BEA WebLogic Network Gatekeeper™

Product Description

Version 1.0
Document Revised: March 14, 2005

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

4. Application Development and Test

5. Service Provider and Application Administration

6. Network Interface

7. Policy

8. Charging

9. Operation, Administration and Maintenance

10.Alarm and Event Handling

11.Statistics and Performance Data

12.Security

13.Service Extensibility

14.Hardware Architecture

15.Software Architecture Overview

16.Software Module Descriptions

17.Redundancy and Load Balancing

A. Charging Data

B. Policy Data

C. Technical Specification

D. References

CHAPTER 1

Introduction and Roadmap

The following sections describe the audience for and organization of this document:

Document Scope and Audience

The purpose of the document is to provide descriptive information about WebLogic Network Gatekeeper. The following topics are covered:

- benefits and possibilities
- available functions
- support for application development and test
- hardware architecture and individual hardware components
- software architecture and individual software modules

Intended audience is business developers, marketing and sales personnel, support engineers, system administrators and other personnel with an interest in WebLogic Network Gatekeeper.

Guide to this Document

The document contains the following chapters:

- Chapter 1, "Introduction and Roadmap," informs you about the structure and contents of this document, writing conventions, and other WebLogic Network Gatekeeper related documentation.
- Chapter 2, "WebLogic Network Gatekeeper Introduction," introduces you to WebLogic Network Gatekeeper and its main benefits and possibilities.
- Chapter 3, "Service Descriptions," describes the functionality of the service capabilities available through WebLogic Network Gatekeeper.
- Chapter 4, "Application Development and Test," describes the Web Services APIs and the Application Development Tools for WebLogic Network Gatekeeper.
- Chapter 5, "Service Provider and Application Administration," gives an introduction to the administration related to the applications connected to WebLogic Network Gatekeeper through distributed APIs. It also gives an introduction to WebLogic Network Gatekeeper Partner Management Tools.
- Chapter 6, "Network Interface," describes WebLogic Network Gatekeeper's network interface and some of the prerequisites to be fulfilled before the communication with the network can be set up.

- Chapter 7, “Policy,” describes the functions of the policy engine and policy areas covered. It also shows the means for writing custom policies.
- Chapter 8, “Charging,” describes the charging possibilities offered by the WebLogic Network Gatekeeper and the means to integrate with external billing and settlemet systems.
- Chapter 9, “Operation, Administration and Maintenance,” describes the WebLogic Network Gatekeeper management application and gives an overview of the main Operation, Administration and Maintenance (OAM) tasks. In addition, the chapter describes PRM/CRM integration using WebLogic Network Gatekeeper Partner Management Tools.
- Chapter 10, “Alarm and Event Handling,” describes how alarms and events are handled by WebLogic Network Gatekeeper.
- Chapter 11, “Statistics and Performance Data,” describes the statistics and performance data generated by WebLogic Network Gatekeeper.
- Chapter 12, “Security,” gives an overview of the main security areas covered by WebLogic Network Gatekeeper.
- Chapter 13, “Service Extensibility,” describes the means for extending WebLogic Network Gatekeeper with new service APIs and network connectivity.
- Chapter 14, “Hardware Architecture,” describes the basic hardware configuration used by WebLogic Network Gatekeeper
- Chapter 15, “Software Architecture Overview,” describes the overall WebLogic Network Gatekeeper software architecture.
- Chapter 16, “Software Module Descriptions,” describes the individual WebLogic Network Gatekeeper software modules.
- Chapter 17, “Redundancy and Load Balancing,” describes the WebLogic Network Gatekeeper fault tolerance, high availability, and load balancing mechanisms from an application and network perspective.

Terminology

The following terms and acronyms are used in this document:

- API—Application Programming Interface
- Application—A telecom enabled computer application accessed either from a telephony terminal or a computer.

- Service Provider—An organization offering services provided by one or more applications to end users.
- AS—Application Server
- ATE—Application Test Environment
- CBC—Content Based Charging
- CORBA—Common Object Request Broker Architecture
- End User—Person that uses an application. An end user can be identical to a subscriber, for instance in a prepaid service. The end user can also be a non-subscriber, for instance in an automated mail-ordering application where the subscriber is the mail-order company and the end user is a customer to this company.
- Enterprise Operator —See Service Provider.
- ESPA—Extended and value added telecom web services APIs and service capabilities.
- HTML—Hypertext Markup Language
- IIOP—Internet Inter-ORB Protocol
- IN—Intelligent Network
- INAP—Intelligent Network Application Part
- IOR—Interoperable Object Reference
- IP—Internet Protocol
- JDBC—Java Database Connectivity, the Java API for database access.
- MAP—Mobile Application Part
- Mated Pair—Two physically distributed installations of WebLogic Network Gatekeeper nodes sharing a subset of data allowing for high availability between the nodes.
- MPP—Mobile Positioning Protocol
- NS—Network Simulator
- Operator—The Network Gatekeeper owner
- ORB—Object Request Broker
- OSA—Open Service Access

- PAP—Push Access Protocol
- Plug-in—A network plug-in the Network Gatekeeper to a network based service node or OSA/Parlay SCS through a specific protocol.
- SCF—Service Capability Function or Service Control Function
- SCS—Service Capability Server
- Service—A network provided service capability.
- Service Capability—See Service
- SIP—Session Initiation Protocol
- SLEE—Service Logic Execution Environment
- SLEE Service—A software module that is designed to execute in the SLEE.
- SMPP—Short Message Peer-to-Peer Protocol
- SMS—Short Message Service
- SMSC—Short Message Service Centre
- SNMP—Simple Network Management Protocol
- SOAP—Simple Object Access Protocol
- SPA—Service Provider APIs
- SS7—Signalling System 7
- Subscriber—A person or organization that subscribes for an application. The subscriber is charged for the service usage. See End User.
- SQL—Structured Query Language
- TCP—Transmission Control Protocol
- User—An application accessing services through one or more APIs and has a user name and a password or a person working with OAM through the Network Gatekeeper management tool that has an administrative user name and password.
- USSD—Unstructured Supplementary Service Data
- VAS—Value Added Service

- VLAN—Virtual Local Area Network
- VPN—Virtual Private Network
- XML—Extended Markup Language

Related Documentation

WebLogic Network Gatekeeper documentation

This product description is a part of the WebLogic Network Gatekeeper documentation set. The other documents are:

- User's Guide - WebLogic Network Gatekeeper
  The user's guide describes WebLogic Network Gatekeeper related operation and maintenance.
- Application Developer's Guide - Parlay X for WebLogic Network Gatekeeper
  The developer's guide describes how to design and implement applications using the Parlay X Web Services exposed by WebLogic Network Gatekeeper.
- User's Guide - WebLogic Network Gatekeeper Application Test environment
  The user's guide describes how to use the WebLogic Network Gatekeeper ATE when it comes to application test.
- API Descriptions - Parlay X for WebLogic Network Gatekeeper
  The API descriptions describe the WebLogic Network Gatekeeper APIs available for developers and applications.

CHAPTER 2

WebLogic Network Gatekeeper Introduction

The following sections provide an overview of WebLogic Network Gatekeeper functionality:

Overview

With the WebLogic Network Gatekeeper and its optional modules, operators can:

- Expose standard network capabilities (SMS, MMS, …) as telecom Web Services to third-party service providers
- Integrate and policify operator specific and legacy APIs
- Enforce real-time application authorization and access control
- Protect subscriber privacy
- Facilitate automation of third-party service provider and application provisioning and administration
- Integrate with existing billing, rating and settlement systems
- Reduce application development time

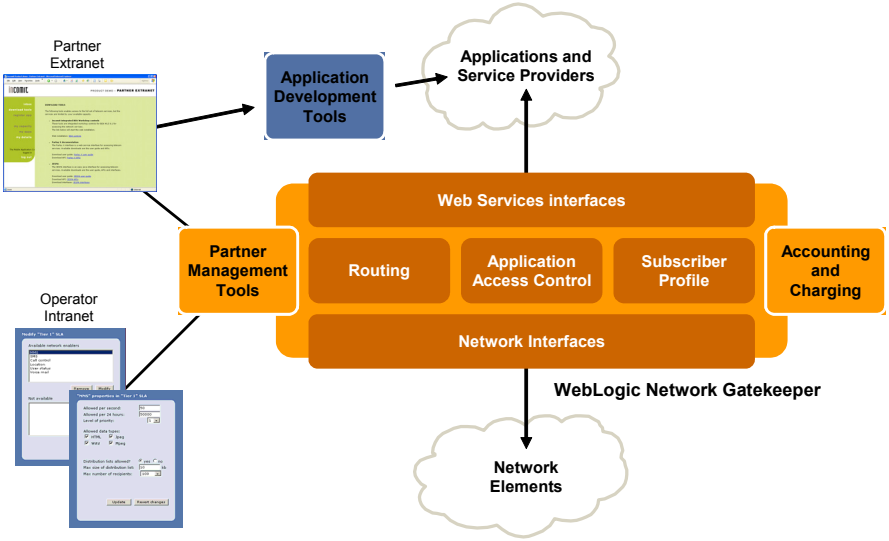


Figure 2-1 WebLogic Network Gatekeeper functional overview

Functions and benefits overview

The below functions are described in detail in the following chapters of the document.

Wide range of APIs based on Web Services standards

By exposing both traditional (legacy) telecom service APIs and new high-level web services, operators and application developers can select the most appropriate APIs for each application. The selection can be based on, for example, performance requirements, application complexity, or developer skills.

In addition, current applications using, for example SMPP and CIMD can be integrated with the WebLogic Network Gatekeeper "as is" and thereby leverage current application investments. That is, they can take advantage of the policy, charging, and SLA enforcement mechanism offered by the WebLogic Network Gatekeeper.

Common access control point for both internal and 3rd party applications

The integrity of the network and thesubscriber is secured by common authentication, authorisation, aliasing and access control procedures.

Policy-based execution for flexible application access control

Policy-based application access control allows the operators to dynamically customise both the Service Level Agreement (SLA) data and the actual access rules to fit their business models and security requirements.

Subscriber privacy and subscriber profile data

Using policy and subscriber profile data, the WebLogic Network Gatekeeper can perform a number of subscriber related checks before a service request is accepted. For example, WebLogic Network Gatekeeper can control which services the subscriber has subscribed to, payment method to use, account status, subscriber specific white and blacklists, and if aliasing shall be used. These checks are applied on both application and network initiated requests.

Wide range of standard network service capabilities

By providing a large set of standard network protocols operators can quickly provide applications with access to GSM, GPRS, SIP, IN and 3G service nodes directly through IP-based service nodes (SMSC, MMSC, …) or via OSA/Parlay gateways.

Built-in network routing

The WebLogic Network Gatekeeper routes service requests to the appropriate network nodes based on address plans and actual destination address.

Extensible application and network interfaces

Using a modular architecture where both the application and network interfaces are extensible makes it easy for operators to create attractive service offerings based on their unique network service capabilities. Also, new application and network interfaces are easily added when new or enhanced service capabilities are introduced in the networks.

Enhanced network protection

Service providers can be given various priority levels and their network access can be handled accordingly. The main functions provided by the WebLogic Network Gatekeeper are:

- Network node access control
- Network node traffic throttling
- Network node traffic shaping

Carrier grade and fully scalable architecture

The highly distributed and replicated system provides carrier grade performance. Linear scalability of both software and hardware allow operators to increase the capacity as the traffic and number of applications grow.

Optional modules for more operator value

To facilitate advanced accounting and charging, application development, and partner management, the following optional modules are available for the WebLogic Network Gatekeeper:

Partner Management Tools

This module provides operators with tools to manage large sets of partners. The tools support automation of traditionally work intensive tasks such as registration, activation, administration and supervision of 3rd party and in-house service providers and their applications.

The tools also allow operators to create groups of partners sharing sets of data. This functionality can be used for tiering or segmentation of partners allowing operators to focus their administrative and partner management resources on the most rewarding partners.

Application Development Tools

This module provides application developers with a set of development tools that allows them to work in standard development environments. The module consists of the following:

- Controls for BEA WebLogic Workshop
- Web Services WSDL files including API desciptions
- Developers' Guides
- A graphical test and verification environment

Supporting standards based development environments, the application development tools allow application developers to concentrate on the APIs instead of proprietary development environments. This gives the developers a head start and reduces development time.

CHAPTER 3

Service Descriptions

The following sections describe the service capabilities of WebLogic Network Gatekeeper:

Overview

This chapter describes the service capabilities available to applications through WebLogic Network Gatekeeper. The descriptions are on a functional level. Each service capability is provided through the Extended Web Services API, as well as through the Parlay X Web Services APIs. The amount of service features supported varies among the APIs. For more information, see Chapter 4, "Application Development and Test" and the specific API descriptions:

- API Descriptions - Extended Web Services for WebLogic Network Gatekeeper
- API Descriptions - Parlay X for WebLogic Network Gatekeeper

Thanks to the modular architecture, it is easy to extend the WebLogic Network Gatekeeper's functionality with legacy and operator specific APIs. For more information on the service integration and extensibility, see Chapter 13, "Service Extensibility".

