bind / invocar el método es que requiere escribir la lógica de enrutamiento y el flujo en las aplicaciones que necesitan ser conectados. Cada cliente de servicios tiene que escribir código que realiza una búsqueda del servicio, vimos la desventaja de que en el capítulo 2. Aunque sin duda es posible hacer esto, e incluso es apropiada en ciertas circunstancias en un entorno ESB, que no está destinada a ser la norma. El centro de diseño de base de la ESB es un servicio que no se invoca directamente por un servicio más, sino más bien forma parte de un flujo de proceso más amplio orientado a eventos.

**Invocación de servicio en ESB.**

Al igual que su predecesor, el cliente-servidor SOA, ESB tiene el concepto de un registro o servicio de directorio en el que se almacena información acerca de los extremos de servicio. Un inherentes a encontrar / bind / invocar la operación se produce como parte de la mecánica de ESB, pero se separa de la lógica empresarial. En un entorno asincrónico ESB, el hallazgo / bind / invocar el conjunto de operaciones puede realmente mapa para el envío de un mensaje XML a una cola o una publicación y suscripción destino tema, y el procesamiento de la respuesta. En la discusión de los contenedores de servicios en el capítulo 6, vimos que la implementación de un servicio, simplemente trata de una entrada / salida metáfora punto final. El código de servicio se centra únicamente en la lógica de la aplicación de un servicio. Mensajes XML son recibidas por el servicio desde un extremo de entrada que es gestionado por el contenedor de servicios. Al término de su misión, la implementación del servicio simplemente coloca su mensaje de salida en el extremo de salida que se llevarán a su próximo destino. La salida del servicio es la respuesta, que las rutas de ESB para el siguiente paso (usando el patrón de respuesta hacia adelante), o en la espalda al solicitante (utilizando el modelo de solicitud / respuesta). El mensaje de salida puede ser el mismo mensaje que se recibió, el servicio puede aumentar o modificar partes del mensaje, o crear una nueva "respuesta" del mensaje. La operación de identificar y localizar el siguiente servicio de la cadena, la invocación de la unión a él, y el de ella es un conjunto de tareas llevadas a cabo por el propio ESB. El medio por el cual el hallazgo / bind / invocar las operaciones se definen no es a través del código escrito, sino a través de herramientas de configuración y despliegue.

**Itinerario del enrutamiento basado en: SOA altamente distribuidas.**

La importancia clave del enfoque de ESB a SOA es que la definición de servicio está separada del mecanismo para la localización y la invocación de servicios. El papel del arquitecto de integración es definir administrativamente un flujo de procesos de negocio compuesta por conectar los servicios juntos en un itinerario mensaje. El recorrido representa un conjunto de operaciones de enrutamiento de mensajes discretos, como los pasos básicos introducidos en el capítulo 4 (ver Figura 7-1).

Mensaje itinerarios son la clave para permitir una SOA altamente distribuidos a través de un

ESB. Los detalles del itinerario se almacenan como metadatos XML y llevaba el mensaje a medida que viaja a través del bus de servicio de un recipiente a otro. Un itinerario puede comenzar en cualquier punto de entrada o evento que puede suceder en el autobús, incluyendo la creación y publicación de un mensaje, iniciado por cualquier servicio que sea accesible en el autobús. Un itinerario puede incluso ser adjunto a un mensaje entrante que llega al dominio de la ESB. Esto podría ser un evento externo, como la recepción de un mensaje SOAP de un compañero de trabajo, externos.

Los pasos lógicos de un itinerario puede representar los extremos de servicio en una SOA que están físicamente repartidos por zonas geográficas y accesible desde cualquier lugar en el autobús, como se ilustra en la Figura 7-2.

Gartner ha utilizado el término "microflujo" para indicar una corta vida, el segmento proceso transitorio. El proceso de itinerario ESB es muy adecuado para microflujos tales. itinerarios de mensajes son muy útiles para un conjunto discreto de operaciones en las que decisiones simples de ramificación se puede hacer, y donde la separación de tiempo entre invocaciones de servicio no es un factor importante. Por simple de ramificación, un itinerario también puede apoyar la noción de un subproceso. Figura 7-3 muestra el flujo de control mediante el cual un proceso padre puede ser suspendido temporalmente para invocar un subproceso, y luego se reanuda cuando vuelve el subproceso.

Un itinerario mensaje contiene metadatos que describen la forma de dirigir el mensaje, incluyendo una lista de direcciones de reenvío se describe como criterios de valoración abstracta o como normas para evaluar a lo largo del camino. El contenedor de ESB es un "inteligente" de contenedores, lo que significa que sabe cómo evaluar el itinerario de un mensaje basado en los metadatos que se deja llevar con el mensaje, combinado con el conocimiento de configuración que se almacena localmente en cada contenedor. Esto hace que para una red altamente distribuida de enrutamiento que no se basa en un motor de reglas centralizado.

Un itinerario mensaje es análogo a un itinerario de viaje que usted lleva cuando va de viaje. Su itinerario de viaje le dice a donde a proceder en cada etapa de un viaje: la información sobre vuelos, alquiler de coches del hotel, etc Imagina que, en lugar de realizar un itinerario, que se detenía y llame a su agente de viajes en cada paso del camino: "OK, me voy el avión ... ¿cuál es mi agencia de alquiler de coches? ... Está bien, tengo el coche ... qué hotel estoy estancia en?" Eso sería el equivalente de un motor de enrutamiento centralizado hub-and-spoke.

El hecho de que no existe una única "agencia de viajes" para referirse de nuevo a es un diferenciador clave que ayuda a que el enfoque de ESB altamente distribuidos. Esto significa que diferentes partes de la red de ESB pueden operar de forma independiente el uno del otro sin depender de cualquier motor de enrutamiento centralizado que podría ser un punto único de fallo. Las capacidades de gestión que son inherentes a un ESB que sea fácil de administrar de forma remota los contenedores distribuidos para empujar a cabo cambios en las configuraciones de itinerario. Dado que el mensaje lleva el estado itinerario y el proceso con él, sólo los mensajes nuevos en el proceso del negocio recibirán las instrucciones de nueva configuración. Los mensajes que ya están en proceso no se verán afectados.

# Anexo D

# Modelado del Servicio - Línea Base Arquitectónica

En este anexo se muestra el análisis de la línea base arquitectónica teniendo en cuenta las vistas de funcionalidades, de referencia, de implementación, de distribución física de elementos y componentes modulares, exponiendo y describiendo los diferentes modelos (de casos de uso, de análisis, de diseño, de implementación y de despliegue) que componen la implementación del servicio de Tono de Timbre Personalizado (CRBT, Color Ring Back Tone) y el servicio de personalización del CRBT del operador EMCALI.

## D.1 Características del servicio.

El desarrollo del piloto consiste en la implementación del servicio CRBT en conjunto con la personalización del mismo por parte del cliente desde un ambiente web, en la cual se integra la plataforma de telecomunicaciones desde la cual se ejecuta el servicio CRBT y la plataforma TI que permite la personalización y el cobro del mismo.

A continuación se mencionan las características más importantes con las cuales cuenta el prototipo CRBT:

**Funcionales:**

*Reproduce RBT*: creados a partir de componentes de audio, reemplazando el RBT tradicional, el cual se reproduce para cualquier llamada entrante del mismo.
*Personaliza CBRT*: desde un ambiente web se escoge la componente de audio deseado.
Permite al usuario la personalización del servicio gracias a la integración Telco – TI.

**No funcionales:**

Permite la suscripción a los clientes que pertenecen a la NGN de EMCALI. Es independiente del dispositivo final (teléfono convencional, móvil, softphone, SIP) y de la red (NGN, PSTN, GSM). Es decir, el RBT se reproduce hacia cualquier terminal de usuario que tenga conexión con la NGN de EMCALI (**convergencia a nivel de terminal y de red**).

Permite al administrador de Rhino SLEE suscribir nuevos usuarios.

De acuerdo al análisis de las características generales del servicio CRBT y las condiciones de red del operador EMCALI (ver Anexo B) se presenta la línea base arquitectónica del sistema solución.

# D. 2 Línea base arquitectónica.

### D.2.1 Vista de funcionalidades

#### D.2.1.1. Modelo de caso de uso

**Servicio CRBT.**

La Figura 26 y Tabla 5, describe las funcionalidades y el escenario del servicio CRBT el cual es ejecutado en el ambiente de telecomunicaciones.

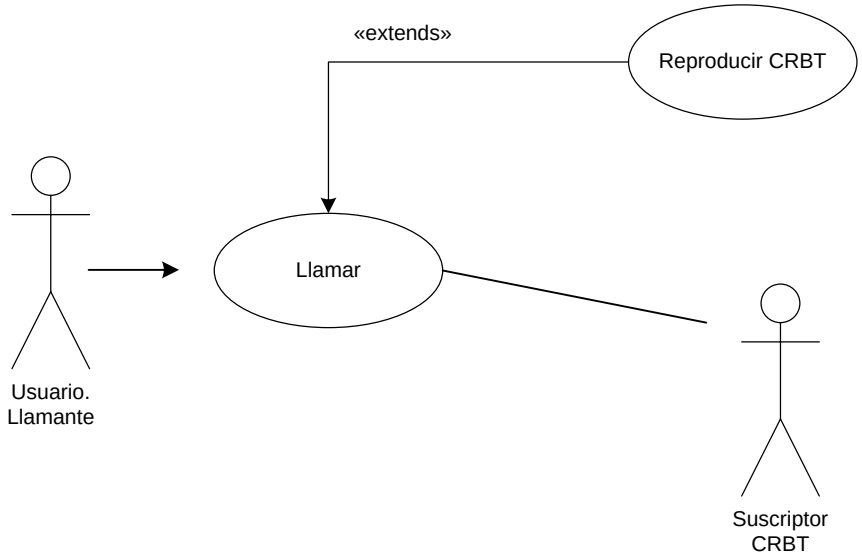



Figura 26 Diagrama de casos de uso del servicio CRBT

| **Caso de Uso** | **Llamar** |
|---|---|
| **Actores** | Usuarios realiza llamada (iniciador).<br>Usuario A. |
| **Importancia** | Alta |
| **Propósito** | Reproducir al actor Usuario A un componente de audio, previamente definido, que reemplaza el RBT tradicional. |
| **Resumen** | Cuando el Usuario A inicia una solicitud de llamada, la entidad llamar a través de la entidad Reproducir CRBT verifica si el Usuario B tiene el servicio CRBT suscrito, en caso afirmativo el Usuario A escucha el audio personalizado de lo contrario el Usuario A escuchara el RBT tradicional. |
| **Precondiciones** | |

| |
|---|
| El Usuario B deber estar suscrito al servicio y tener el archivo audio de su preferencia |
| **Escenario** |

| **Actores** | **Sistema** |
|---|---|
| Envía solicitud de inicio de llamada | Recibe la solicitud de inicio de llamada<br>Verifica que el usuario llamado tenga activo el servicio CRBT (E1)<br>Carga el perfil de usuario suscriptor<br>Retorna solicitud de envío de llamada |
| Recibe solicitud de inicio llamada<br>Envía solicitud de inicio de llamada a usuario B<br>Recibe estado disponible de usuario llamado<br>Envía estado disponible de usuario llamado (usuario B) | Recibe estado disponible de usuario llamado<br>Envía archivo de audio al usuario llamante |

| |
|---|
| **Postcondiciones** |
| Llamada establecida. |
| **Excepciones** |
| E1: Servicio inactivo. No se carga el perfil y el proceso de establecimiento de llamada continua normalmente. |

Tabla 5 Descripción del escenario de caso de uso Enviar Audio

**Servicio Personalización CRBT.**

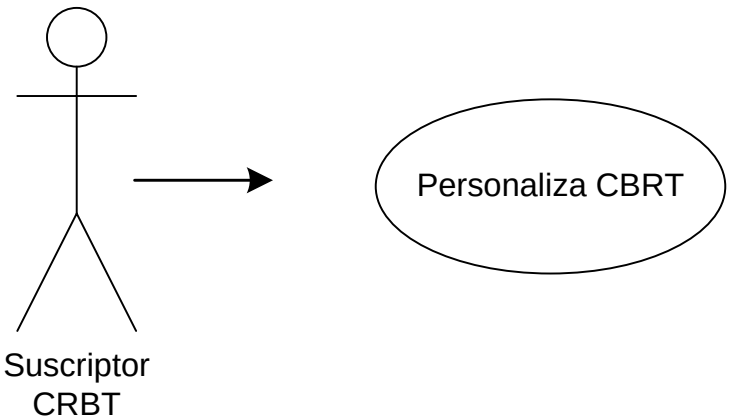



Figura 27 Diagrama de casos de uso del servicio Personalización CRBT

| **Caso de Uso** | **Personaliza CRBT** |
|---|---|
| **Actores** | Suscriptor envía solicitud de personalización (iniciador).<br>Suscriptor del servicio. (Usuario B) |
| **Importancia** | Alta |
| **Propósito** | Actualizar el nombre del audio seleccionado por el usuario en la tabla de perfiles del sistema Rhino |
| **Resumen** | Cuando el suscriptor del servicio selecciona el nombre del audio preferido un mensaje va hacia el servidor ESB, quien a su vez realiza dos tares, la primera: enviar un mensaje al servidor Rhino que actualiza el perfil en la tabla de perfiles; segundo: un mensaje al BSS para generar el cargo del respectivo cobro por el cambio de la canción. |
| **Precondiciones** | |
| El Usuario B deber estar suscrito al servicio. | |
| **Escenario** | |

| **Actores** | **Sistema** |
|---|---|
| Escoge el nombre de audio deseado y activa la opción de actualización (usuario B)<br>Envía solicitud de actualización de perfil | Recibe la solicitud de actualización de perfil.<br>Verifica que el usuario tenga el servicio CRBT suscrito.<br>Actualiza el perfil en la tabla de perfiles.<br>Genera el cobro por el servicio en el sistema comercial.<br>Retorna mensaje de perfil actualizado. |

| **Postcondiciones** |
|---|
| Perfil Actualizado. |
| **Excepciones** |
| E1: Servicio inactivo. No se actualiza el perfil, se notifica de la inexistencia del servicio suscrito para el usuario B. |

Tabla 6 Descripción del escenario de caso de uso Personalizar CRBT

### D.2.2 Vista del modelo de referencia

#### D.2.2.1 Modelo de diseño

**Servicio CRBT**

Con base en el modelo de análisis se ha desarrollado un modelo final de diseño a través de las responsabilidades de las clases y el funcionamiento del sistema. La Figura 28 y Figura 29, muestran la estructura y comportamiento final del servicio implementado.

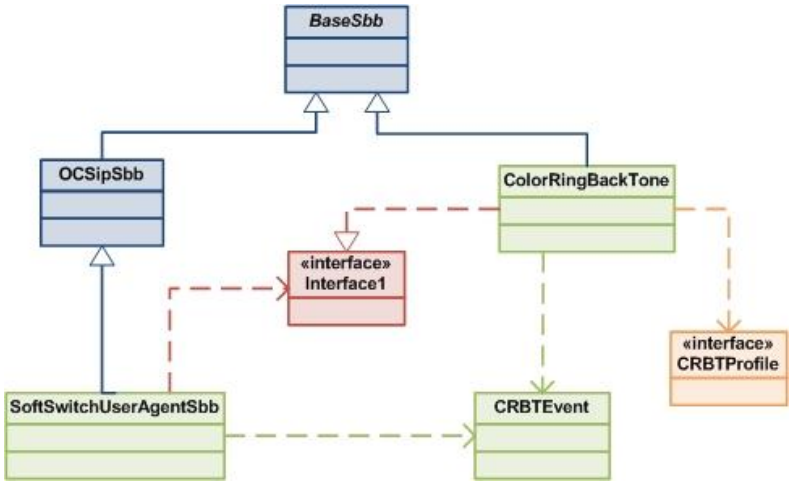



Figura 28 Diagrama de clases de diseño del prototipo CRBT

La Figura 28 muestra la estructura general del servicio CRBT, a continuación la descripción de cada uno de los componentes.

La clase BaseSbb es una clase re utilizada con el fin de implementar métodos necesarios y comunes para el funcionamiento del servicio y contiene métodos típicos en la construcción de elementos SBB.

La clase OCSipSbb es una clase re utilizada con el fin de implementar métodos necesarios y comunes para el funcionamiento del servicio y contiene lógica orientada al manejo de mensajes SIP.

SoftSwitchUserAgentSbb: esta clase basada en el código fuente del BackToBackUserAgentSbb de Opencloud, se ajusta para los requerimientos particulares del servicio CRBT dentro de la NGN de EMCALI. Su función es el control de los mensajes SIP generados durante todo el establecimiento y finalización de la llamada para el servicio CRBT. Con el fin de mantener el estado de los Bloques de Construcción del Servicio (SBB,Service Building Block), esta clase usa los campos de Persistencia Gestionada por Contenedor (CMP, Container Managed Persistence). A diferencia de la programación normal en Java, este nuevo paradigma basado en eventos, requiere del manejo de estructuras diferentes. Esta es la razón de no usar parámetros dentro de la clase si no los campos CMP.

El control de los mensajes SIP se realiza sobre la manipulación de eventos dentro de la clase. En este punto, se manejan los eventos de tipo Request y de tipo Responses. Los primeros son todos los mensajes SIP como INVITE, CANCEL, BYE y ACK. Los segundos, todos aquellos con respuesta 1xx y 2xx, como mensajes OK (código 200), Ringing (codigo180) y Session Progress (código 183).

ColorRingBackToneSbb: esta clase se encarga del control del servicio relacionado con la lógica y la comunicación con el servidor de medios. En este punto se majean los mensajes MGCP como el CRCX RESPONSE que realiza la interacción con el MMS. Estos mensajes son atendidos por los eventos respectivos guardando su estado en los campos CMP.

CRBTSbbLocalObject: interfaz que contiene los métodos que la clase SoftSwitchUserAgentSbb usa para la comunicación con la lógica del servicio. Los métodos descritos por esta interfaz son:createCRBTConnection, playCRBTMedia, isCRBTSubscriber, deleterCRBTConnection y cancelCRBTProcess.

CRBTProfileCMP: interfaz para acceder a los campos de perfil. Esta clase contiene los métodos necesarios para realizar la lectura del archivo media relacionado al suscriptor.

CRBTEvent: clase encargada de comunicar todos los eventos producidos dentro de las transacciones del servicio. Este punto realiza el control del flujo de mensajes necesarios entre las clases que manejan la señalización SIP y los mensajes MGCP.

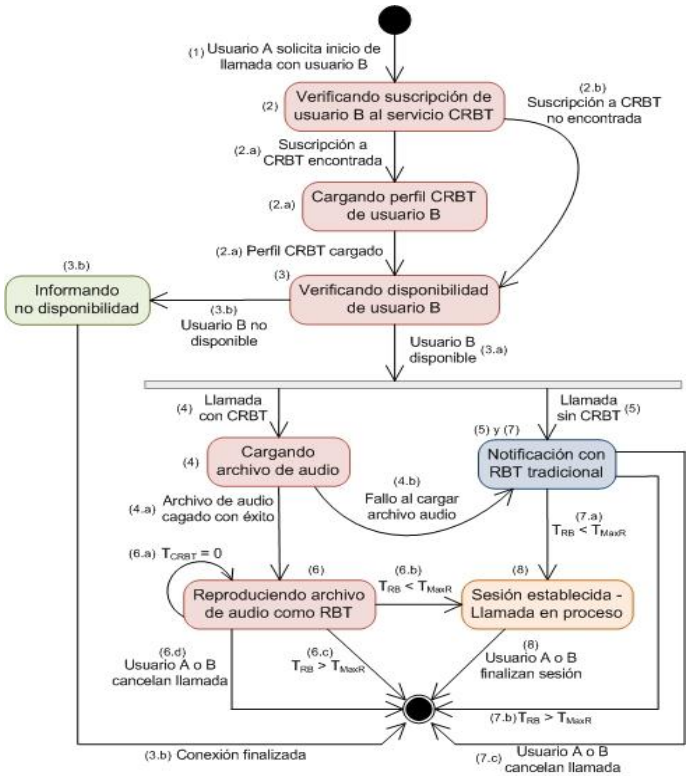



Figura 29 Diagrama de estados del caso de uso Reproducir RBT de audio

El anterior diagrama (Figura 29) representa el prototipo CRBT implementado. A continuación se exponen y describen los distintos estados, eventos y transiciones, que ocurren en el caso de uso Reproducir RBT de audio en la NGN de EMCALI

El flujo del prototipo inicia cuando un usuario A solicita establecer una sesión de llamada con un usuario B.
Cuando el sistema verifica que el usuario B tenga suscrito el servicio CRBT:
Si la suscripción del usuario B al servicio CRBT es encontrada, el sistema carga su perfil CRBT y procede a verificar la disponibilidad del usuario B.
Si la suscripción del usuario B al servicio CRBT no es encontrada, el sistema procede directamente a verificar la disponibilidad del usuario B.
Cuando el sistema verifica la disponibilidad del usuario B:
Si el usuario B se encuentra disponible, el sistema procede a iniciar la sesión de llamada hacia este usuario: con CRBT, si la suscripción al servicio CRBT fue encontrada; sin CRBT, si tal suscripción no fue encontrada.
Si el usuario B no se encuentra disponible (ocupado, no accesible o no conectado), el sistema informa al usuario A dicho estado y finaliza la conexión.
Para una llamada con CRBT, el sistema carga el archivo de audio, previamente definido en el perfil CRBT del usuario B, el cual fue obtenido en el punto 2-a.
Si el archivo de audio es cargado exitosamente, el sistema procede a reproducir el componente de audio como RBT de la llamada.
Si el archivo no es encontrado o existe un error en la carga del mismo, el sistema procede con la llamada sin CRBT.
Para una llamada sin CRBT, el sistema emplea el RBT tradicional para notificar al usuario A acerca del proceso de la llamada.
Cuando el sistema reproduce el archivo de audio como RBT al usuario A:
Si el tiempo de duración del archivo de audio ($T_{CRBT}$) llega a su fin ($T_{CRBT} = 0$) y el usuario B aún no responde la llamada, el archivo de audio se reproduce de nuevo.
Si el tiempo de respuesta del usuario B ($T_{RB}$) es menor al tiempo máximo de espera para la respuesta de la llamada ($T_{MaxR}$)[26] ($T_{RB} < T_{MaxR}$), el sistema detiene la reproducción del RBT de audio y establece la llamada, permitiendo el intercambio de voz entre los usuarios.
Si $T_{RB} > T_{MaxR}$, el sistema detiene la reproducción del RBT de audio y finaliza la conexión.
Si el usuario A o B cancelan la llamada, el sistema detiene la reproducción del RBT de audio y finaliza la conexión.
Cuando el sistema notifica con el RBT tradicional al usuario A:
Si $T_{RB} < T_{MaxR}$, el sistema establece la llamada.
Si $T_{RB} > T_{MaxR}$, el sistema finaliza la conexión.
Si el usuario A o B cancelan la llamada, el sistema finaliza la conexión.
Una vez la llamada se establece y se encuentra en proceso, si el usuario A o B terminan la sesión, el sistema finaliza la conexión.

[26] Para el caso de EMCALI, el $T_{MaxR}$ es de 1 minuto, valor definido por el responsable del Softswitch de esta empresa.

**Servicio personalización CRBT**

Con base en el modelo de análisis se desarrolló un modelo final de diseño a través de las responsabilidades de las clases y el funcionamiento del sistema. La Figura 30 y la Figura **31** muestran la estructura y comportamiento final del servicio implementado.