Messaging

The messaging service capability makes it possible for an application to send, store, and receive SMSes and Multi Media (MMS) messages. In addition, the service capability supports send lists, smart messaging, EMS, and distribution of ring tones and logos. The send list feature allows for send list distribution of messages.

An operator administrator creates mailboxes with INBOX and OUTBOX folders for each subscriber or application.

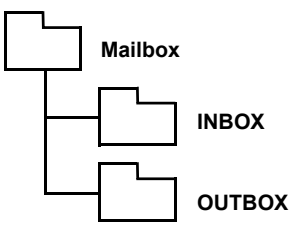


Figure 3-1 Mailbox structure

An application is notified when a sent message has been successfully delivered to a recipient and when the messaging service receives a message in an INBOX related to the application. Old messages are automatically removed from the mailboxes. The cleanup interval and age of messages to be deleted are configurable.

In addition, destination address short codes and message prefixes can be connected to a mailbox.

A destination address short code is a number that is used by the end user instead of the real mailbox address. The same destination address short code can be used for several mailboxes if it is combined with a message prefix. The message prefix is a string entered by the end user as the first part of the message.

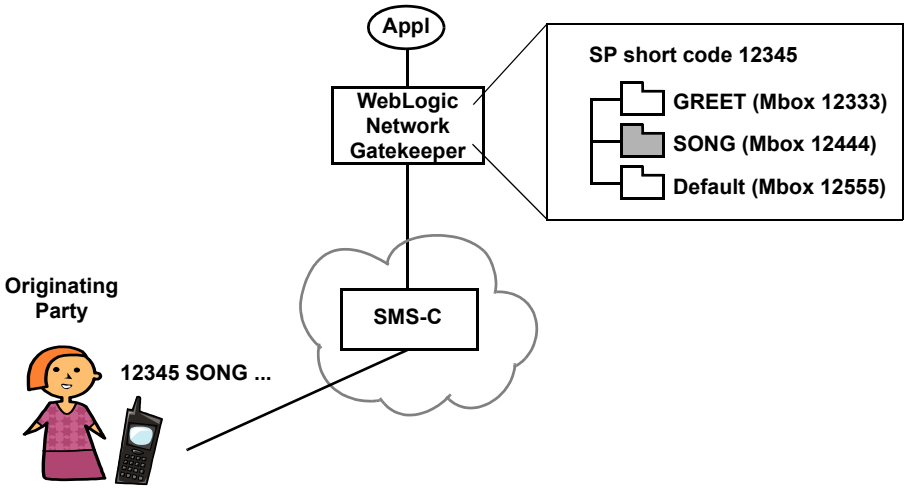


Figure 3-2 Example of messaging using short codes

For example, a service provider can have a destination address short code that is used to access all the service provider's messaging based applications. For example 12345. The messages are distributed among the service provider's mailboxes through the use of message prefixes. In this case, each application has its own mailbox. Let's say that the service provider has two applications aimed for a radio show, one for greetings and one for requesting songs. The message prefix for to use can be defined as GREET and SONG. These prefixes are translated by the service capability into the actual mailbox addresses, for example 12333 and 12444.

That is, if an end user wants to request a song, he or she enters 12345 as destination address and starts the actual message with SONG.

In addition, if the service provider wants a general mailbox that is not connected to a specific task, it is possible to define a default mailbox using the same destination address short code (12345). When specifying a default mailbox, no message prefix is specified. This means that all messages

sent to 12345 that does not start with GREET or SONG is delivered to the default mailbox, for example 12555.

Charging

The charging service capability makes it possible for an application to charge an end user based on the content (content based charging, CBC) of a used service, for example a song request or a music video, rather than based on the amount of time used. Reservation/payment in parts and immediate charging are supported.

In the example shown in Figure 3-2, "Example of messaging using short codes," on page 3-3, the end user could be charged one amount for requesting a song, and another amount for sending a greeting.

In immediate charging the amount is withdrawn from the subscriber's account at the same time the service is ordered.

To make sure the subscriber does not have to pay for a service not delivered, reservation/payment in parts can be used. In this case, the charging service reserves the whole or a part of the amount in the subscriber's account. The amount is not withdrawn until it has been verified that the subscriber has received the whole or a defined part of the service paid for. Reservation/payment in parts can also be used by an application before delivering a service to make sure that the amount to be charged is available on the subscriber's account.

The charging service capability also supports adding money to subscriber accounts.

Call

The call service capability provides applications with functions for call routing, call management, and call leg management. More than two call legs can be connected to a call simultaneously.

Two main usage scenarios for call control are identified; application initiated and network triggered calls.

Network triggered calls

Network triggered call is used for applications where call set up is triggered from the network, see Figure 3-3, "Example of a network triggered call," on page 3-5.

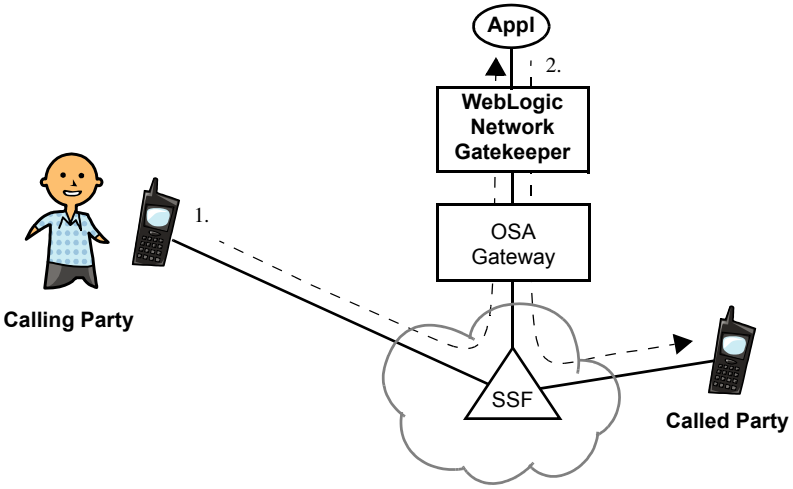


Figure 3-3 Example of a network triggered call

The call service capability contains functions that makes it possible for an IN/VAS application to provide:

- call re-routing
- user interaction through announcements and voice prompts
- call leg handling

An application can combine call control with user interaction, user location, and user status. Together, these services make it possible to build all types of traditional IN/VAS services.

Application initiated calls

Typical usage for application initiated calls are voice chat applications and different types of click-to-call functionality in web and office applications. Figure 3-4, “Example of an application initiated call,” on page 3-6 shows an example where a call is set up using a web based address book application.

In addition, a call between two or more persons can be set up through an application interface. During the call it is possible to add and remove participants through the interface. Also, notifications when individual participants answers and hangs up can be presented.

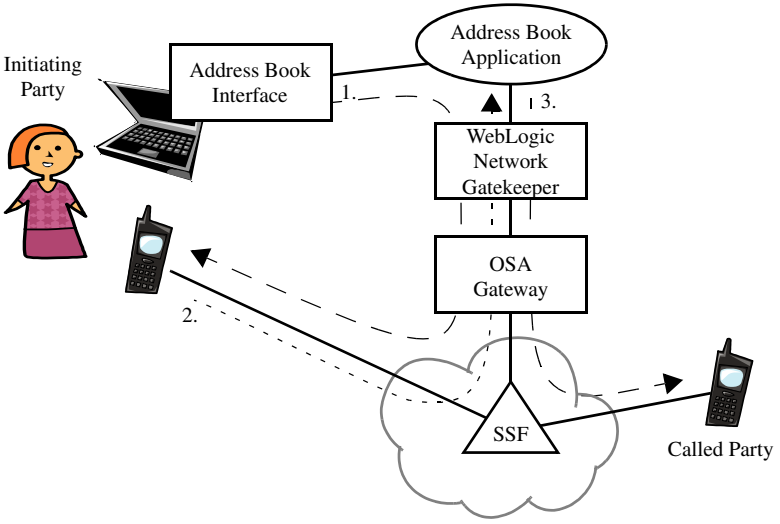


Figure 3-4 Example of an application initiated call

Subscriber profile

The subscriber profile service capability makes it possible for an application to obtain and manage application subscriber profiles. A subscriber profile consists of data related to a subscriber and the subscriber's telephony terminal, see Table 3-1 on page 6. The data marked as *read-only* in the table can only be updated by the operator.

Table 3-1 Subscriber Profile Data

Data	Description/Example
Name	Subscriber name
Alias	Alias to ensure the subscribers anonymity towards other users.
Address	Complete postal address
Home phone	-

Office phone	-
Private mail	Home e-mail
Office mail	Office e-mail
Terminal ID	IMSI number
Terminal vendor	For example Nokia or Ericsson
Terminal model	For example 6610 or T630
Screen size	Character rows x columns
Colour terminal	Yes/No
MMS terminal	Yes/No
Fax number	-
Group identity	For example family, office location or work group
Gender	Male/Female
Birth date	In format YYYY-MM-DD
Nationality	-
Mother tongue	-
Currency	-
Miscellaneous	Any type of additional information
Last updated	Date and time the account was last updated (read-only)
Updated by	The user that updated the account (read-only)

Subscription type	Type of subscription, Prepaid, Postpaid, Time Limited, or Free (read-only)
Payment method	Payment method: Credit card or Invoice (read-only)
Balance	Account balance (read-only)
Application subscriptions	List of subscribed applications (combinations of service provider and application IDs) (read-only)

User interaction

The user interaction service capability makes it possible for an application to interact with call participant(s) during a call or with messaging users during a messaging session. The application communicates with call participant(s) through announcements or messages and with messaging users through text messages.

Call based user interaction

The call participant(s) communicate through speech or tone sending. That is, both speech recognition and DTMF (using the terminal's key set (0-9, *, #)) can be used. Announcements can be purely informative or they can prompt a participant to reply through speech or sending DTMF tones back to the application.

Message based user interaction

With message based user interaction, the application and the end users communicate through text messages (SMSes or USSD messages).

SMS based user interaction provides application initiated SMSes with a transaction ID to connect requesting/prompting SMSes with end user's replies.

USSD messages from an application can be purely informative or they can prompt the end user to reply. USSD messages can also be used by the end user to initiate service sessions with applications. When initiating service sessions or replying to an application generated USSD message, the end user can only use the terminal's key set (0-9, *, #). The application can use any type of character supported by the end user's terminal.

User location

The user location service capability makes it possible for an application to obtain the geographical location parameters of telephony terminals. The service capability supports:

- single location requests
- periodic location request

Both single and periodic requests supports multiple destination addresses in one request.

The user location can be specified as a base point (longitude, latitude) or as a descriptive (abstracted) position.

Using longitude and latitude, the location is specified as a base point (longitude, latitude) and a geometrical area in which the telephony terminal is located. The geometrical area is referred to as an uncertainty shape related to the base point. The uncertainty shapes are expressed in either circles or ellipses.

Using an abstracted position describes the user's location in terms of:

- Street address
- Zip code
- City
- State
- Area (operator defined)
- Country
- Network (operator defined)

Use of abstracted location information requires interaction with a geographic information system.

When supported in the network, extended location information is also provided, such as altitude, terminal type, and time stamps.

Also, the service capability supports provisioning of geographical information for a terminal, such as city or street address, to applications.

Exactly which parameters that are available to the requesting applications is dependant on the underlying network equipment.

Circle and ellipse uncertainty shapes

When requesting a user location, a set of parameters is returned from the network which describes a circle or ellipse uncertainty shape. Which set of parameters is returned depends on the network equipment. The following three uncertainty shapes can be described:

- Full circle or ellipse
- Circle or ellipse sector
- Circle or ellipse arc

If the network equipment provides applications with more parameters, the calculations of user locations will be more accurate. Figure 3-5, "Parameters for calculating circle uncertainty shapes," on page 3-10 and Figure 3-6, "Parameters for calculating ellipse uncertainty shapes," on page 3-11 shows that using more parameters narrows the shaded area in which the user location is found.