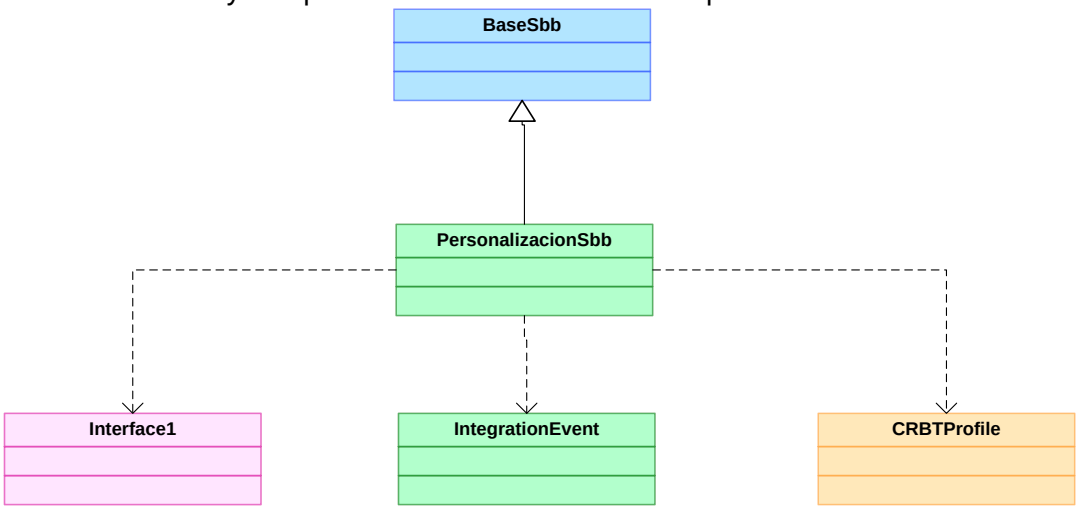



Figura 30 Diagrama de clases de diseño del prototipo CRBT

La Figura **31** muestra la estructura general del servicio de personalización CRBT, a continuación la descripción de cada uno de los componentes.

BaseSbb es una clase re utilizada con el fin de implementar métodos necesarios y comunes para el funcionamiento del servicio y contiene métodos típicos en la construcción de elementos SBB.

PersonalizacionCRBT es una clase basada en el código fuente del FromJ2EESbb de OpenCloud. Su función es recibir el dato proveniente del servidor ESB a tráves del evento IntegrationEvent y actualizar la tabla de perfiles.

IntegrationEvent: clase encargada de comunicar todos los eventos recibidos del ESB.

El Figura **31** representa el prototipo del servicio PersonalizaciónCRBT implementado. A continuación se describe el proceso de comercialización del servicio CRBT:

(1) El usuario B, cuenta con el servicio de telefonía y desea obtener el servicio CRBT.
(2) El usuario B ingresa a la página de la empresa outsourcing contratada por el operador de telecomunicaciones.
(3) El usuario B obtiene un listado de las canciones disponibles.
(4) El usuario B escoge la canción de su preferencia, revisa las condiciones comerciales y si está de acuerdo acepta la transacción.
(5) Un mensaje con los datos de teléfono, fecha de transacción, nombre de archivo media (.wav) es enviado al ESB.
(6) El ESB realiza dos tareas:

a. Envía el número del teléfono y nombre del archivo media (.wav) al servidor SLEE, en el cual un servicio de personalización está corriendo; el SLEE crea el registro en la tabla de perfiles definida para el funcionamiento del servicio CRBT.
b. El ESB envía el número del teléfono, nombre de archivo y fecha de compra al sistema BSS/OSS de la empresa de telecomunicaciones, la cual a su vez genera un cargo que será facturado.

(7) El sistema outsourcing almacena los datos de la transacción que utilizará para el cobro de la comisión de venta a la empresa de telecomunicaciones.

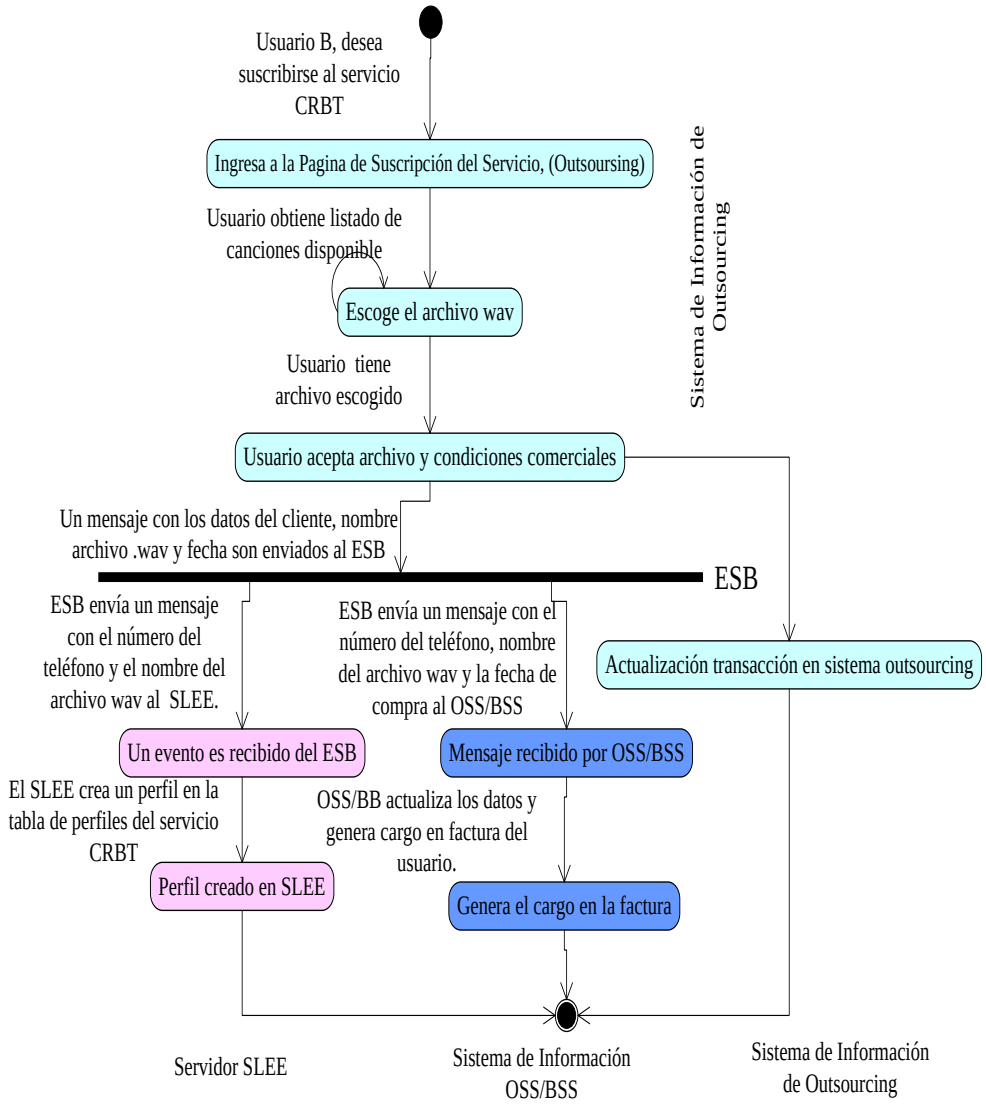



Figura 31 Diagrama de estados del servicio Personalizar CRBT

**Modelo de implementación**

**Servicio CRBT y Servicio Personalización CRBT.**

Este modelo representa los módulos funcionales en la prestación del servicio CRBT dentro de la NGN de EMCALI, la Figura 32 muestra los diferentes bloques que componen el VAS construido y los elementos lógicos que dan soporte al servicio.

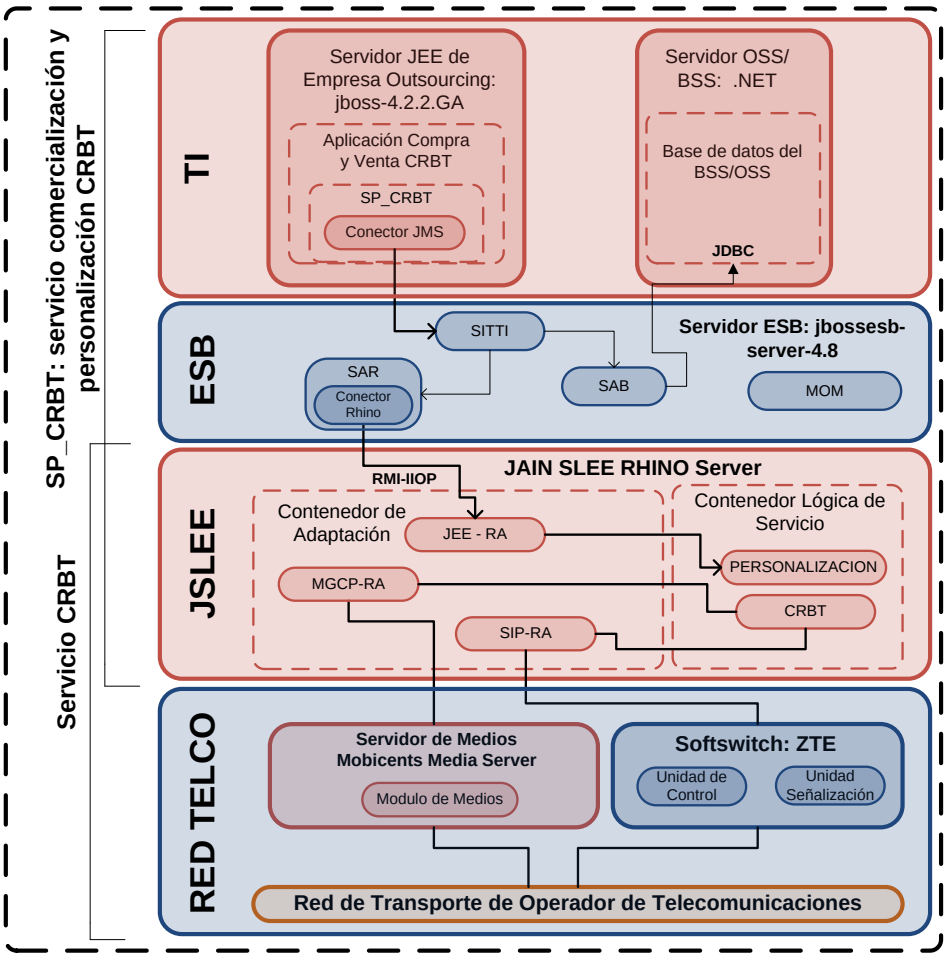



Figura 32 Arquitectura modular del sistema solución

Servidor JAIN SLEE (RHINO SLEE): este componente contiene módulos funcionales denominados unidades desplegables. En ellas se encuentran elementos necesarios para la prestación del servicio CRBT. La primera unidad incluye los bloques que ejecutan la lógica de la aplicación. Estos bloques estos construidos por eventos, perfiles y los SBB. Las otras dos unidades la componen los bloques encargados de realizar la comunicación con el módulo de control de la NGN de EMCALI y el módulo de medios (SIP RA y MGCP RA)

Servidor de Medios (MMS): este componente modular es el encargado de contener los archivos de audio que son reproducidos al usuario llamante. Dentro del él se destaca el bloque de controlador MGCP encargado de implementar la interfaz del estándar para este protocolo. El bloque EndPoint es una representación lógica de una entidad física como un enlace troncal o una fuente de audio. Para el prototipo se seleccionó el “custom endopoint” el cual, a criterio

del desarrollador, es posible crear una propia representación lógica del recurso físco (archivo de audio)

Unidad de control y señalización: este componente es el punto central de la arquitectura NGN de EMCALI, encargado del flujo de información y control desde los usuarios hasta las capas superiores (Servidores de aplicaciones) y viceversa. Está compuesto de un nivel de servicio, un nivel de control y un nivel de adaptación (ver Anexo C).

Sistemas de información empresarial (EIS, Enterprise Information System): este modulo lo conforman todos aquellos recursos de carácter empresarial que manejan la información del negocio como por ejemplo, los recursos de planeación empresarial, el procesamiento de transacciones, bases de datos, etc. En este caso particular la base de datos implementada es PostgreSQL.