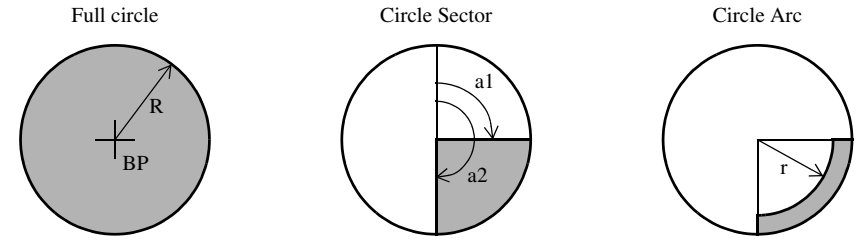
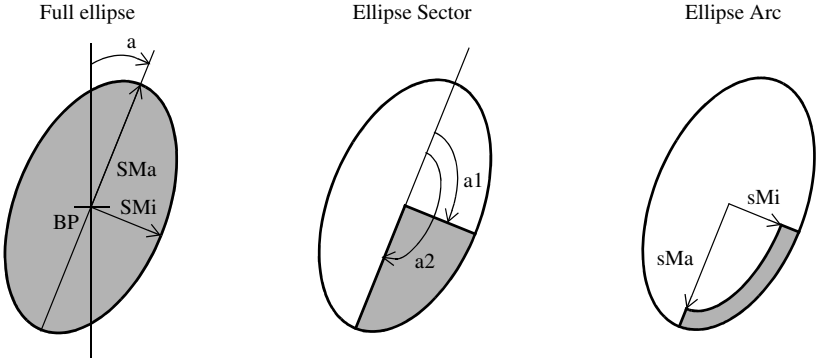


Figure 3-5 Parameters for calculating circle uncertainty shapes



Parameters:

a - Angle of Semi Major	+	a1 - Segment Start Angle	+	sMa - Inner Semi Major
BP - Base Point		a2 - Segment End Angle		sMi - Inner Semi Minor
SMa - Semi Major				
SMi - Semi Minor				

Figure 3-6 Parameters for calculating ellipse uncertainty shapes

Terminal altitude

If the terminal's altitude is provided, the actual terminal altitude is somewhere within a span defined by the provided altitude value and two times the altitude uncertainty, see Figure 3-7, "Terminal altitude definition," on page 3-12.

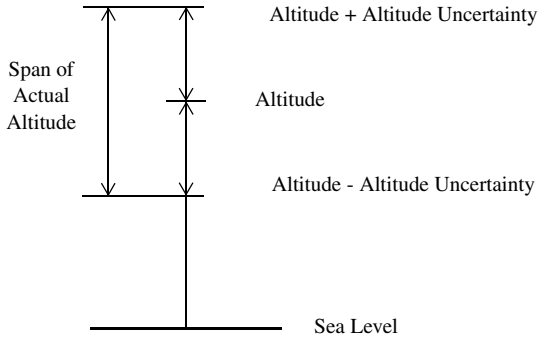


Figure 3-7 Terminal altitude definition

A positive altitude value means above sea level, whereas a negative value means below sea level.

User status

The user status service capability makes it possible for an application to obtain the status of fixed, mobile and IP-based telephony terminals. Possible values are:

- Reachable
- Busy
- Not reachable

The service capability supports both single and periodic status request and as well as multiple destination addresses in one request.

CHAPTER 4

Application Development and Test

The following sections describe WebLogic Network Gatekeeper development tools:

Overview

Application developers wanting to access telecom network functionality through WebLogic Network Gatekeeper can use WebLogic Network Gatekeeper Application Development Tools. They include the following:

- Easy to use Web Services APIs
- Controls for WebLogic Workshop
- Network Gatekeeper Application Test Environment
- Developer's guides
- API descriptions

Besides the Web Services APIs, legacy APIs as SMPP and CIMD can be exposed through the WebLogic Network Gatekeeper.

Web Services APIs

The standard Parlay X Web Services APIs and the value added Extended Web Services API make it possible for developers to manage with only a basic understanding of the provided network services. In addition, current applications using, for example SMPP and CIMD can be integrated with the WebLogic Network Gatekeeper "as is" and thereby leverage current application investments.

Applications accessing network services through WebLogic Network Gatekeeper can execute in practically any application server or external process. The communication with WebLogic Network Gatekeeper is made through a SOAP/HTTP connection, see Figure 4-1, "WebLogic Network Gatekeeper application interfaces," on page 4-3.

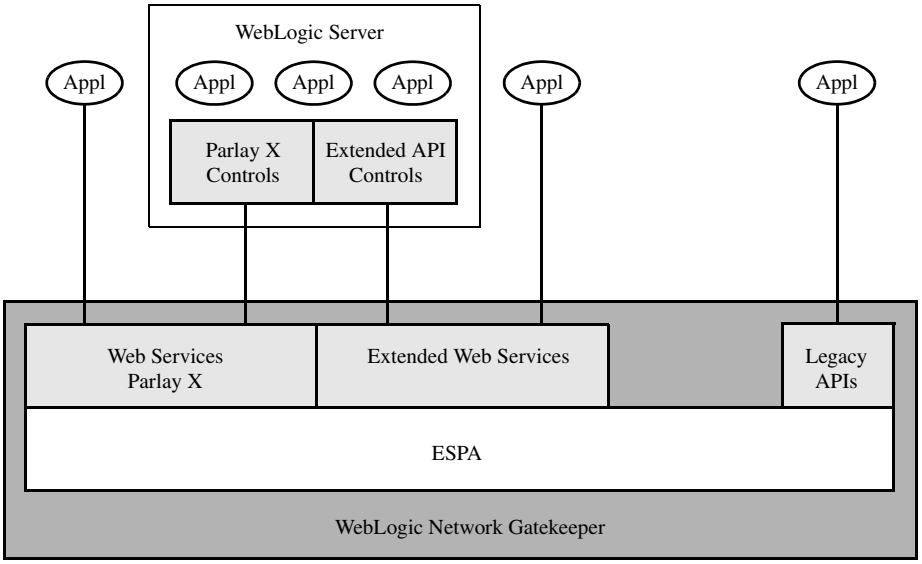


Figure 4-1 WebLogic Network Gatekeeper application interfaces

Extended API compared to Parlay X

Both the Extended API and Parlay X provides applications with high-level and easy to use Web Service APIs towards the network service capabilities. Parlay X is an industry standard specified by the Parlay Group. The Extended API are developed by BEA. The Extended API are, compared to Parlay X, more feature rich in its service implementations. Also, the Extended API has support for authentication and authorisation of service providers and applications. The below comparison tries to show the differences between the two when they provide interfaces to the same network service capability.

In general, the Extended API make it possible for the applications to handle callbacks and notification listeners. Using Parlay X this has to be handled through manual OAM procedures.

Also, the Extended API often make it possible to choose synchronous or asynchronous mode for the requests where Parlay X only provides synchronous mode with polling from the applications.

User interaction and access (application authentication/authorization) interfaces are only provided by the Extended API. Both these interfaces can, as well as all other, be used together

with any of the Parlay X interfaces. That is, it is possible for an application developer to select the most appropriate interface and use the Extended API to access some service capabilities and Parlay X for other.

Parlay X is specified by OSA/Parlay and the services provided by Parlay X is decided by standardization bodies. Using the Extended API, an operator can provided it is own proprietary services through a Web Services API. See Chapter 13, “Service Extensibility”.

Access

The WebLogic Network Gatekeeper offers an authentication/authorization interface through Extended API access. No access interface is defined in Parlay X. The Extended API access authentication/authorization interface can be used together with the Parlay X interfaces.

Extended API -Access	Parlay
• applicationLogin • applicationLogout • change ApplicationPassword	No API

Messaging

The Extended API makes it possible to handle enabling/disabling of notifications through the applications. In Parlay X this has to be handled through manual OAM procedures. Moreover, the Extended API make it possible for applications to administrate their mailboxes. For example, an application can open and close the mailbox, list messages in the mailbox, and delete messages from the mailbox.

Both the Extended API and Parlay X offers the possibility to send logos and ringtones through SMS.

Extended API -Messaging	Parlay X SMS
• closeMailBox • deleteMessages • disableMessagingNotification • enableMessagingNotification • getMMS • getMessageProperties • getSMS • listMessages • listNewMessages • openMailbox • sendEMail • sendMMS • sendSMS	• getReceivedSms • getSmsDeliveryStatus • sendSms • sendSmsLogo • sendSmsRingtone **Parlay X Multimedia Message** • getMessage • getMessageDeliveryStatus • getReceivedMessages • sendMessage

Charging

The the Extended API and Parlay X interfaces provides almost the same functions when it comes to content based charging and payment. The main difference is terminology, the Extended API use debit/credit while Parlay X uses charge/refund.

Extended API -Content Based Charging	Parlay X Payment
• close • createChargingSession • creditAmountWait • creditUnitWait • debitAmountWait • debitUnitWait • directCreditAmountWait • directCreditUnitWait • directDebitAmountWait • directDebitUnitWait • getAmountLeftWait • getSessionId • getUnitLeftWait • rateRequestWait • reserveAmountWait • reserveUnitWait	• chargeAmount • chargeReservation • chargeVolume • getAmount • refundAmount • refundVolume • releaseReservation • reserveAdditionalAmount • reserveAdditionalVolume • reserveAmount • reserveVolume

Call

The Extended API supports two and multiparty call and provides the possibility to add and remove participants to a call. In addition the Extended API makes it possible for the applications to add and remove call listeners themselves. The Extended API also supports call user interaction.

<table>
<tr><th>Extended API -Call</th><th>Parlay X Network Initiated Call</th></tr>
<tr><td rowspan="3">• addListener
• addNetworkCallListenerRequest
• addParticipantRequest
• addParticipantWaitRequest
• createCallRequest
• createEmptyCallRequest
• deassignRequest
• endRequest
• getOriginatorRequest
• getParticipantsRequest
• removeNetworkCallListenerRequest
• removeParticipantRequest</td><td>• handleBusy
• handleCalledNumber
• handleNoAnswer
• handleNotReachable
• handleOffHook</td></tr>
<tr><td>Parlay X Third Party Call</td></tr>
<tr><td>• cancelCall
• endCall
• getCallInformation
• makeACall</td></tr>
</table>

Subscriber Profile

The Extended API provides the possibility the set and get subscriber profile data in both synchronous and asynchronous mode.

Extended API -Subscriber Profile	Parlay
• getSubscriberProperty • getSubscriberPropertyWait • setSubscriberProperty • setSubscriberPropertyWait	No API

User Interaction

Extended API -Messaging User Interaction	Parlay X
• close • createChargingSession • creditAmountWait • creditUnitWait • debitAmountWait • debitUnitWait • directCreditAmountWait • directCreditUnitWait • directDebitAmountWait • directDebitUnitWait • getAmountLeftWait • getSessionId • getUnitLeftWait • rateRequestWait • reserveAmountWait • reserveUnitWait	No API
Extended API -Call User Interaction	
• abortAction • close • createCallUserInteraction • sendInfoAndCollect • sendInfoAndCollectWait • sendInfo • sendInfoWait	

User Location

Besides single location requests in both synchronous and asynchronous mode, the Extended API provides support for triggered, periodic, extended and geo location requests. Parlay X has support for single location request only.

Extended API -User Location	Parlay X Terminal Location
• getExtendedLocation • getExtendedLocationWait • getGeoLocation • getGeoLocationWait • getLocation • getLocationWait • startPeriodicGeoLocation • startPeriodicLocation • startTriggeredLocationReporting • stopPeriodicGeoLocation • stopPeriodicLocation • stopTriggeredLocationReporting	• getLocation

User Status

The Extended API provides support for single status request in both synchronous and asynchronous mode and for triggered status reports. Parlay X has support for single status request only.

Extended API -User Status	Parlay X User Status
• getStatusRequest • getStatusWaitRequest • startTriggeredStatusReportingRequest • stopTriggeredStatusReportingRequest	• getUserStatus

Developer's guides and API descriptions

Application developers are provided with developer's guides and API descriptions for Extended Web Services and Parlay X

The developer's guides can be used together with any Java design environment. The guides provide information about:

- Recommended design workflows
- How to use the APIs

Controls for WebLogic Workshop

WebLogic Network Gatekeeper Application Development Tools provide controls for the WebLogic Workshop. Developers are presented with drag-and-drop graphical components representing features in the Extended API and Parlay X.

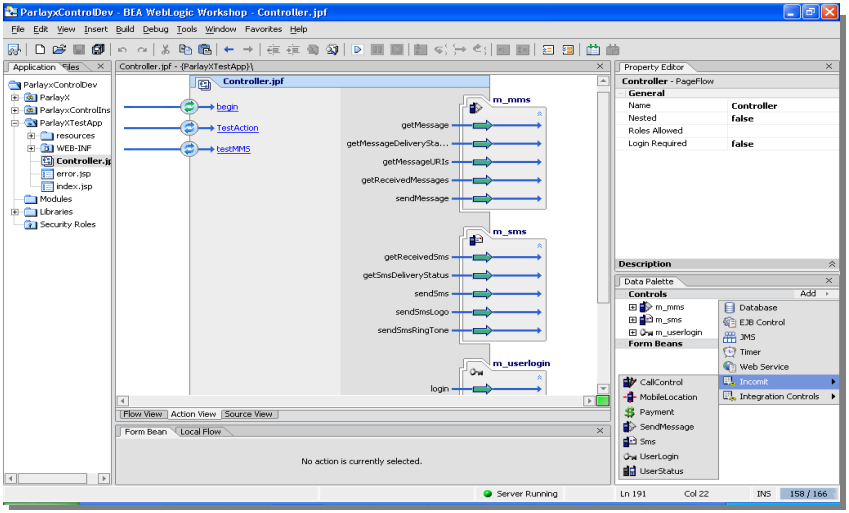


Figure 4-2 Network Gatekeeper controls for WebLogic Workshop

The controls are embedded as WebLogic Server components and deployed into the WebLogic Server as enterprise Java beans, which provides high performance, scalability and extensibility.