### D.2.3 Vista de Implementación

#### Modelo de implementación

A continuación se describe los componentes ejecutables finales del sistema para el funcionamiento del servicio CRBT en la NGN de EMCALI.

| Componente | Descripción |
|---|---|
| Servicio.jar | Componente ejecutable de clases que realizan la lógica del servicio |
| SipRA.jar | Componente ejecutable que maneja la interconexión con el nodo control de la NGN de EMCALI y los mensajes SIP al interior del servicio |
| MGCP.jar | Componente ejecutable encargado de realizar la comunicación con servidor de medios por medio de los mensajes MGCP |
| DAS | Elemento encargado de llevar a cabo el análisis de dígitos de un número entrante a la NGN de EMCALI. En caso de que el usuario llamado sea suscriptor del servicio CRBT, la solicitud de inicio de sesión será direccionada al servidor RHINO. En este punto se puede configurar un número o una serie de números los cuales serán enrutados al servidor JSLEE. |
| Profile.sql | Componente que se ejecuta en la base de datos. Permite el acceso al perfil del suscriptor. |
| Crbt-beans.xml | Archivo ejecutable que contiene la configuración del número de endpoint o conexiones simultaneas permitidas dentro del MMS |
| *.wav | Componente que contiene todos los archivos de audio a ser reproducidos. |
| J2eedu.jar | Componente ejecutable que maneja la integración entre el servidor ENS y Rhino. |
| Personalizaciondu.jar | Componente ejecutable que permite actualizar el perfil en la tabla de perfiles. |
| CRBTdu.jar | Componente ejecutable encargado de manejar la lógica del servicio |

| | |
|---|---|
| | CRBT. |
| Sleeconnector.jar | Componente encargado de la integración ESB y Rhino-SLEE |
| Serviciosesb.esb | Componente que contiene la lógica del servicio de integración , enviando mensajes al BSS y Rhino-SLEE. |
| JDBCConnector | Encargado de la conexión con la base de datos. |
| WebAPP | Aplicación web, la cual le permite al usuario suscriptor del servicio realizar la actualización del perfil. |

Tabla 7 Descripción de componentes del sistema solución

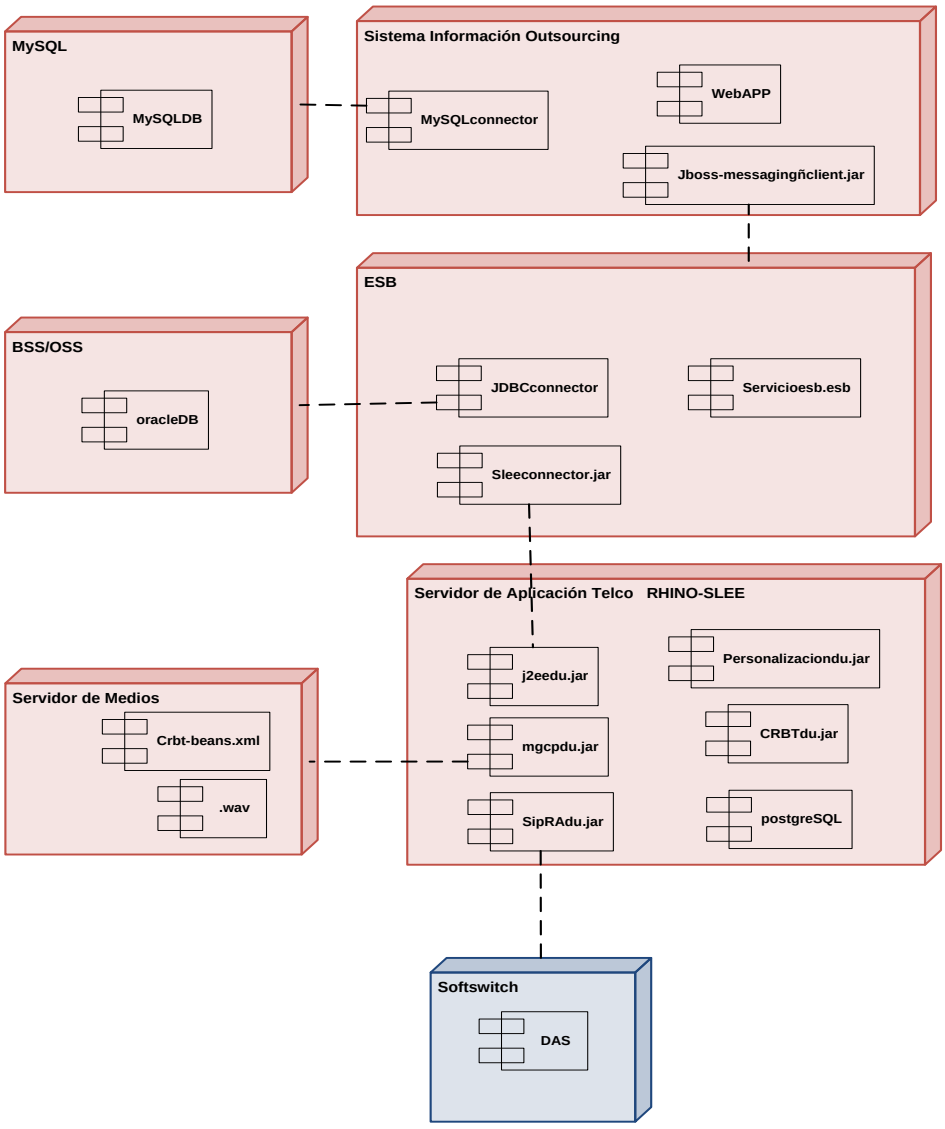



Figura 33 Diagrama de componentes del sistema solución

### D.2.4 Vista de distribución física de elementos

## Modelo de despliegue

Dentro de este modelo, se presenta el diagrama de despliegue, Figura 34, de los componentes físicos que representan la interacción con la NGN de EMCALI, los elementos que prestan el servicio CRBT y la integración con TI.

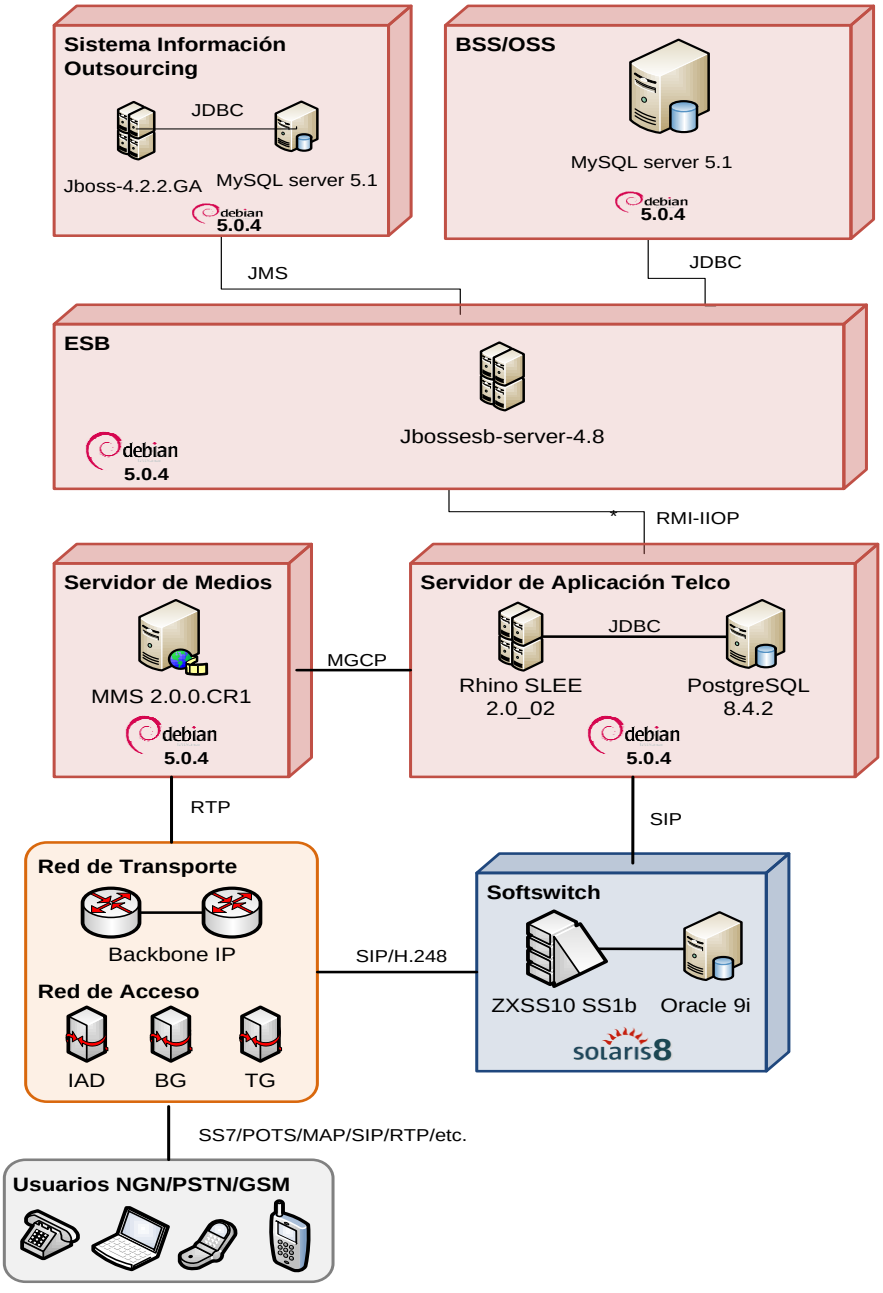



Figura 34 Diagrama de despliegue del sistema solución

***Softswitch***: nodo de control de la NGN de EMCALI que implementa funciones de procesamiento de control integrado, tales como control de llamadas, adaptación a protocolos

de acceso, interconexión e interoperabilidad. El Softswitch ZXSS10 SS1b de ZTE se encuentra instalado sobre un OS Sun Solaris 8, y utiliza una base de datos Oracle9i. Las características de la máquina dependen del contrato de venta del equipo, documento al cual no se tiene acceso. Sin embargo, en [14]se especifican los siguientes parámetros máximos: un BHCA alrededor de 17 millones de llamadas, una capacidad de soportar hasta 1500 puntos de 64K o 100 puntos de 2M de SS7, y un manejo de una variedad de protocolos, como SS7: INAP/CAP/MAP/TUP/ISUP, SIP, SIGTRAN, H.248, MGCP, H.323, entre otros.

***Servidor de aplicaciones Telco***: nodo que controla el ciclo de vida completo, la lógica de servicio y el almacenamiento de datos, del prototipo CRBT. Establece la conexión SIP con el Softswitch para el manejo de señalización de llamada, y MGCP con el servidor de medios para el control de la reproducción de componentes de audio. Se instaló Rhino SLEE versión 2.2_02 de OpenCloud, sobre un PC de escritorio HP Compaq dc 7800p con Intel Core 2 Duo E8300 @ 2.83GHz y 2GB de Memoria de Acceso Aleatorio (RAM, Random Access Memory). Se utilizó el OS Debian 5.0.4 (lenny), y la Máquina Virtual Java (JVM, Java Virtual Machine) versión 1.6.0_18 Java HotSpot 64-bit Server. El motor de base de datos instalado para este ambiente de ejecución es PostgreSQL 8.4.2, al cual se accede a través de JDBC. En este caso, Rhino SLEE no se implementó en la configuración de nodo por máquina, debido a escasos recursos computacionales y de red.

***Servidor de medios***: nodo encargado de reproducir el archivo de audio al usuario que inicia la llamada a través del Protocolo de Transporte en Tiempo real (RTP, Real-time Transport Protocol). Se instaló el MMS versión 2.0.0.CR1 de Mobicents, sobre un PC portátil Gateway M-1631U con AMD Turion X2 TL-60 2.0 GHz y 4GB de RAM. Se utilizó el OS Debian 5.0.4 (lenny) y la JVM versión 1.6.0_18 Java HotSpot 64-bit Server.

***Red de transporte***: proporciona todos los elementos de red del nivel de transporte de la NGN de EMCALI. Entre ellos se destaca el backbone IP (información detallada en el Anexo C).

***Red de acceso***: proporciona todos los elementos de red del nivel de acceso de la NGN de EMCALI. Entre ellos se destacan el Dispositivo de Acceso Integrado (IAD, Integrated Access Device), la Pasarela de Frontera (BG, Border Gateway) y la Pasarela de Troncal (TG, Trunk Gateway) (información detallada en el Anexo B).

***Mediador ESB****:* encargado de la mediación entre el servidor de aplicaciones Telco y los sistemas TI. Almacena el servicio de integración Telco TI el cual a su vez, utiliza los servicios SAR (Servicio de Actualización Rhino) y SAB (Servicio de Actualización BSS/OSS). Se instaló el jbossesb-server 4.8 en un PC portatil Gateway M-1631U con AMD Turion X2 TL-60 2.0 GHz y 4GB de RAM. Se utilizó el OS Debian 5.0.4 (lenny) y la JVM versión 1.6.0_18 Java HotSpot 64-bit Server.

***Sistema de Información Outsourcing****:* sistema encargado de ejecutar la aplicación de comercialización y personalización del servicio CRBT la cual ejecuta el servicio SP_CRBT hacia el mediador ESB encargado de terminar la tarea de integración. Se instaló el servidor

jboss 4.2.2 GA en un pc portátil HP ProBook 4320s Intel(R) Core (TM) i5 CPU M460 2.53 Ghz 3 GB RAM 32 bits.

***BSS/OSS***: el sistema BSS/OSS fue emulado con una base de datos Mysql, se instaló un MySQL server 5.1 en un pc portátil HP ProBook 4320s Intel(R) Core (TM) i5 CPU M460 2.53 Ghz 3 GB RAM 32 bits.

***Usuarios***: este componente hace referencia a todos los usuarios que interactúan con le red del operador EMCALI. Entre ellos se encuentran usuarios PSTN, NGN, clientes móviles, usuarios web y usuarios SIP.

### D.2.5 Vista de componente modulares

**Modelo de diseño**

Este modelo expone la forma del sistema desde el punto de vista de una organización estructurada de paquetes, Figura 35

**Nivel de aplicación**: representa los diferentes usuarios que acceden a la NGN de EMCALI, Ellos son suscriptores del servicio CRBT o clientes finales.

Cliente web: cliente que contiene un software en su PC para realizar llamadas a través de la red IP hacia la NGN de EMCALI o clientes que tiene el servicio CRBT suscrito y desea actualizar el perfil.

Cliente PSTN: usuario de una red PSTN bien sea de ECMALI o cualquier otra que tenga acuerdos con este operador

Cliente móvil: cliente asociado a una empresa móvil telecomunicaciones (TIGO, Movistar, COMCEL) que establece conexión con un usuario NGN de EMCALI

Cliente SIP: usuario que accede desde un teléfono SIP a la NGN de ECMALI

**Nivel de servicio**: expresa todo los componentes encargados del manejo de aplicación

CRBT: contiene las clases que ejecutan la lógica del servicio CRBT y el control del flujo de información entre los diferentes componentes.

Personaliza CRBT: contiene las clases que ejecutan la lógica para actualizar el perfil (nombre del audio).

Servicioesb.esb: contiene las clases que permiten la integración ESB – RhinoSLEE y el sistema BSS/OSS actualizando la tabla cargos de la base de datos.

Servicioweb actualiza CRBT: contiene las clases y las formas que le permiten al usuario actualizar el perfil desde un ambiente web.

**Nivel de control del servicio:** representa los componentes que realizan el manejo de la señalización SIP y MGCP con los nodos de control y de medios respectivamente

DAS: contiene la lógica de análisis de dígitos para el direccionamiento de los números destino hacia el nodo del servicio CRBT

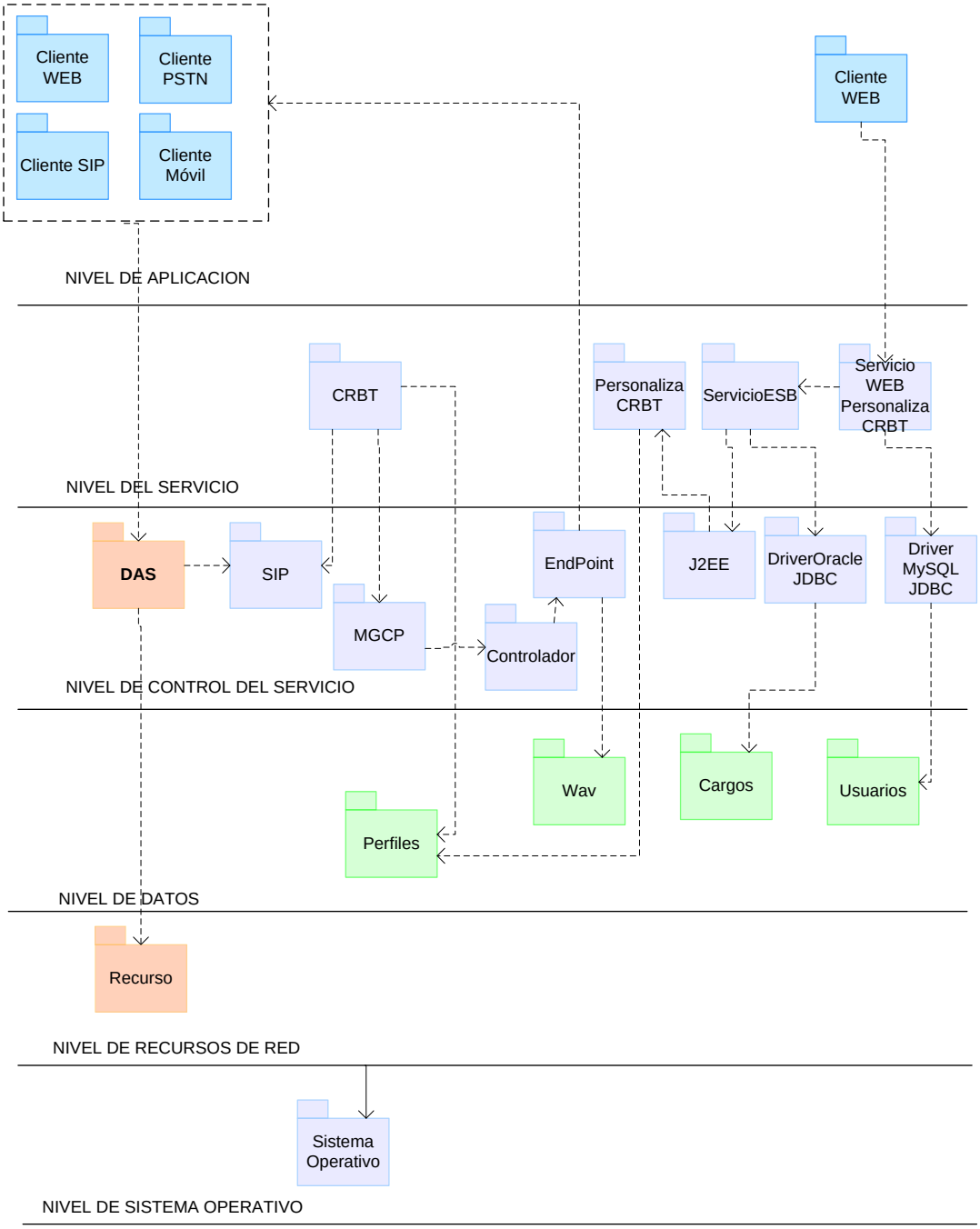



Figura 35 Diagrama de paquetes de diseño del sistema solución

SIP: contiene las clases que controla la señalización SIP. Estas clases reciben mensajes del nade de control e envían peticiones hacia el nodo del servicio

MGCP: contiene las clases que realizan la gestión de los mensajes del protocolo MGCP para la comunicación con el nodo de medios de servicio.

Controlador: contiene las clases para recibir solitudes e enviar respuestas a peticionas basadas en el protocolo MGCP.

Endpoint: contiene los archivos de configuración para establecer el Protocolo de Transporte en tempo Real (RTP, Real-time Transport Protocol) con los usuarios finales y seleccionar el archivo de audio de un perfil en particular.

J2EE: contiene las clases que realiza la integración ESB – Rhino. Gestiona los mensajes recibidos desde el ESB.

Driver Oracle, MySQL JDBC: contiene las clases necesarias para realizar las conexiones a las bases de datos Oracle y MySQL.

**Nivel de datos**: este punto maneja la persistencia de la información relacionada al servicio y al usuario

Perfiles: contiene las clases que incluyen el perfil de suscriptor del servicio

Wav: contiene los archivos de audio configurados dentro de los perfiles del suscriptor

Cargos: contiene las clases que incluyen los datos referente al cobro del servicio personalizar CRBT.

Usuarios: contiene las clases que permiten acceder a la información del usuario del servicio suscrito CRBT.

**Nivel de recursos red:** este nivel proporciona los recursos de red necesarios para el aprovisionamiento del servicio

Recurso: asigna los recursos más adecuados para manejar la solicitud de algún servicio en particular. Estos recursos varían según la red (NGN, Móvil, PSTN) en la que se envía la solicitud de llamada por parte de cualquier usuario del nivel de aplicación.

**Nivel de sistema operativo**: en este punto está contenido el software base para el despliegue de los servicios y componentes del sistema

Sistema operativo: hacer referencia a los sistemas Linux como Debain 5.0 y Sun solaris 8 sobre el cual corre tanto el servicio como los componentes de control de la NGN de EMCALI

# Anexo E

# Instalación y configuraciones

## E.1 Instalación y Configuración Rhino

A continuación se presenta una guía básica para la instalación y configuración de RHINO como Kit de Desarrollo de Software (SDK, Software develompment Kit) en Windows y como SLEE en el sistema operativo Linux.

RHINO SLEE en Linux antes de instalar y configurar RHINO se debe tener en cuenta el hardware y software necesario para la ejecución del servidor JSLEE. Es importante señalar que en caso de realizar pruebas de desempeño, alta disponibilidad y fiabilidad sobre el servidor, es necesario tener equipos hardware con soporte para 64 bits. De esta manera, el sistema operativo, JAVA y Postgres, deberán ser descargados según esta característica. Todo el software[27] requerido deber ser instalado con versiones de 64 bits a fin de realizar pruebas de rendimiento estables y adecuadas.

**Requerimientos de hardware, software y sistema operativo**

**Sistema operativo**

La versión de producción de RHINO está oficialmente soportada por Solaris 10 y RedHat AS4. Sin embargo, ha sido desplegado exitosamente sobre otras versiones de Linux. Para el caso específico del proyecto, la especificación 5.0 de Debian se tomó como sistema operativo base para el desarrollo del prototipo debido a su gran robustez, rendimiento y soporte frente otras versiones gratuitas encontradas en el mercado.

**Requisitos hardware**

Los requerimientos mostrados en la Tabla 8 corresponden a los recursos solicitados por la especificación 2.1 de RHINO.

**Requisitos software**

Instalación de Java RHINO SLEE 2.1 requiere Java SE 5 o 6. Es recomendable usar la versión más actualizada. Esta puede ser descargada desde http://java.sun.com. La instalación solo requiere ejecutar desde la ventana de comandos el archivo .bin descargado y seguir el wizard de configuración. Los pasos mostrados a continuación resume el proceso de instalación

[27] Para el desarrollo del prototipo se emplearon las versiones de Linux, Java y PostgreSQL, de 64 bits.

Ingresar como súper usuario en el Shell Ubicarse en la dirección donde esta descargado el instalador Ejecutar el archivo .bin ; Esto se hace digitando la sentencia ./Nombre completo del instalador incluida la extensión. Seguir el wizard de configuración Configuración de variables de entorno.

| HARDWARE | MÍNIMO | RECOMENDADO |
|---|---|---|
| Tipo de Maquina | Producto actual en hardware y CPU | |
| Numero de Maquinas (un nodo del clúster por maquina) | 1 | 2 o mas |
| Numero de núcleos CPU por maquina | 2 | 2 a 16 |
| RAM por maquina | 1 GB | 2+ GB |
| Interfaz de red | Ethernet | |
| Requerimientos de interfaz de red por maquina | 2 interfaces de 100MB (una interfaz para la comunicación del clúster) | 2 o más interfaces de 1 GB (una interfaz para la comunicación del clúster) |

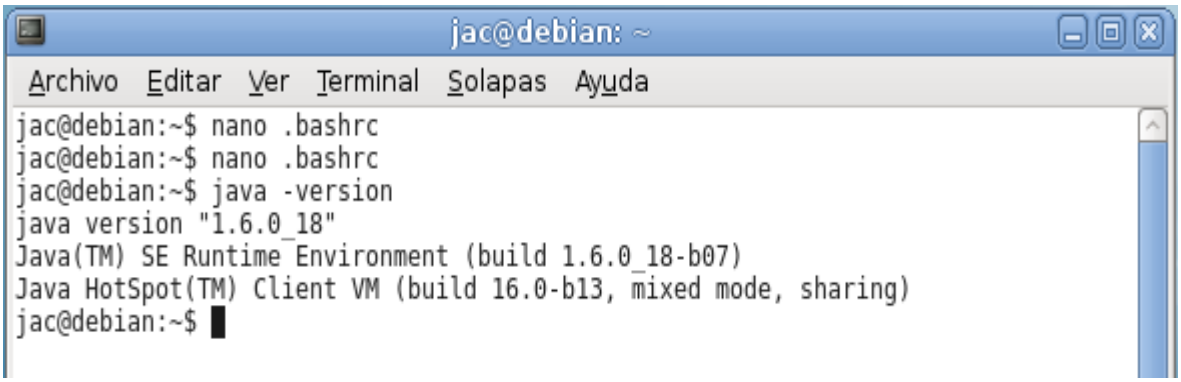
Tabla 8 Requisitos hardware Fuente: OpenCloud

Para ejecutar cualquier comando java sin ir hasta la dirección de instalación se debe configurar las variables de entorno. Los pasos abajo mencionados describen el proceso

Ingresar en el Shell (como usuario normal o super usuario)
Ejecutar la sentencia: nano .bashrc; nano puede cambiarse por gedit o por cualquier editor de texto
Agregar al archivo abierto el siguiente texto:

#Java configuration
export PATH= ruta completa de la carpeta donde se encuentra instalado java/bin:$PATH
JAVA_HOME**=** “ruta completa de la carpeta donde se encuentra instalado java” (comillas incluidas)
export JAVA_HOME

Guardar cambios
Desde el Shell ejecutar la sentencia: java -version; deberá aparecer algo como




**Instalación de PostgreSQL**

RHINO SLEE requiere de un sistema de gestión de base de datos postgres para almacenar el estado de memoria de trabajo. La memoria de trabajo principal de RHINO contiene el estado de ejecución de los nodos, el estado de configuración de los adaptadores de recursos, perfiles, etc. La base de datos postgres solo suministra un respaldo de la memoria de trabajo del servidor RHINO.