On-line help is provided for the developers using the Extended API or Parlay X Controls in WebLogic Workshop.

Test flow

WebLogic Network Gatekeeper provides test tools that make it possible to perform both functional and non-functional tests on an application before running the application against a test or live network. Figure 4-3, "Application test flow," on page 4-11 shows the complete application test flow, from the developers' functional test to deployment in a live network. An application developer performs functional tests using Network Gatekeeper ATE, see "Network Gatekeeper application test environment" on page 4-11. The other tests in the flow are performed in cooperation between the service provider and the operator.

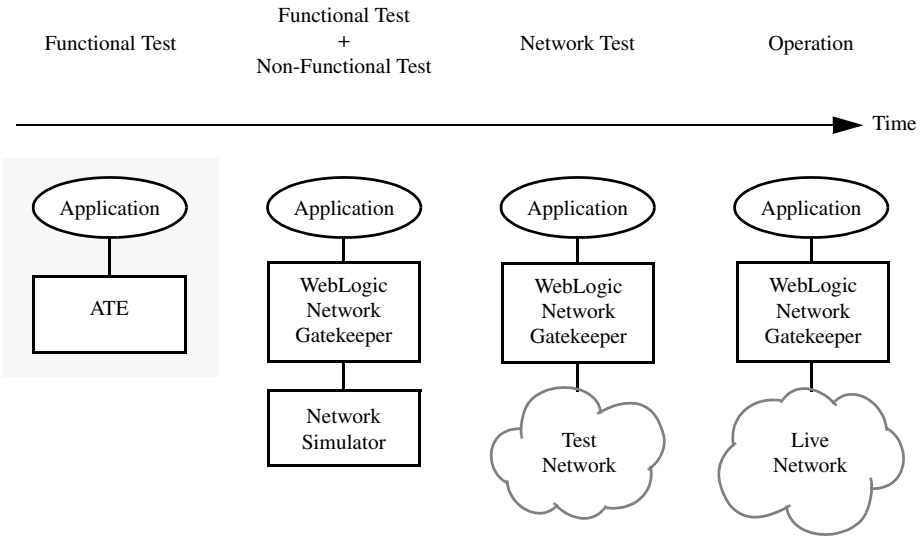


Figure 4-3 Application test flow

Network Gatekeeper application test environment

The Network Gatekeeper Application Test Environment (ATE) is an interactive, graphical test environment where applications to be connected through WebLogic Network Gatekeeper can be tested. The current version of the ATE supports messaging and user location.

Interactive graphical interface

The WebLogic Network Gatekeeper ATE interface is based on a GUI where a map is displayed. The map can be changed to fit different geographical areas. Through the GUI it is possible to add

mobile terminals. These terminals are given a phone number. When the terminal has been defined, it can be moved around the map to simulate to different locations.

It is also possible to send and receive messages through the terminals in the GUI.

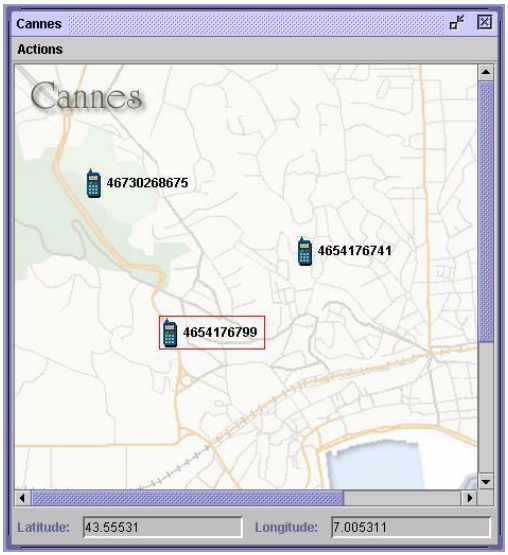


Figure 4-4 Network Gatekeeper ATE GUI

Testing applications

Applications uses the WebLogic Network Gatekeeper ATE web service end point during test. After successful verification, the end point is changed to the web service end point of the WebLogic Network Gatekeeper. See Figure 4-5, “WebLogic Network Gatekeeper ATE in relation to WebLogic Network Gatekeeper,” on page 4-13.

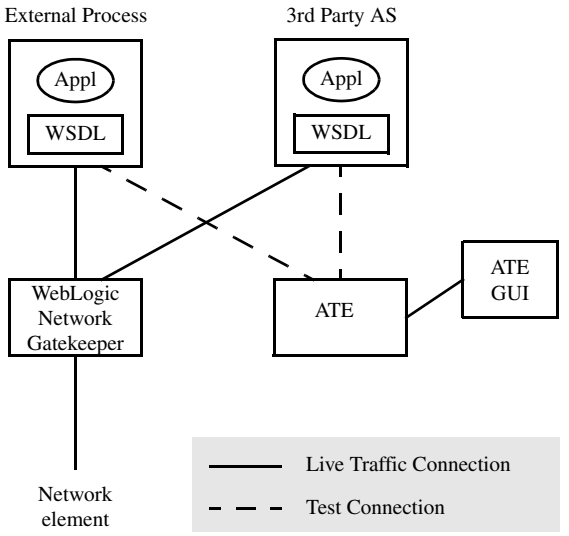


Figure 4-5 WebLogic Network Gatekeeper ATE in relation to WebLogic Network Gatekeeper

CHAPTER 5

Service Provider and Application Administration

The following sections describe tools for managing service providers and applications:

Overview

All service providers and their applications have to be registered in the WebLogic Network Gatekeeper. The service provider and application registration is performed through the WebLogic Network Gatekeeper management tool or using management systems integrated with the WebLogic Network Gatekeeper through Network Gatekeeper Partner Management Tools.

The applications are made aware of the WebLogic Network Gatekeeper's location using addressing information in the Web Services WSDL files.

For information related to the security in the communication between the WebLogic Network Gatekeeper and the applications, see Chapter 12, "Security".

Administration model

The main concepts in the administration of service providers and applications are accounts, account groups and Service Level Agreements (SLAs).

All service providers need to have a registered account in the WebLogic Network Gatekeeper. To support tiering or grouping of service providers, the accounts are connected to account groups and each group is associated with an SLA.

When a service provider account has been created, application accounts can be registered on that service provider account. As for the service provider accounts, the application accounts are connected to account groups and these groups are associated with SLAs.

In order to manage applications executing in locations outside the service provider's domain, for example office applications executing on the end users' PCs at customer organisations, it is possible to create application instance groups. The application instance group identifies the customer organisation. An application instance group SLA is used to regulate how many application instances can be active at the same time.

Applications executing within the service provider's domain use a one-to-one relation between the application account, the application instance group, and the number of allowed instances in the application instance group.

The administration model is shown in Figure 5-1, "Service provider and application administration model," on page 5-3.

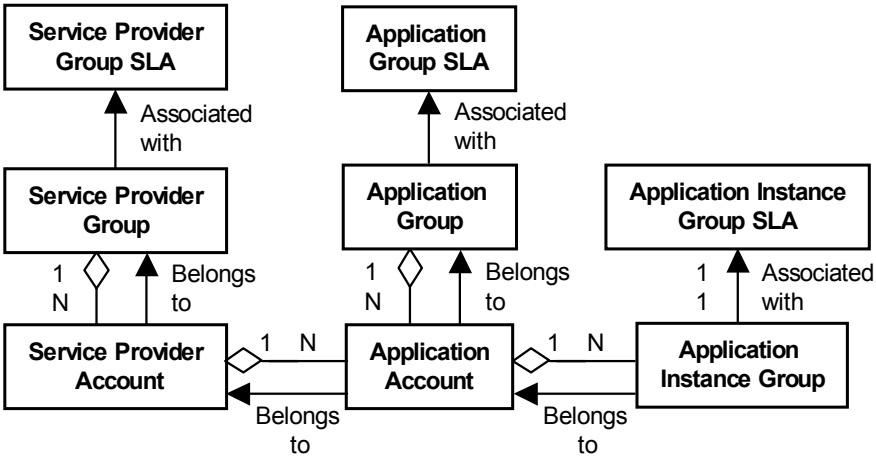


Figure 5-1 Service provider and application administration model

The SLAs on service provider and application level regulates, for example, available service capabilities, maximum allowed bandwidth and allowed number of concurrent sessions. It also specifies access to charging capabilities and revenue sharing schemas.

The network protection and and access control functionality in WebLogic Network Gatekeeper, introduces two traffic contract types and a set of rules to enforce these new contracts. The first traffic contract type regulates the relation between a service provider group and the network nodes, that is the service provider traffic SLA, and the second regulates the relation between WeLogic Networ Gatekeeper and the network nodes, that is the total traffic SLA.

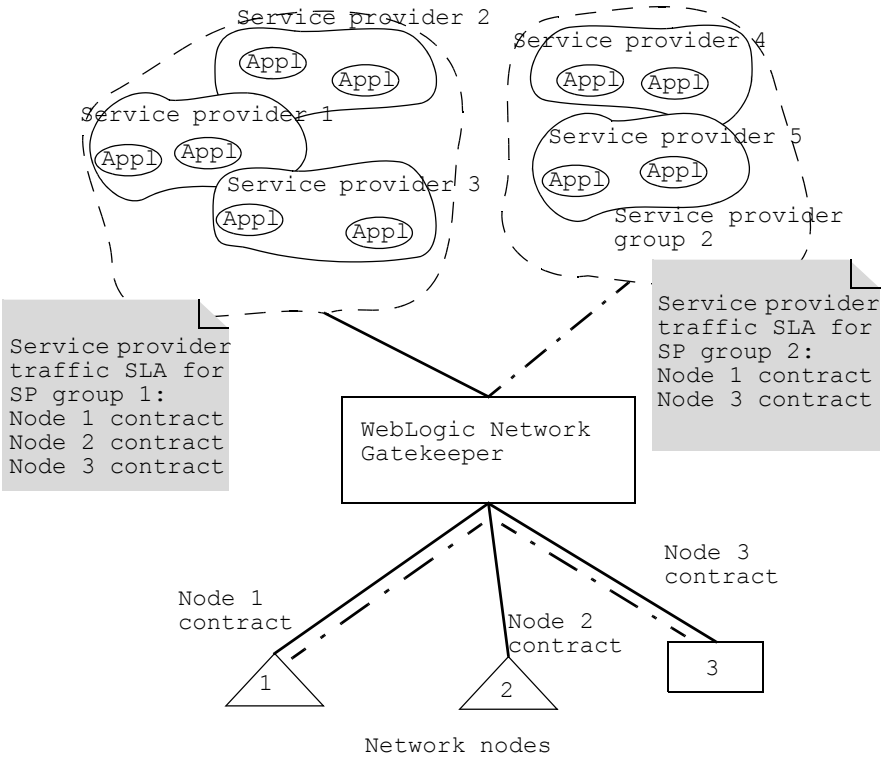


Figure 5-2 Example of service provider traffic SLAs in the WebLogic Network Gatekeeper

In the example above, service providers in service provider group 1 are allowed to access all network nodes, since their service provider traffic SLA (valid for all service providers within the group) contains node contracts for all nodes.

Service providers in service provider group 2 are only allowed to access network node 1 and 3, because their service provider traffic SLA only contains node contracts for node 1 and 3.

Partner Management Tools

WebLogic Network Gatekeeper Partner Management Tools support automation of traditionally work intensive task related to service provider and application administration. The tools are used for shifting most of the registration work to the service provider. The operator's role changes from

entering registration data to approving registration data. Using WebLogic Network Gatekeeper Partner Management Tools facilitate the administration of large numbers of service provider and application accounts and prevents an increasing administration overhead. It also provides the service providers with a defined and structured channel to communicate desired account changes and to retrieve usage statistics for the accounts.

The Partner Management Tools support the service provider and application administration model described above.

Service provider tasks

Service providers can perform the following tasks through Partner Management Tools:

- Register as a service provider
- Log in
- Add/view/update/delete application accounts
- Activate/deactivate application accounts
- View/update/delete the service provider account

Operator tasks

The operator can perform the following tasks through Partner Management Tools:

- Add/view/update/delete service provider groups
- Add/view/update/delete application groups
- Add/view/update/delete application instance groups
- Add/view/update/delete service provider accounts
- Approve/reject service provider account applications
- Activate/deactivate service provider accounts
- Add/view/update/delete application accounts
- Approve/reject application account applications
- Activate/deactivate application accounts
- Add/view/update/delete white lists and white list entries on a service provider level

- Add/view/update/delete white lists and white list entries on a global level

Statistics

For descriptions on available statistic types, see *Statistics and Performance Data* on page 1.