RHINO SLEE ha sido probado sobre las versiones 7.4.*, 8.1.* y 8.3.*. Este gestor de base de datos puede instalarse en una maquina independiente aunque generalmente es instalado sobre el mismo equipo. A continuación se mencionan los pasos para la instalación.

Descargar el archivo bin de Postgres desde http://www.postgresql.org/download/. (para este caso se uso la versión 8.3.9-1)
Ingresar como súper usuario en el shell
Ubicarse en la carpeta donde se encuentra ubicado el instalador
Ejecutar la sentencia ./nombre completo del instalador incluida la extensión
Seguir el setup wizard de configuración

Configuración de variables de entorno PostgreSQL

De igual forma que en la configuración de variables de Java, se edita el archivo .bashrc. El texto adicionado debería quedar de la siguiente forma:

#Java configuration
export PATH= ruta completa de la carpeta donde se encuentra instalado java/bin:/ruta completa donde se encuentra instalado Postgres/bin:$PATH
JAVA_HOME= “ruta completa de la carpeta donde se encuentra instalado java” (comillas incluidas)
export JAVA_HOME

Al utilizar las versiones 8 o superiores, la configuración para que Postgres acepte sockets TCP/IP del servidor RHINO no es necesaria, en caso de que se cuente con una menor, el parámetro tcpip_socket del archivo postgresql.conf ubicado en la carpeta DATA debe ser cambiado a tcpip_socket -1.

La configuración aquí expuesta es realizada bajo el supuesto de que Postgres y RHINO han sido instalados en la misma máquina, si por el contrario, la instalación del gestor de base de datos se realiza en otro componente hardware, el archivo pg_hba.conf ubicado en el directorio DATA deberá ser cambiado. Para más información consultar los documentos que vienen con el instalador de RHINO.

Configuración de direcciones IP, nombre de host y direcciones multicast

Antes de instalar el servidor RHINO es necesario configurar los siguientes parámetros. La expone dicha configuración

| CARACTERÍSTICA | DESCRIPCIÓN |
|---|---|
| Dirección IP | Asegúrese de que el equipo tenga una dirección IP visible en la red |
| Nombre de Host | Asegúrese de que el sistema pueda resolver (hacer ping) interfaces de loopback (127.0.0.1) y direcciones externas (diferentes a la de localhost) |
| Direcciones Multicast | Si el sistema local tiene un firewall instalado, modifique los permisos para permitir tráfico multicast UDP<br>Por definición, las direcciones Multicast están el rango 224.0.0.0/4 (224.0.0.0 a 239.255.255.255) . este rango está separado del rango de direcciones Unicast que la maquina usa para sus direcciones de host)<br>RHINO SLEE usa Mulicast UDP para distribuir su memoria principal de trabajo entre los miembros del clúster. Durante la instalación, estos parámetros son configurados. Por defecto, el número de puestos requeridos son: 45601, 45602, 46700 a 46800<br>Todos los nodos en el clúster deben usar la misma dirección multicast. Esto permite que se encuentren visibles.<br>Asegúrese de que el firewall este configurado para permitir mensajes multicast a través de los puertos y direcciones configuradas durante la instalación. |
| Reloj del sistema | Debido a que RHNO SLEE trabaja de manera síncrona en sus procesos, la configuración del reloj del sistema es prioritaria para este servidor. Encaso de que se tenga varias maquinas, léase los documentos de instalación para configurar el reloj del sistema. Para este caso solo se tienen un equipo, por la que la sincronización del reloj para este caso no es necesaria |

Tabla 9 Parámetros de configuración del equipo y de red Fuente: OpenCloud

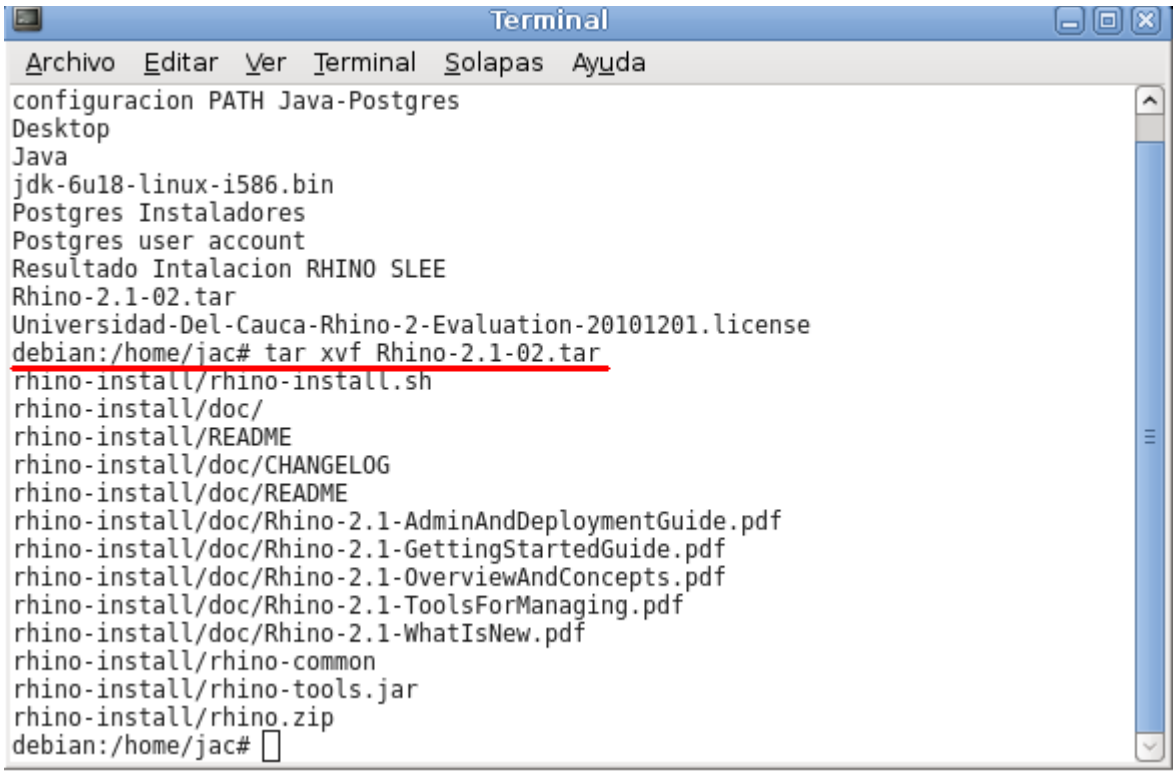
**Instalación y configuración de RHINO SLEE**

Antes de proceder, es necesario contar con la licencia educativa. Esta licencia es solicitada por escrito a un contacto de Openclud el cual evaluará la solicitud y enviará el link de descarga del servidor y el código de activación en caso de una respuesta afirmativa. La solicitud puede ser diligenciada en la siguiente dirección:
https://developer.opencloud.com/devportal/display/OCDEV/Educational+Community+License.

Recuerde que la versión SDK de RHINO y el RHINO SLEE son diferentes. El primero se enfoca en el desarrollo de aplicaciones basadas en JAIN SLEE soportadas por el sistema operativo Windows pero con restricciones en mecanismos contra fallos, configuración en clúster, cantidad de tráfico, entre otras. El segundo, es un servidor en ambiente de producción igualmente soportado por JAIN SLEE pero con la única restricción de volumen de eventos soportados, muchos más que la versión SDK pero menos que la versión licenciada de producción.

Los pasos mencionados a continuación corresponden al proceso de instalación y configuración del servidor RHINO en el sistema operativo debían 5.0

Ingresar como súper usuario en el Shell
Ubicarse en la carpeta donde se encuentra el archivo .tar de Rhino
Ejecutar la sentencia: tar xvf nombre completo del archivo Rhino incluida la extensión. Deberá aparecer algo como:



Ubicarse en la carpeta generada al descomprimir el archivo. La carpeta tiene como nombre rhino-install
Ejecutar el script rhino-install. Recuerde que los archivos .sh .bin y demás son lanzados con la sentencia ./nombre del archivo incluida la extensión. En esto caso sería algo como: ./rhino-instal.sh
Una vez ejecutado el comando, el script lanza un archivo el cual deber ser configurado según lo solicitado. La Tabla 10 contiene los parámetros de configuración para el servidor RHINO

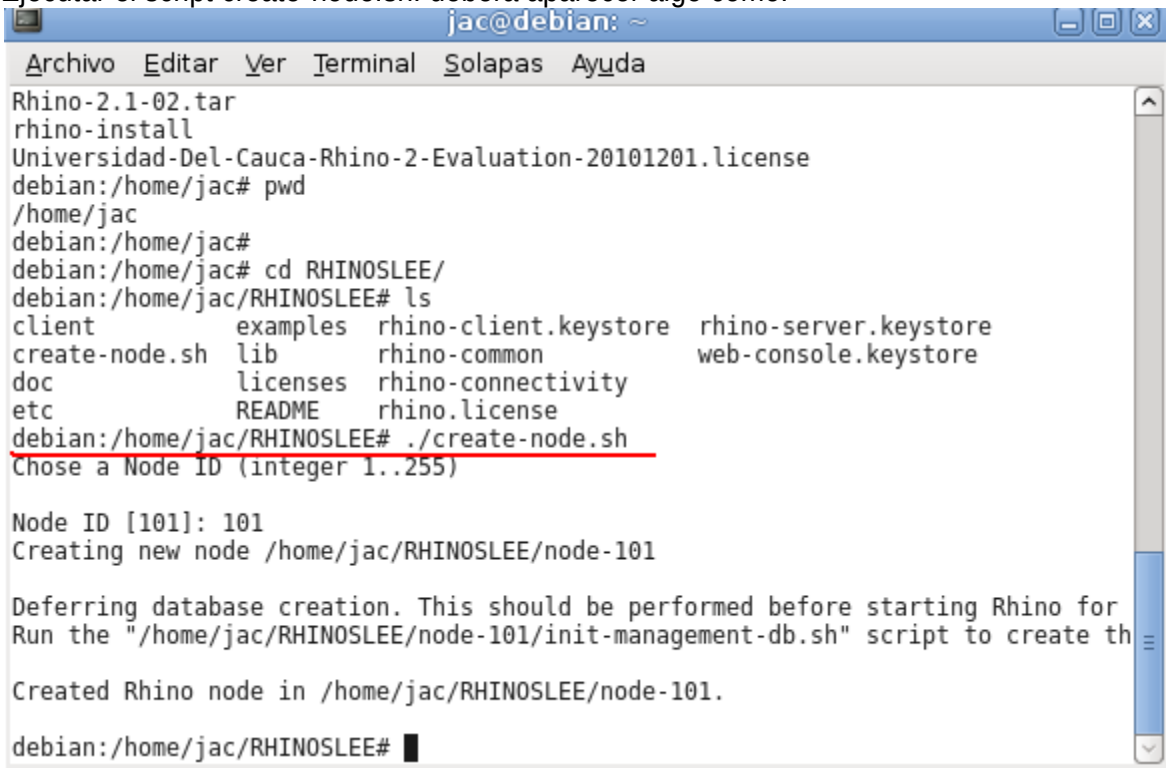
| PARÁMETRO | CONFIGURACIÓN POR DEFECTO | DESCRIPCIÓN |
|---|---|---|
| Directorio | /RHINO | Ruta del directorio raíz de RHINO. |
| Postgres Host | Localhost | Dirección IP donde se instalo el servidor Postgres |
| Puerto Postgres | 5432 | Puerto de comunicación del servidor Postgres |
| Postgres User | User | Nombre del usuario que se definió al instalar Postgres (en la mayoría de los casos se instala el servidor Postgres como postgres/postgres-) |
| Nombre base de datos | rhino | Sera creada en el servidor postgres y configurada con las tablas por defecto del RHINO |
| Puerto de registro de la interfaz gestión RMI | 1199 | Se usa para acceder a la gestión desde un cliente Java RMI a fin de usar líneas de comando RHINO |
| Puerto de objetos de la interfaz de gestión RMI | 1200 | Se usa para acceder a la gestión desde un cliente Java RMI a fin de usar líneas de comando RHINO |
| Puerto de servicio remoto para interfaz JMX | 1202 | Se usa para acceder al servidor remoto JMX. La consola web de rhino usa esto para gestión remota |
| Puerto de para comunicación HTPPS | 8443 | Se usa para la consola web y suministrar gestión remota |
| Directorio JAVA | | Directorio de instalación de java JSE/JDK. |
| Tamaño de heap | 512 | Argumento dentro del la JVM para especificar la cantidad de memoria (en MegaBytes) para que RHINO se ejecute |
| Clúster ID | 100 | Un único numero entero que identifica al clúster |
| Comienzo del pool de direcciones | 224.0.24.1 | Conjunto de direcciones multicast que serán usadas para la comunicación |
| Fin del pool de direcciones | 224.0.24.8 | |
| Ubicación del cliente psql | psql | RHINO necesita del cliente interactivo PostgrSQL, psql. Debe ir el PATH completo del cliente psql |