Operator intranet, extranet and CRM/PRM system interfaces

The Partner Management Tools provide modular Web Services and CORBA based interfaces. This gives the operator the freedom to incorporate the Partner Management Tools into CRM/PRM systems, intranets and extranets. The following interfaces exists:

Service provider interfaces:

- Service provider registration callback - used for recieving registration results
- Service provider log in
- Service provider - used for the administration of the service provider’s accounts

Operator interfaces:

- Operator log in
- Operator - used for the administration of all accounts

Common interfaces:

- Alarm listener - receives alarms from the WebLogic Network Gatekeeper
- Notification listener - receives notifications from the WebLogic Network Gatekeeper
- Event listener - receives events from the WebLogic Network Gatekeeper
- Listener manager - administrates the listeners

For example, a service provider can register on the operator's web site. When the operator has approved the account, the service provider can log in to the operator’s extranet and register applications and view account data. The operator handles all verification and approval of the registered applications over the intranet.

Back end system interfaces

The Partner Management Tools can be integrated with back end systems and network nodes as, for example SMSCs, MMSCs and pre-paid systems to create and update accounts. The integration is made through a plug-in interface similar to the plug-in interface used for the traffic towards the network nodes..

Application connection

Applications using Web Services need access to the Extended API or Parlay X WSDL files. The connection is set up through a SOAP/HTTP connection with WebLogic Network Gatekeeper, see Figure 4-1, “WebLogic Network Gatekeeper application interfaces,” on page 4-3.

Other administration areas

For an application using the charging (content based) or messaging services, messaging or charging accounts have to be created.

If the application use the user interaction service, announcements have to be recorded and installed in the network. For more information on these areas, see User’s Guide - WebLogic Network Gatekeeper.

CHAPTER 6

Network Interface

The following sections describe the WebLogic Network Gatekeeper network interface:

Overview

The WebLogic Network Gatekeeper network interface consists of a network routing function and a set of network plug-ins. The routing function routes service requests to the correct service node or OSA gateway, depending on the requests address and/or address format. The plug-ins transform the service request to fit the protocol used by the specified network element.

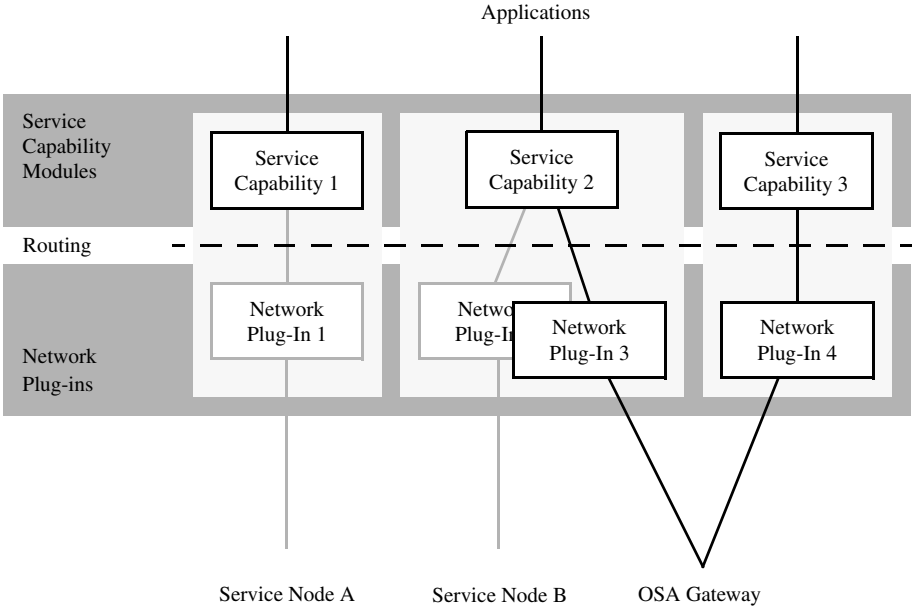


Figure 6-1 Plug-in and routing overview (Web Services APIs not shown)

Network routing

The network routing function is used for routing network requests from the service capabilities modules to the appropriate network plug-ins. Based on the request's address plan, actual destination address and the network plug-ins' load levels, it provides the service capability modules with an appropriate network plug-in.

First, the plug-ins that support the address plan, for example E.164, are identified. Then, the routing function matches the address provided in the request with the address patterns supported

by the plug-in. If more than one plug-in with normal load level matches the selection criterion, the routing function selects a plug-in instance according to a round robin schema.

The network routing function can take the underlying nodes' load and capacity into account when selecting an appropriate plug-in. For more information, see "Plug-in manager policy execution" on page 7-6.

Network triggered events

For a network triggered event, the plug-in is provided with a service capability module instances reference according to a round robin schema. The plug-ins transform the protocol specific requests to fit the service capability modules' protocol independant plug-in interfaces.

The WebLogic Network Gatekeeper takes care of network initiated events that no application has registered for. For example, in the call control case the following options are available:

- Route the call to original destination
- Release the call

In the messaging case the following options are available:

- Send a configurable reply message
- Drop the message

Core network plug-ins and supported protocols

A list of supported protocols for each service capability can be found in "Technical Specification" on page C-1.

Plug-ins for extended protocol support

The concept with network plug-ins makes it easy extend the system to support new nodes/protocols and upgraded protocol versions. For more information, See Chapter 13, "Service Extensibility".

CHAPTER 7

Policy

The following sections describe WebLogic Network Gatekeeper policy functionality:

Overview

The WebLogic Network Gatekeeper's behaviour in relation to the service providers, applications and the networks can be easily changed to fit an operator specific context. This is achieved using rule based policies together with a mix of internal and external data sources.

The WebLogic Network Gatekeeper distinguish between core and custom policies. The core policies are general policies that are a standard part of the WebLogic Network Gatekeeper. Their main usage areas are authentication, authorisation, service capability access, and usage control.

Custom policies are operator specific policies defined by the operator and implemented by BEA Systems or a selected partner.

Core policy areas

Authentication

Authentication policies are applied on service provider and application level. They define the configurable parts of the authentication process between an application and WebLogic Network Gatekeeper. Using these policies, the authentication process can be customized to fit the relation between the operator and the service provider.

Authorisation and access control

Access control policies are applied on service provider and application level and defines both which service capabilities and which functions within a service capability an application is allowed to use. Using the policies the operator can provided customized service offerings depending on the applications' needs and the service provider's trustworthiness.

The service provider's and application's Service Level Agreement (SLA) is used to specify which service capabilities and service capability functions the application is allowed to use.

Service capability usage/quality of service

Usage policies can be applied on both service provider and application level. They make it possible to specify differentiated Quality of Service (QoS) offerings for the service providers and their applications.

On service provider level these policies define both a guaranteed and a maximum number of requests an service provider is allowed to send through a specific service capability during a specified time period.

On application level these policies define the maximum number of active sessions or maximum number of requests as well as a guaranteed number of requests an application is allowed to send through the service capability during a specified time period.

Network protection and access Control

Access control

Service requests from the applications are given access to individual network nodes depending on the service provider owning the application.

It is possible to maximize the total request rate for a service provider's applications towards individual network nodes.

All service requests from WebLogic Network Gatekeeper to an individual network node can be barred/unbarred through a single operation.

Traffic throttling/overload protection

The maximum request rate from WebLogic Network Gatekeeper towards each individual network nodes can be specified.

For network nodes with the ability to report their current load, WebLogic Network Gatekeeper can dynamically adapt the amount of traffic towards the individual node depending on the node's load status. Two overload levels can be defined:

- Overloaded
- Severely overloaded

In the overloaded situation, only traffic from prioritized service providers are let through. If the severely overloaded situation occurs, all request sending towards the node is stopped and the severely overloaded is reported back to the requesting applications.

Node load and delay reduction

It is possible to reduce the load on the underlying nodes through barring excessive send lists. The maximum send list size is configurable on individual node level. Send lists exceeding the specified limit will be rejected. The feature supports the following service capabilities:

- Messaging
- User location

- User status

Unexpected network initiated event handling

Weblogic Network Gatekeeper handles unsubscribed network initiated events. The following options are available in the call control case:

- Route the call to the original destination
- Play a customized announcement
- Release the call

The following options are available in the messaging case:

- Send a customized reply message
- Drop the message

Quality of service assurance

Traffic from high priority service providers always have precedence before traffic from low priority service providers to assure the guaranteed amount of traffic.

Other policies areas

Policy is also used in the following areas:

- Charging (for example service provider and application specific service code insertion for rating purposes)
- Subscriber privacy (for example in user location)

Rule based policy execution

At start up, all policy rules are automatically imported to the rule engine's working memory. When policy rules are added or updated, these are imported to the rule engine using the WebLogic Network Gatekeeper Management Tool.

The rule based policies are triggered at Policy Enforcement Points (PEPs). PEPs are implemented in the service capability modules. At the PEPs, a request to use a service capability can be rejected or allowed. The decision whether the request should be rejected or allowed is taken by the Policy Decision Point (PDP) in the rules engine. To take the policy decision, the PDP needs the request

data and the rule associated with the PEP. The policy is then enforced at the PEP. An overview is shown in Figure 7-1, “Policy execution flow for application initiated request,” on page 7-5.

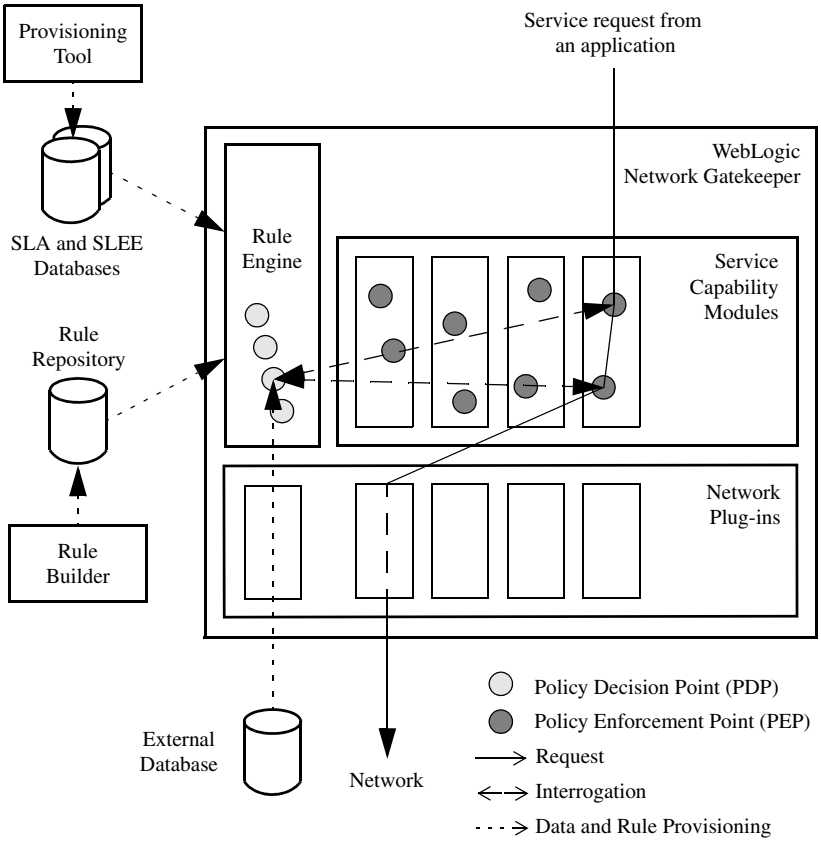


Figure 7-1 Policy execution flow for application initiated request

Basic request data is always provided by the PEP. Additional data needed for the rule engine’s decision can be provided by the SLA database, the SLEE database, and/or from other external databases.

Together, the data and the rules make it possible for the PDP to take a policy decision. The decision is used by the PEP and the request is rejected or allowed depending on the decision. In

addition, the PDP can modify the incoming request data or add new data based on the rules used. The new or modified data is returned to the PEP together with the policy decision.

Plug-in manager policy execution

To peform network protection and network access control, the Policy Enforcement Point (PEP) in the plug-in manager is used.

The policy execution flow through the plug-in manager is shown in Figure 7-2, “WebLogic Network Gatekeeper policy execution flow in an application initiated service request flow,” on page 7-7.

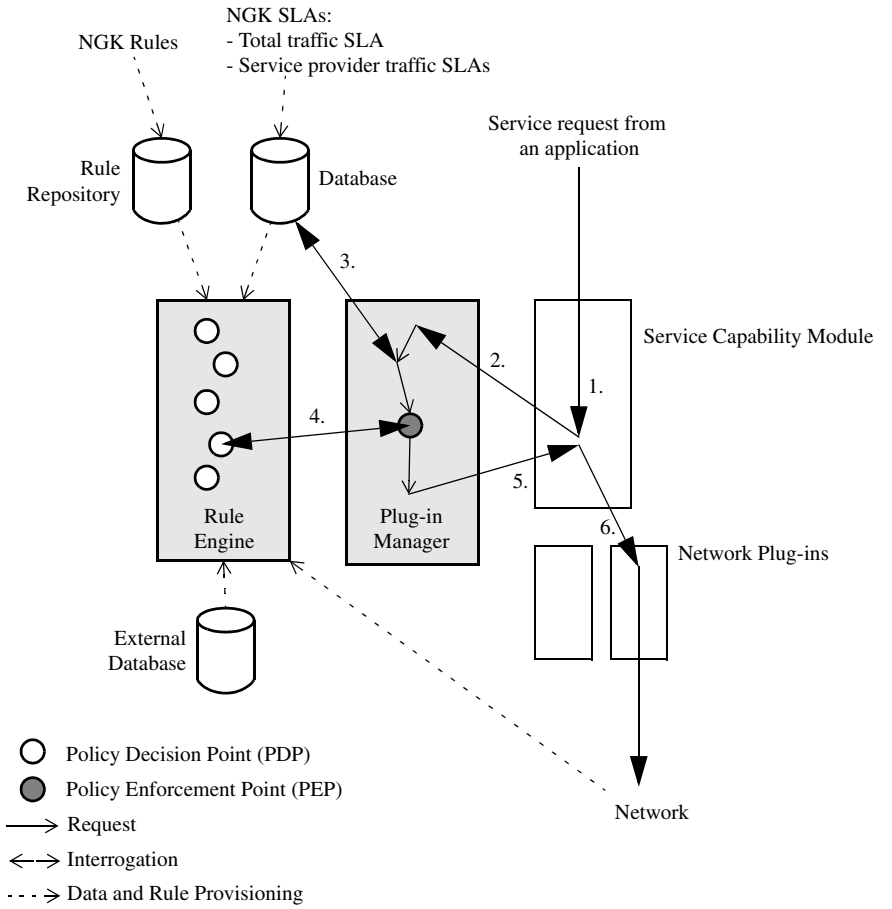


Figure 7-2 WebLogic Network Gatekeeper policy execution flow in an application initiated service request flow

Below follows a description of the most important steps in the policy execution flow shown in Figure 7-2, "WebLogic Network Gatekeeper policy execution flow in an application initiated service request flow," on page 7-7.

1. A service capability module receives a service request from an application.

2. The service capability module requests a network plug-in for the service request.
3. The plug-in manager retrieves a list of possible plug-ins based on the address plan and destination address indicated in the service request.
4. The plug-in manager sends the list of possible plug-ins and the service request data to the rule engine for a policy decision. The rule engine makes a policy decision based on the following:
   - the request data
   - the network gatekeeping rules
   - the specified data in the service provider traffic SLA
   - the specified data in the total traffic SLA
   - additional data provided by external databases or by the network node.

   The rule engine returns a list of the plug-ins that have passed the policy decision. Also, the plug-in manager is informed if no plug-in has passed.
5. The plug-in manager selects one plug-in from the list and sends it to the service capability module.
6. The service capability module routes the service request to the selected plug-in.

CHAPTER 8

Charging

The following sections describe WebLogic Network Gatekeeper charging functionality:

Overview

The WebLogic Network Gatekeeper makes it possible to tailor the type of charging to be used for each application using service capabilities through the WebLogic Network Gatekeeper. An application can use one or more of the following charging alternatives:

- Charging based on time period or used amount (CDR based charging)
- Charging based on the content or value of the used service

The CDR data can be stored in the WebLogic Network Gatekeeper's internal charging database or retrieved in real-time by billing and post processing systems through a billing gateway. Both pre-paid and post-paid accounts are supported.

CDR based charging

CDRs are used for charging based on time period or used amount. Charging based on time period is typically used to charge for calls. Amount can be used, for example to charge for a positioning service.

Data generation

Charging data is generated every time an application uses service capabilities through a service capability module. The charging data is recorded by the service capability module while the application interacts with the network. When the interaction is closed, the service capability module stores the charging data as a CDR in the database. One storage of data in the database represents a charging transaction. (If integrated with a billing gateway, the charging data is sent directly to the billing gateway.)

In the database, all CDRs are stored in the same table. Different service capability modules produce different kinds of charging data. Therefore, some of the fields in the charging table are optional and others may contain different types of data depending on the service capability module. General descriptions of the fields in the charging table are provided in Table 8-1 on page 3. The service capability module specific charging data descriptions are provided in "Charging Data" on page A-1.

The CDRs can also be used to create extensive service usage statistics.

Table 8-1 CDR Contents

Field	Description
transaction_id	A unique ID for the transaction. Automatically provided.
session_id	An ID connecting related transactions within a service capability module. For example, a call containing three call-legs will produce three separate transactions within the same session.
service_name	The service capability module that has created the CDR.
user_id	The application that used network services through the service capability module.
start_of_usage	The date and time the service capability module started to use services in the network.
connect_time	The date and time the destination party responded. Used for call control only.
end_of_usage	The date and time the service capability module stopped to use services in the network.
duration_of_usage	The total time the service capability module used the network services.
amount_of_usage	The used amount. Used when the charging is not time dependent. For example, flat rate services.
originating_party	The originating party's address.
destination_party	The destination party's address.
transaction_part_number	If the transaction is divided into sub-transactions (transaction parts), the part number indicates number of the sub-transaction. If sub-transactions are not used, transaction part number is equal to 1.

Field	Description
completion_status	Indicates the completion status of the transaction. 0 - failed 1 - completed 2 - partial Partial is only used if the transaction is divided into sub-transactions. 3 - completed but result delivery to application failed
additional_info	Any additional charging data specified by the service capability module. The data is tagged for XML processing.
service_provider	The ID of the service provider that hosts the application.
revenue_share_ percentage	A percentage to be used for revenue sharing. Added by the application or by the policy service.
party_to_charge	The ID of the party that initiated the request that resulted in the transaction.
charging_info	Service code added by the application or by the policy service.
slee_instance	The SLEE instance hosting the service capability module that generated the CDR.

Content based charging and accounting

Content or value based charging makes it is possible to charge a user or subscriber based on the value of a used service rather than the amount or time used. This can be used, for example when down loading music video clips or in m-commerce applications. Both pre-paid and post-paid end-user accounts are supported.

Revenue sharing

Revenue sharing means that part of the revenues generated by an application can be shunted back to the service provider. The actual percentage is specified in the CDR.

For pre-paid accounts, settlement can be handled in real-time by the charging service or near real-time through a billing gateway.

For post-paid accounts, settlement is handled by CDRs and post processing systems.

Billing system integration

CDR database

Integration through the CDR database is suitable when the applications connected to an WebLogic Network Gatekeeper use post-paid accounts.

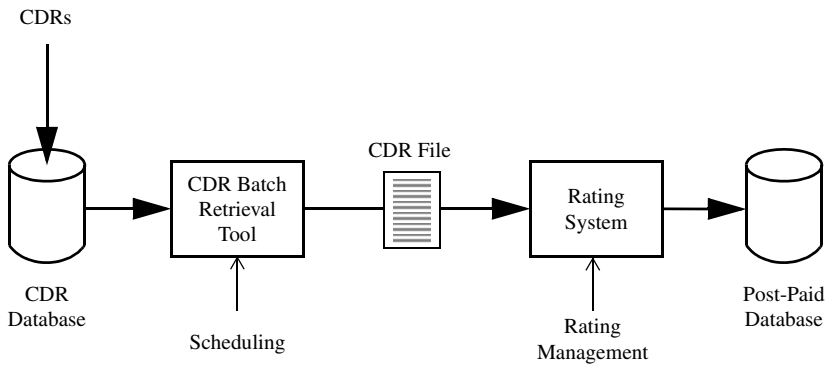


Figure 8-1 Billing integration through CDR database

When integrating through the CDR database, a CDR batch retrieval tool retrieves the CDRs from the database and stores them in a file format. The CDR file is processed by a rating system that transforms the CDR to billing information and stores it in a post-paid accounts database. The flow is shown in Figure 8-1, “Billing integration through CDR database,” on page 8-5.

Billing gateway and other similar

Integration through a billing gateway supports real-time settlement of pre-paid account as well as storing billing information in a post-paid database.

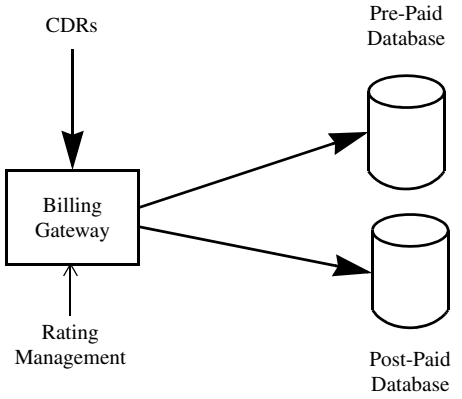


Figure 8-2 Billing integration through billing gateway

When integrating through a billing gateway, the billing gateway retrieves the CDRs in real-time through a CORBA based charging listener. Rating, rating management, billing information storage, and pre-paid accounts settlement are handled by the billing gateway. The flow is shown in Figure 8-2, “Billing integration through billing gateway,” on page 8-6.

Charging plug-in

Applications using the charging service capability module need to have a connection with an accounts database. The integration between the charging service capability module and the database is made through a customized plug-in. The plug-in is developed to fit the characteristics of the accounts database. With this solution, both pre-paid and post-paid accounts can be handled. For example, it can be verified before a call is set up that there is money are on a pre-paid account.

The flow is shown in Figure 8-3, “Billing integration through a charging plug-in,” on page 8-7.

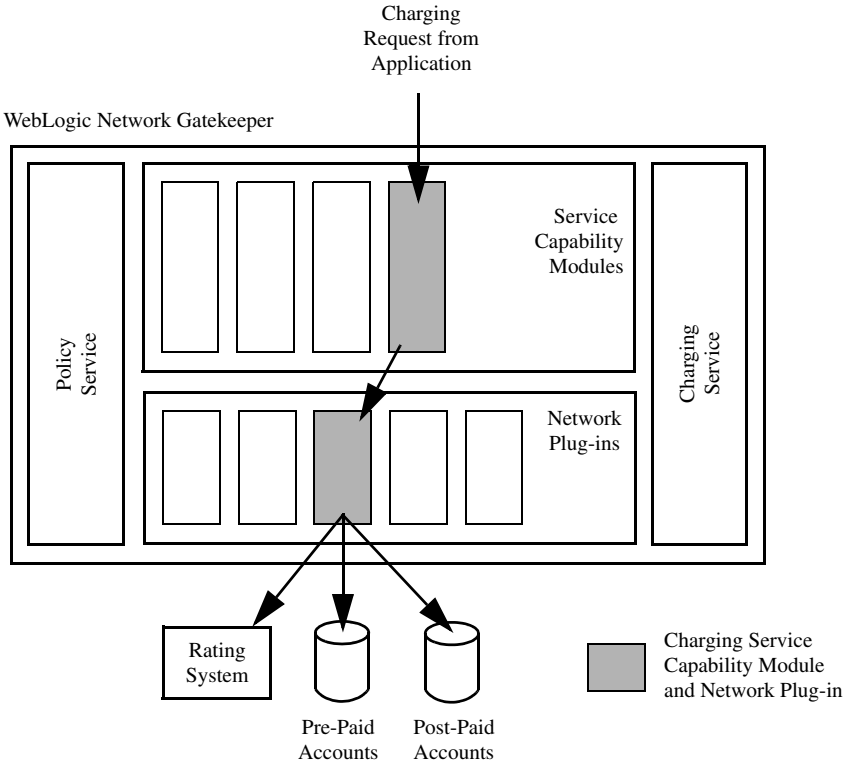


Figure 8-3 Billing integration through a charging plug-in

CHAPTER 9

Operation, Administration and Maintenance

The following sections describe WebLogic Network Gatekeeper Operation, Administration, and Maintenance (OA&M) functionality:

Overview

WebLogic Network Gatekeeper features a management tool with both a graphical and a text based interface. It supports local and remote management as well as integration with other management systems through CORBA Interface Definition Language (IDL) and SNMP.

Management users can be divided into user groups with access to different parts of the management. Within the user groups, individual users can have different levels of access, see "Management security" on page 12-5.

PRM and CRM systems can be easily integrated with the WebLogic Network Gatekeeper using WebLogic Network Gatekeeper Partner Management Tools.