Tabla 10 Configuración de parámetros de RHINO SLEE Fuente: OpenCloud

En caso de que cometer un error en la configuración o se desee hacer algún cambio es necesario modificar el archivo config_variables ubicado en la dirección $RHINO_HOME/etc/defaults/config

**Crear nodos del clúster**

Ubicarse en la dirección base de RHINO
Ejecutar el script create-node.sh. deberá aparecer algo como:

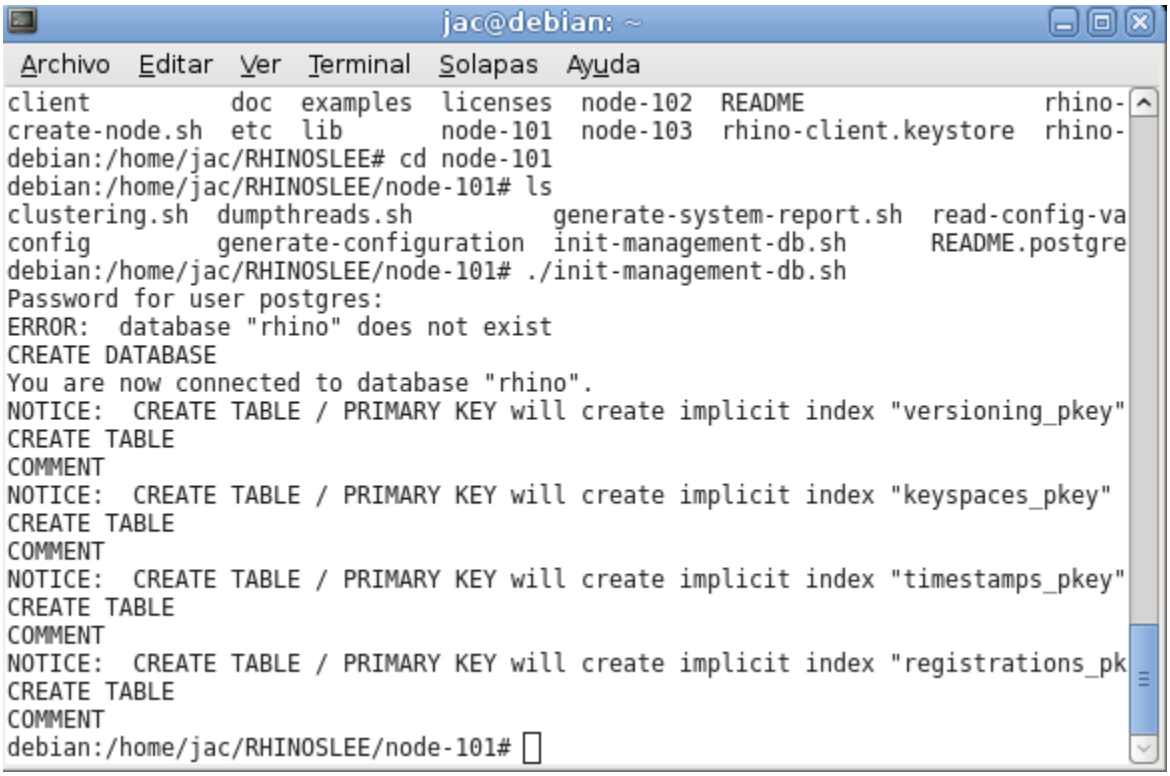



Repetir lo anterior para crear tantos nodos como desee. Para este caso específico se han creado tres nodos identificados como node-101, node-102, node-103 los cuales formaran el clúster de tres servidores.

**Inicializar la base de datos**

Los siguientes pasos mencionados son para la creación de la base de datos que RHINO necesita para sincronizar la memoria de trabajo entre los nodos y del almacenamiento de perfiles, entre otras características

Ubicarse en la dirección del que será el nodo principal. En este caso el nodo con id 101
Inicializar el servidor postgres. Esto se hace desde la pestaña de aplicaciones en caso de que se halla instalado cualquier ambiente grafico como GNOME. De lo contrario, lanzarlo por el Shell con el comando start
Dentro de node-NNN ejecutar el script init-management-db.sh; al dar los parámetros solicitados deberá salir algo como:




El script crea la base de datos que necesita el servidor para guardar el estado de trabajo. Esta inicialización de la base de datos solo se corre una vez. En caso de que los nodos sean eliminados es necesario ejecutarlo de nuevo desde un solo nodo.

**Arrancar el servidor RHINO**

A continuación se describe el procedimiento para arrancar los diferentes nodos que conformaran el clúster.

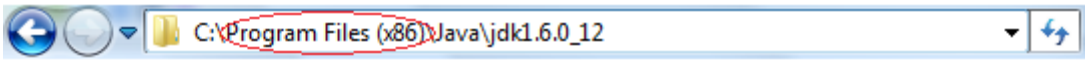
Ubicarse en la dirección del nodo principal
Ejecutar el script start-rhino.sh con los parámetros -p -s -k. (./start-rhino.sh -p -s -k). para más información acerca de los parámetros referirse a la documentación de RHINO o abrir el script con cualquier editor de texto
Acceder como súper usuario desde otro Shell a la dirección del nodo 2.
Ejecutar el script start-rhino con parámetros -s -k (./start-rhino.sh -s -k)
Acceder como súper usuario desde otro Shell a la dirección del nodo 3.
Ejecutar el script start-rhino con parámetros -q -k
Si todo ha salido bien, en el terminal de cualquier de los nodos deberá aparecer algo como: current membership set: [101,102,103]
Prueba de servicio de llamada

Una vez lanzados los nodos del clúster se procederá a desplegar las diferentes unidades que componen los servicios de Register, Call y SIP Proxy

Ingresar como súper usuario
Ubicarse en $RHINO_HOME/examples/sip-examples-2.2_02
Modificar el parámetro PROXY_DOMAINS dentro del archivo sip.properties por la dirección IP del servidor RHINO
Ejecutar el script deployesexamples.sh
Si todo sale bien, deberá salir algo como BUILD SUCCESSFULL
Instalar cualquier softphone y configurarlo con la dirección del servidor RHINO
Realizar la llamada entre usuarios
RHINO SDK en Windows
Antes de instalar la versión SDK de RHINO se debe realizar la configuración de Java, Postgres y tener en cuenta requisitos mínimos de memoria de la Tabla 8. En este caso se usó un equipo con 4 GB de memoria y sistema operativo Windows Vista

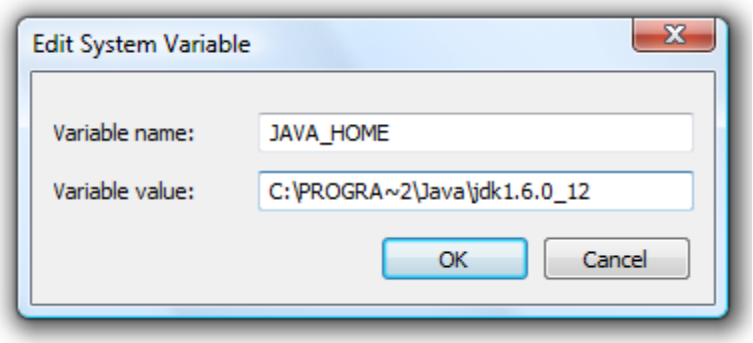
**Configuración Java:**

JDK 6 actualización 12 fue la versión usada (recuerde que RHINO funciona para la versión 5 o 6.).
Crear variable de entorno JAVA_HOME con la ruta de instalación de Java. Si existen espacios en dicha ruta, escribir el nombre corto generado por non-8dot3 para la carpeta o archivo en específico (utilizar comando dir /x dentro de la carpeta que lo contiene). En este caso la ruta de instalación de Java es C:/Program Files (x86)/Java/jdk1.6.0_12:

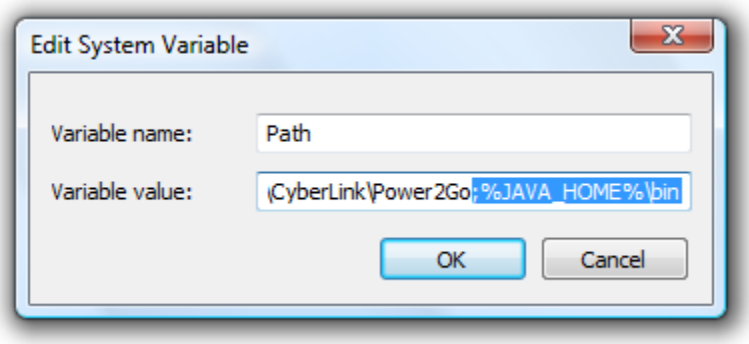



```
C:\>dir /x
 Volume in drive C has no label.
 Volume Serial Number is 2C24-2859

 Directory of C:\
    .
    .
11/08/2009  05:40 p.m.    <DIR>          PROGRA~1      Program Files
15/12/2009  06:52 p.m.    <DIR>          PROGRA~2      Program Files (x86)
    .
    .
```



Editar la variable de entorno Path, agregando al final del campo valor de variable el texto ;%JAVA_HOME%/bin:




Verificar que las variables de entorno se encuentran configuradas correctamente, utilizando los comandos java -version y javac -version en cualquier ubicación del equipo:

```
C:\>java -version
java version "1.6.0_12"
Java(TM) SE Runtime Environment (build 1.6.0_12-b04)
Java HotSpot(TM) Client VM (build 11.2-b01, mixed mode, sharing)

C:\>javac -version
javac 1.6.0_12
```

Configuración de red:

Asegurar que el sistema tiene una IP visible en la red.
Asegurar que el sistema puede resolver localhost a la interfaz de loopback utilizando el comando ping -4 localhost:

```
C:\>ping -4 localhost

Pinging PIPE-PC [127.0.0.1] with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 127.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Asegurar que el sistema resuelve el nombre de la máquina a una IP externa, no a la dirección de loopback, utilizando el comando ping -4 NombreMáquina (en este caso el nombre de la máquina es PIPE-PC):

```
C:\>ping -4 PIPE-PC

Pinging PIPE-PC [169.254.77.51] with 32 bytes of data:
Reply from 169.254.77.51: bytes=32 time<1ms TTL=128
Reply from 169.254.77.51: bytes=32 time<1ms TTL=128
Reply from 169.254.77.51: bytes=32 time<1ms TTL=128
Reply from 169.254.77.51: bytes=32 time<1ms TTL=128