Management tool

The management tool can be installed on both Windows 2000/XP and UNIX machines. It connects to WebLogic Network Gatekeeper through a CORBA connection.

Graphical interface

The graphical interface displays all SLEEs and all SLEE services installed in the WebLogic Network Gatekeeper, see Figure 9-1, "WebLogic Network Gatekeeper management tool graphical interface," on page 9-3. It is easy to add SLEEs to the management tool as the system is extended. The SLEE concept is explained in Chapter 15, "Software Architecture Overview".

The management tool supports the development and installation of specialized Graphical User Interfaces (GUIs). That is, GUI plug-ins to support operator specific tasks. For repetitive tasks, as the generation of large numbers of accounts, batch files can be created and processed through the management tool.

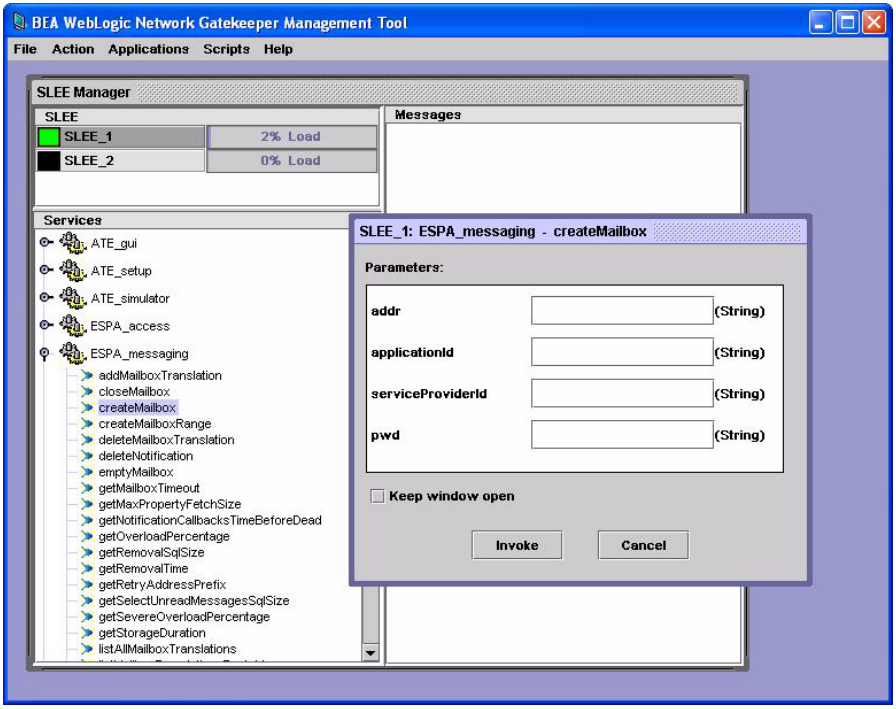


Figure 9-1 WebLogic Network Gatekeeper management tool graphical interface

Text based interface

The text based interface is a command line interface providing the same basic functions as the graphical interface. The main benefit with the text based interface is that it can be used to manage the WebLogic Network Gatekeeper over a remote connection such as telnet.

OAM tasks overview

For the WebLogic Network Gatekeeper, the following main OAM areas are supported:

- Application and service provider administration
- SLEE and SLEE service operation
- User administration
- Statistics handling

- Charging data export
- Alarm and event administration
- Routing administration
- User interaction announcement administration
- Service extension
- System backup and restoration
- System upgrade
- Alarm and fault handling

Complete information about the WebLogic Network Gatekeeper OAM can be found in User's Guide - WebLogic Network Gatekeeper.

OSS and PRM/CRM integration

OSS

The whole or selected parts of the management application can be integrated with external Operation Support Systems (OSS) through a CORBA IDL or SNMP interface. Alarm supervision systems can add alarm and event listeners to the WebLogic Network Gatekeeper alarm and event services. SNMP traps are sent to the registered SNMP mangers.

PRM/CRM

The WebLogic Network Gatekeeper provides a Web Service and CORBA IDL based interface, the Partner Management Tools, for the administration of service providers and their applications. This interface is used for integrating WebLogic Network Gatekeeper with Partner Relationship Management (PRM) and Customer Relationship Management (CRM) systems.

The Partner Management Tools is a common access point where registration data, service contract data and XML-based Service Level Agreements (SLAs) are received, processed and distributed to the affected parts of the WebLogic Network Gatekeeper.

For more information about the Partner Management Tools, the tasks they support, and the service provider and application administration model, see Chapter 5, "Service Provider and Application Administration".

CHAPTER 10

Alarm and Event Handling

The following sections describe WebLogic Network Gatekeeper alarm and event handling:

Overview

All the WebLogic Network Gatekeeper software modules can raise alarms and events. The difference between alarms and events is that alarms are unexpected and might need corrective action, while events are expected and of importance to the operator.

Alarm handling

All alarms are stored in the alarm list in the WebLogic Network Gatekeeper database. The type of information stored for each alarm is presented in Table 10-1 on page 2.

Table 10-1 Alarm Entry Information

Information	Description
Unique ID	A sequential number to identify the alarm.
Source	Specifies the name of the SLEE service (software module) that raised the alarm and the IP address of the SLEE the service is installed in.
Severity	Specifies the alarm's severity level. One of the following: 1 - warning 2 - minor 3 - major 4 - critical The individual alarms' severity level can be reconfigured by the operator.
Identifier	Specifies the alarm type using a number and a heading.
Info	Alarm information provided by the software module the raised the alarm.
Timestamp	Specifies the date and time the alarm was raised.

All alarms have a default severity level.

It is possible to set an alarm filter specifying an alarm severity level. Only alarms with a severity level equal to, or higher than the specified filtering level will be written to the alarm list.

Alarms are retrieved from the alarm list based on software module, severity level, and time period. The different alarm types and recommended corrective actions are presented in User's Guide - WebLogic Network Gatekeeper.

Event handling

All events are logged in the event log in the WebLogic Network Gatekeeper database. The type of information stored for each event is presented in Table 10-2.

Table 10-2 Event Log Entry Information

Information	Description
Unique ID	A sequential number to identify the event.
Source	Specifies the name of the SLEE service (software module) that raised the event and the IP address of the SLEE the service is installed in.
Level	Specifies the event's importance level. One of the following: 1 - low 2 - medium 3 - high
Identifier	Specifies the event type using a number and a heading.
Info	Event information provided by the software module the raised event.
Timestamp	Specifies the date and time the event was raised.

It is possible to set an event filter specifying an importance level. Only events with a importance level equal to, or higher than the specified filtering level will be written to the event log.

Events are retrieved from the event log based on software module, importance level, and time period.

OSS integration

CORBA

An Operation Support System (OSS) integrates with the WebLogic Network Gatekeeper alarm and event services through registration of alarm and event listeners over a CORBA/IDL interface.

SNMP

In addition, the WebLogic Network Gatekeeper supports sending of alarms as SNMP traps to SNMP managers. The alarms sent to the SNMP managers can be filtered on alarm severity.

CHAPTER 11

Statistics and Performance Data

The following sections describe the statistics and performance data that WebLogic Network Gatekeeper collects:

Overview

Both performance data and usage statistics can be retrieved from the WebLogic Network Gatekeeper. The usage statistics makes it possible to retrieve statistics on the usage of service capabilities through the WebLogic Network Gatekeeper. For a specified time period, usage statistics be retrieved based on the following:

- service provider
- application
- SLEE
- all SLEEs

The performance data keeps track on the system load when it come to the SLEE and the JVM.

Usage statistics

The statistics functions measure the usage of the service capabilities through the WebLogic Network Gatekeeper. A number of statistics types have been defined. One or more types are defined for each service capability, see Table 11-1, “Statistics types per service capability,” on page 11-2.

Table 11-1 Statistics types per service capability

Service capability	Statistics types
Call	Call control: • Number of network initiated calls • Number of application initiated calls Multiparty call control: • Number of network initiated call legs • Number of network initiated additional call legs • Number of application initiated call legs • Number of application initiated additional call legs
Charging	• Number of immediate charging transactions • Number of reservation/payments in parts charging transactions

Service capability	Statistics types
Messaging	SMS: • Number of sent messages • Number of received messages MMS: • Number of sent messages • Number of received messages
Subscriber Profile	• Number of set property requests • Number of get property requests
User Interaction	Call user interaction: • Number of played announcements • Number of played voice prompts • Number of recorded messages Generic user interaction: • Number of sent messages with response request • Number of sent messages without response request • Number of received messages
User Location	Number of location requests
User Status	Number of status requests

Customized statistics reports can be generated for the system and for individual service providers and/or applications. For statistics generation options, see Table 11-2 on page 4.

Table 11-2 Statistics generation options (one SLEE, all SLEEs, service provider or application)

Statistics Area	Parameter
System statistics	• The whole system or for an individual SLEE (The SLEE concept is explained in Chapter 15, "Software Architecture Overview".) • All statistics types or for a single statistics type • A specified time period or not
Service provider and/or application statistics	• All the service provider's applications or a single application • All statistics types or for a single statistics type • A specified time period or not

A pre-defined report type is the weekly report. It shows the total service capability usage hour by hour during a specified week. The weekly report also shows total usage for each day and the average transaction rate (transactions/second (tps)) during the busy hour of each day. A busy hour is defined as the 60 minutes during which the largest number of transactions are handled. That is, any 60 minute interval can be the identified as the busy hour. For example: 10:05:00-11:05:00.

The statistics data is stored in the database and can be retrieved and printed to file or viewed using the management tool.

Performance data

The load is measured separately for each server in the WebLogic Network Gatekeeper system. By monitoring the load on the individual servers and balancing the load among the servers accordingly, system performance is kept at its maximum for every system configuration.

Detailed resource utilization data can be viewed for the whole SLEE, the JVM and the defined load share contexts. (Load share contexts are described in Chapter 15, "Software Architecture Overview".) The resource utilization data includes:

- current load
- heap used
- heap total
- heap initial

- heap last GC
- And for each load share context:
  - task pool size (threads)
  - task pool used (threads)
  - task queue size (tasks)
  - task queue used (tasks)
  - orb pool size (threads)
  - orb pool used (threads)

CHAPTER 12

Security

The following sections describe WebLogic Network Gatekeeper security:

Overview

The security areas covered by the WebLogic Network Gatekeeper are related to the interworking between the WebLogic Network Gatekeeper and the applications, data storage and management security.

Identification and authentication

Web Services

All applications, that is application instances in application instance groups (see *Service Provider and Application Administration* on page 1) accessing the WebLogic Network Gatekeeper through the Web Services interfaces uses a Kerberos type of service token-based authentication.

The application instance group is provided with a user name (the application instance group ID) and a password. When an application instance in the application instance group wants access to the WebLogic Network Gatekeeper the application instance logs in using the user name and password together with the application account ID and service provider ID to retrieve a service token.

This mechanism can be extended as an option using, for example Passport or other extended Kerberos Key Distribution Centre (KDC) solutions.

Other integrated interfaces

If an integrated interface like CIMD or SMPP is exposed to applications through the WebLogic Network Gatekeeper, that interface's authentication mechanism will be used. Such mechanisms are typically based on a user name and password combination.

Authorisation and service access

Web Services

At log in, the WebLogic Network Gatekeeper verifies that the application instance group's current number of active service sessions is less than the maximum number of allowed concurrent service sessions specified in the application instance group's SLA. If less than the specified maximum, a service token is sent to the application instance. In the case when the number of concurrent service sessions equals the maximum number, the oldest service session is terminated before the service token is sent to the application instance.

For every service request, the application instance includes the service token in the SOAP request's header. Before a service request is accepted, policy is used to verify that the request fulfils the criteria specified in the service provider SLA and in the application SLA.

Other integrated interfaces

Integrated interfaces make use of the same authorisation and service access control mechanisms as described above. That is, policy is used to verify that the service requests fulfil the criteria specified in the service provider SLA and in the application SLA.

SLA data

The following data is specified in the SLAs:

SLA Level	Criteria
Service Provider (Enterprise Operator)	Traffic and charging related data and destination address black- and white-lists. For example, what network capabilities are available for the service providers in the group, maximum bandwidth available. It also specifies access to charging capabilities and revenue sharing schemas.
Application	Traffic and charging related data and destination address black- and white-lists. For example, what network capabilities are available for the applications in the group, maximum bandwidth available. It also specifies access to charging capabilities and revenue sharing schemas.
Application Instance (Group)	The maximum number of allowed concurrent sessions.

Integrity and confidentiality

To guarantee the integrity and confidentiality in the communication between the WebLogic Network Gatekeeper and the application, the communication can be encapsulated and encrypted using SSL (Web Services) or a Virtual Private Network (VPN) (Web Services and integrated interfaces).

The VPN is a firewall to firewall connection, with VPN routers attached to the WebLogic Network Gatekeeper and the application server, see Figure 12-1, "WebLogic Network Gatekeeper to application VPN connection," on page 12-4.