Ping statistics for 169.254.77.51:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

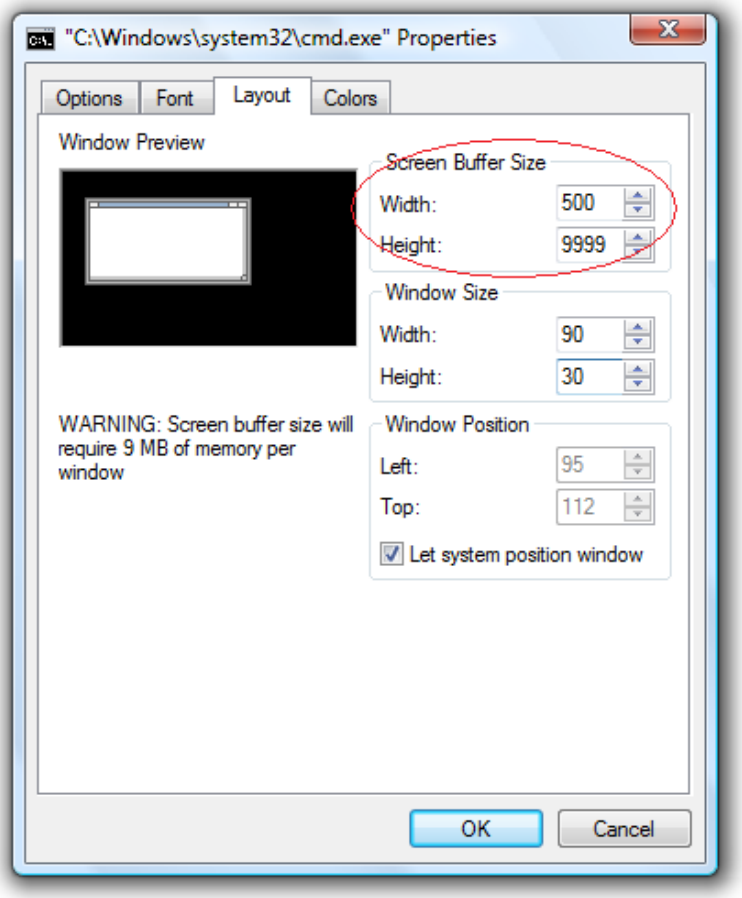
Desempaquetando Rhino SDK

Descomprimir el archivo zip en $RHINO_HOME, que en este caso es C:/RhinoSDK.
Leer las notas de la presente versión de Rhino SDK:

$RHINO_HOME/doc/index.html >> contiene información de último momento.
$RHINO_HOME/doc/CHANGELOG >> resume los cambios realizados entre las versiones previas de Rhino y la presente.

Ejecución de Rhino SDK

Modificar visibilidad de la ventana de comandos que ejecutará Rhino SDK: clic derecho en la barra de título de la ventana y seleccionar Properties. Luego, en la pestaña Layout, en la sección Screen Buffer Size, modificar el valor de Width a 500 y de Height a 9999, u otros valores que permitan observar mayores registros de Rhino SDK (Precaución: el tamaño del buffer necesitará 9 MB de memoria por ventana, así que sólo realizar esta modificación con la consola de comandos que ejecuta Rhino):




Iniciar Rhino SDK: en consola ejecutar start-rhino.bat dentro de la carpeta $RHINO_HOME.

Detener Rhino SDK: en otra consola ejecutar stop-rhino.bat –-[nice|terminate] dentro de la carpeta $RHINO_HOME.

Utilizar PostgreSQL (en el SDK también es posible utilizar Derby), ya que es la base de datos que utiliza la versión de producción de Rhino:

Instalar PostgreSQL v8.4.2 en $POSTGRESQL_HOME, que en este caso es C:/Program Files (x86)/PostgreSQL/8.4.
Digitar postgres como contraseña para el súper-usuario postgres y cuenta de servicio postgres.
Utilizar el puerto 5432 para el servicio.
El servidor acepta por defecto conexiones socket TCP/IP, ya que es una versión posterior a la 8.0. Verificar en $POSTGRESQL_HOME/data/postgresql.conf que el servidor se encuentre configurado de forma que escuche todas las direcciones IP: listen_addresses = '*'.
El servidor tiene por defecto configurado el control de acceso para las conexiones locales, necesario para el caso en donde Rhino y PostgreSQL se instalan en el mismo equipo. Verificar en $POSTGRESQL_HOME/data/pg_hba.conf que se encuentre una configuración similar a la siguiente:

```
# TYPE  DATABASE  USER     CIDR-ADDRESS        METHOD
# IPv4 local connections:
host    all       all      127.0.0.1/32        md5
# IPv6 local connections:
host    all       all      ::1/128             md5
```

NOTA – si Rhino y PostgreSQL son instalados en diferentes máquinas, se debe editar este mismo archivo y adicionar a la configuración, descrita anteriormente, la base de datos y el usuario de la conexión a realizar.

Si se han realizado cambios a los archivos de configuración de PostgreSQL mencionados, es necesario reiniciar completamente el servidor PostgreSQL para aplicar dichas modificaciones. Decirle al servidor que cargue nuevamente los archivos de configuración no habilita el funcionamiento de red TCP/IP como lo es cuando la base de datos es inicializada.

Editar $RHINO_HOME/config/config_variables de la siguiente forma:

```
# Management database settings
MANAGEMENT_DATABASE_NAME=rhino_sdk
MANAGEMENT_DATABASE_HOST=localhost
MANAGEMENT_DATABASE_PORT=5432
MANAGEMENT_DATABASE_USER=postgres
MANAGEMENT_DATABASE_PASSWORD=postgres
```

En $RHINO_HOME/config/rhino-config.xml comentar la configuración Derby: eliminar el final de comentario (-->) que aparece en la línea:

```
<!-- Begin Derby-specific configuration
To use PostgreSQL rather than the embedded Derby database, you'll want to comment out this
section and uncomment the PostgreSQL section below. -->
```

, y eliminar el inicio de comentario (<!--) que aparece en la línea:

```
<!-- End of Derby-specific database. -->
```

En $RHINO_HOME/config/rhino-config.xml descomentar la configuración PostgreSQL: adicionar el final de comentario (-->) inmediatamente después de la línea:

```
<!-- From here on is the configuration for PostgreSQL if you want to use that instead. -->
```

, y adicionar el inicio de comentario (<!--) inmediatamente antes de la línea:

```
<!-- End PostgreSQL-specific section -->
```

Inicializar la base de datos de PostgreSQL:

En consola ejecutar init-management-db.bat postgres (de ahora en adelante siempre se necesita pasar la palabra postgres a este archivo lote):

```
C:\RhinoSDK>init-management-db.bat postgres
Creating database...
Using PostgreSQL 8.4.2
ERROR: no existe la base de datos ½rhino_sdk¶
  .
  .
  .
CREATE DATABASE
Using PostgreSQL 8.4.2
Database has been initialised.
Press any key to continue . . .
```

Salida cuando se inicializa la base de datos por primera vez.

```
C:\RhinoSDK>init-management-db.bat postgres
Creating database...
Using PostgreSQL 8.4.2
DROP DATABASE
CREATE DATABASE
Using PostgreSQL 8.4.2
Database has been initialised.
Press any key to continue . . .
```

Salida cuando la base de datos ya ha sido inicializada anteriormente.

Al inicializar la base de datos se crean las tablas que utiliza Rhino SLEE y la tabla usada por el ejemplo de conectividad del protocolo SIP (RA y servicios) proveído por OpenCloud. Los comandos SQL ejecutados en esta acción, teniendo en cuenta que la base de datos utilizada es PostgreSQL, se muestran a continuación:

```
create table versioning (
  name text not null primary key,
  application text not null,
  ocbb text not null,
  description text not null,
  version integer not null
);
GO
CREATE TABLE keyspaces (
  dbid text not null,
  key_id text not null,
  mode int4 not null,
  timeout int4 not null,
  table_name text not null,
  primary key (dbid, key_id, mode)
);
GO
CREATE TABLE timestamps (
  dbid text not null primary key,
  era int8 not null,
  last_update int8 not null
);
```

```
GO
create table registrations ( -- for SIP location service
  sipaddress varchar(80) not null,
  contactaddress varchar(80) not null,
  expiry bigint,
  qvalue integer,
  cseq integer,
  callid varchar(80),
  flowid varchar(80),
  primary key (sipaddress, contactaddress)
);
GO
```

Iniciar Rhino SDK con la nueva configuración de base de datos PostgreSQL. Nuevamente en consola ejecutar start-rhino.bat dentro de la carpeta $RHINO_HOME.

## E.2 Instalación Jbossesb-server.

A continuación se presenta algunas consideraciones a tener en cuenta en la instalación del ESB server

El jbossesb-server puede ser instalado en sistemas operativos Windows o Linux.

Requerimientos mínimos del sistema:

Sistema Operativo:

El servidor jbossesb-server-4.8 puede ser instalado en Windows o en Linux, para el caso del desarrollo del prototipo y pruebas funcionales se usó Windows y para las pruebas de desempeño se utilizó Linux Debían 5.0.

Para el funcionamiento y compilación de los ejemplos y del servicio construido se debe instalar los siguientes componentes:

JDK 5 (v1.5.0_06 o superior), Instalar como se menciona en la sección anterior.
Ant (v1.6.5 o superior), Instalar como se menciona en la sección anterior.
JBoss ESB Server 4.8

Instalación y configuración:

La instalación y configuración del jbossesb-server se encuentra descrita en la guía de inicio del jbossesb-server [15].

# Bibliografía.

[1] OMA., "About OMA." *OMA.* [En línea] 2010. [Citado el: 26 de Agosto de 2009.] http://www.openmobilealliance.org/AboutOMA/Default.aspx.
[2] OMA Architecture Working Group (ARC)., *OMA Service Environment.* OMA. 2007. Reporte. OMA-AD-Service-Environment-V1_0_4-20070201-A.
[3] Sun Microsystems, Inc., *JAIN and Java in Communications.* Sun Microsystems, Inc. Santa Clara : s.n., 2004. Reporte de Referencia.
[4] ETSI TISPAN., *Open Service Access (OSA);Application Programming Interface (API);Part 1: Overview (Parlay 6).* ETSI. 2008. Estándar. ETSI ES 204 915-1 V1.1.1.
[5] ETSI 3GPP., *Universal Mobile Telecommunications System (UMTS);Virtual Home Environment (VHE)/Open Service Access (OSA).* 2006. Especificación Técnica. ETSI TS 123 127 V6.1.0.
[6] ETSI TISPAN., *Open Service Access (OSA);Application Programming Interface (API);Part 3: Framework(Parlay 6).* 2008. Estándar. ETSI ES 204 915-3 V1.1.1.
[7] Muller, Pierre Arnaud, y otros., *Service Oriented Architectures for convergent Service Delivery Platforms. Applying Service Oriented Architectures to Service Delivery Platforms.* EURESCOM. 2006. Reporte Técnico. EDIN 0532-1652.
[8] Oracle Communications., *Oracle Communications Converged Application Server.* Oracle. 2009. Hoja de Datos.
[9] Moriana Group., "Service Delivery Platforms – definition and evolution." *MORIANAGROUP.com.* [En línea] 2010. [Citado el: 22 de Enero de 2010.] http://www.morianagroup.com/index.php?option=com_content&view=article&id=357&Itemid=190.
[10] Nokia Corporation Networks., *Service Delivery Platforms.* Nokia. 2005. Reporte de Referencia.
[11] OpenCloud., *Rhino Core Platform.* OpenCloud. Hoja de Datos.
[12] —. *A service delivery platform for next generation telecommunications services.* OpenCloud. Wellington : s.n., 2009. Reporte de Referencia.
[13] David, Chapell., *Enterprise Service Bus.* s.l. : O'Reilly Media, 2004. 0-596-00675-6.
[14] ZTE-Corporation., *ZXSS10 SS1b Softswitch Control Equipment Technical Manual.* China : s.n., 2007.
[15] Jboss., Jboss Community. [En línea] [Citado el: 15 de 11 de 2011.] http://www.jboss.org/jbossesb.
[16] ZTE Corporation., *ZXUP10 System introduction.* ZTE Corporation. Shenzhen : s.n., 2008. Especificación Técnica. Versión V3.63.