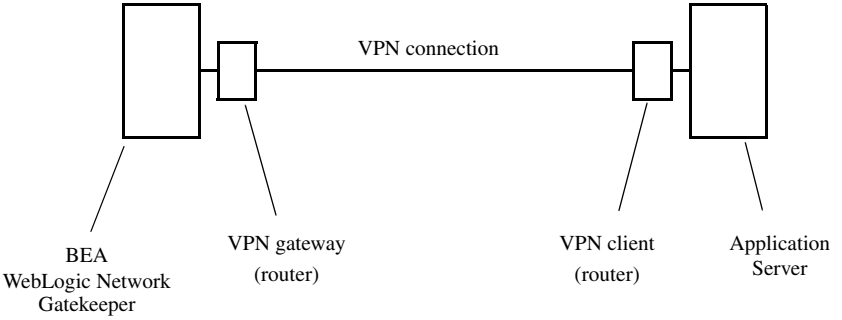


Figure 12-1 WebLogic Network Gatekeeper to application VPN connection

Auditing and non-repudiation

Both successful and unsuccessful log in attempts generate events that are written to the event log and all transactions are stored as CDRs in the database. The policy service generates alarms when service requests are denied due to SLA violations.

Network authentication

The WebLogic Network Gatekeeper protocol plug-ins authenticate with the underlying network nodes if the protocol provides an authentication interface.

Database security

Access to the WebLogic Network Gatekeeper database (see *Database* on page 9) is protected by host address, user name and password combinations. The different SLEE service database users, have access to a limited set of database tables, that is, only tables related to the service. In addition, SLEE service database users cannot grant additional privileges to themselves or other users, other than for the service's own tables.

Sensitive data, as user and database passwords as well as user certificates and private keys are encrypted before being stored in the database.

Management security

User and password

To work with OAM through the WebLogic Network Gatekeeper management tool, an administrative user needs user name and a password is required. The user names and passwords are encrypted before stored in the WebLogic Network Gatekeeper database.

The user name and password are used when logging in to the management tool and when sending commands from the management tool towards WebLogic Network Gatekeeper.

Access levels and administration groups

At registration all administrative users are provided with an access level. The access level is one of the following:

• Read only	A read only user can only read registered data.
• Standard read and write	A standard read and write user can read all types of data but only set non critical data.
• Administrator	An administrator user can read and set all types of data including user accounts.

To limit the users' access to the different parts of the platform logical administration groups can be created. The groups are created by the operator to fit the operators OAM organization. One group consists of one or more logically related software modules.

The administrative users are connected to one ore more of these administration groups depending on their responsibilities. A user maintains the same access level through out all groups he or she is connected to.

All OAM work is performed through the WebLogic Network Gatekeeper management tool’s graphical or text based interface.

CHAPTER 13

Service Extensibility

The following sections describe how to extend WebLogic Network Gatekeeper functionality:

Overview

Extensibility is about extending the WebLogic Network Gatekeeper's functionality to support new APIs, service capabilities and network protocols. That is, new APIs and/or network plug-ins can be added to existing service capability modules. In addition, support for a whole new service capability, including API, service capability module, and protocol plug-in can be provided.

To support service extensibility, BEA systems provides a service extension toolkit. This toolkit simplifies the creation of new service support by providing templates and development guidelines on how to develop APIs, service capabilities modules, and protocol plug-ins for installation in the WebLogic Network Gatekeeper SLEEs.

API/service capability extensibility

The APIs/service capabilities are implemented using a modular architecture. That is, when new service functionality is introduced in the network, corresponding service capability modules can be developed and installed in the WebLogic Network Gatekeeper. The service capability modules provides Web Services interfaces towards the applications.

The service capability modules use the same functions for installation, trace, management and version handling as all other software modules installed in the WebLogic Network Gatekeeper. Installation of service capability modules can be made in run-time.

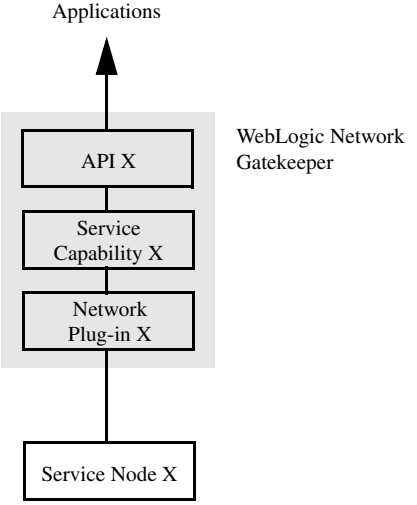


Figure 13-1 New modules related to the support of a new service capability

Custom APIs

The modular API structure makes it possible to create custom APIs or service capability modules combining functionality from lower level API. That is, a custom API can be created using functionality from different underlying service nodes and/or OSA/Parlay SCSes. Figure 13-2, “A custom service module combining service functionality from three different sources,” on page 13-3 shows a custom service module combining functionality from three different sources.

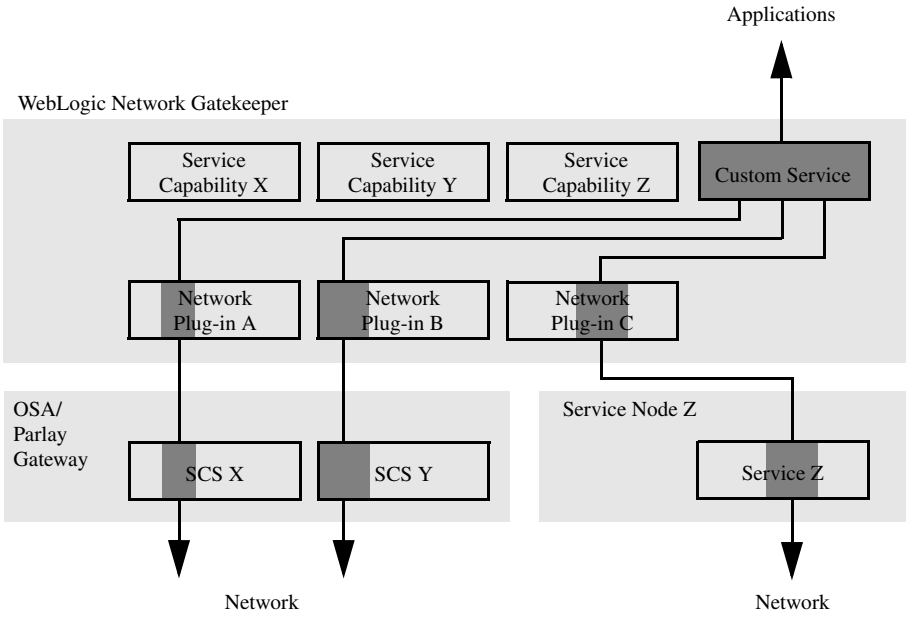


Figure 13-2 A custom service module combining service functionality from three different sources

Network protocol extensibility

When a new service node or system supporting an exisiting service capability module is introduced, that node or system can be available to all applications connected to the WebLogic Network Gatekeeper by introducing a new network plug-in. The only requirement is that the introduced node or system support the same set of service capabilities as the service capability modules’s plug-in interface (The plug-in interface is the API used by a network plug-in to

interface a service capability module) . As long as this requirement is fulfilled, service nodes and systems can be added and changed without affecting the applications. Minor interface deviations are handled by the WebLogic Network Gatekeeper through mapping or reporting errors back to the applications.

As the new plug-in is installed, the address formats (number plan and address range) it supports are registered through the WebLogic Network Gatekeeper's plug-in manager.

If an old service node shall be replaced or supplemented by a new service node supporting a newer protocol version, the nodes can work in parallel using two different network plug-ins. The service capability modules and the applications will not be affected.

Simple installation and OAM

Since the WebLogic Network Gatekeeper has a modular architecture where all functionality run as services in a Service Logic Execution Environment (SLEE), installation and QAM of new WebLogic Network Gatekeeper software modules (as service capability modules and network plug-ins) follow WebLogic Network Gatekeeper standard procedures.

Deployment service for installation, start and activation

The deployment service installs the new software modules's jar file in the WebLogic Network Gatekeeper. Software module specific databases and database tables are automatically created and the software module's characteristics and behaviour are set using an XML based deployment descriptor. The deployment service is also used for starting/stopping and activating/deactivating the software module. For more information, see User's Guide - WebLogic Network Gatekeeper.

Administration through the WebLogic Network Gatekeeper management tool

When a new service module is started through the deployment service, its management interface becomes available through the WebLogic Network Gatekeeper management tool. That is, all methods defined as management methods can be invoked through the management tools text and GUI based interfaces. For more information about the management tool's possibilities, see Chapter 9, "Operation, Administration and Maintenance".

Trace

If a software module is suspected to be faulty, the trace service can be used to locate the fault in the code. The amount of trace information provided by the trace service is dependant on the

amount of trace support implemented in the software module's code and the active trace level. The trace service can, on software module level, be activated/deactivated and the trace level can be changed in run time. All trace information is written to file.

Event log and alarm list

All internal software modules can write event and alarms to WebLogic Network Gatekeeper's event log and alarm list.

CHAPTER 14

Hardware Architecture

The following sections provide an overview of the system configurations on which WebLogic Network Gatekeeper runs:

Overview

An WebLogic Network Gatekeeper consists of two or more UNIX servers and is fully scalable to fit all capacity needs. The UNIX servers are connected through switches. High availability is achieved through duplication of all critical hardware.

Note: This chapter describes the hardware architecture on a general level. For detailed site specific more information, see the deplyment-specific documentation.

Configuration

A configuration consists of a number of UNIX servers connected through switches. The number of servers is determined by the WebLogic Network Gatekeeper capacity needs.

Applications and networks are connected to WebLogic Network Gatekeeper through the switches. A network time server can be used to synchronize the time between the UNIX servers. An example configuration with four servers is shown in Figure 14-1.

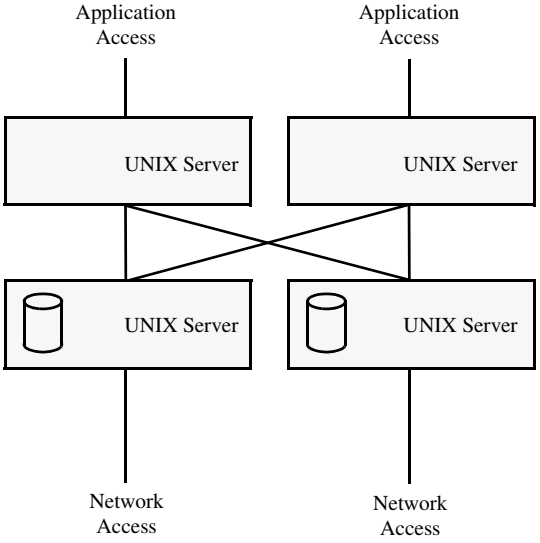


Figure 14-1 Example hardware configuration with four UNIX servers (Network equipment as routers and switches are not shown.)

For security reasons, the two servers used for the application access are separated from the network access servers. The application access servers reside in a De-militarized Zone (DMZ) while the network access servers are in the trusted environment.

High availability

All hardware is duplicated. This eliminates single point of failure on critical hardware components.

Scalability

The number of servers in an WebLogic Network Gatekeeper configuration is scalable from two up to 256 servers. Adding new servers is made in run-time without restarting the WebLogic Network Gatekeeper.

Individual servers can, depending on type, be upgraded with more CPUs, RAM, disk capacity and so on.

CHAPTER 15

Software Architecture Overview

The following sections provide an overview of the WebLogic Network Gatekeeper moduler software architecture:

Overview

WebLogic Network Gatekeeper has a modular software architecture where most functions run as services in a CORBA based Service Logic Execution Environment (SLEE). The main WebLogic Network Gatekeeper software modules are:

- SLEE
- SLEE services including:
  - Utility services (for a list of utility services, see "SLEE utility services" on page 16-3)
  - Service capability modules
  - SESPA (stateless adapters for web service access)
  - Service capability manager
  - Plug-in manager
  - Partner management tools
- SLEE database
- ORB
- WebLogic Network Gatekeeper management tool (not run within the SLEE)
- SLEE agent (not run within the SLEE)

The basic software architecture is shown in Figure 15-1, "WebLogic Network Gatekeeper software architecture overview (Partner management tools SLEE not shown)," on page 15-3. Note that the partner management tool service is run in a separate SLEE. For descriptions of the individual software modules, see Chapter 16, "Software Module Descriptions".

Processes

The following are the most important WebLogic Network Gatekeeper related processes:

- SLEE processes (one for each SLEE)
- SLEE agent processes (SLEE process supervision, one for each SLEE)
- Database processes
- Trace processes

When a SLEE process is started, the installed SLEE services are automatically started.

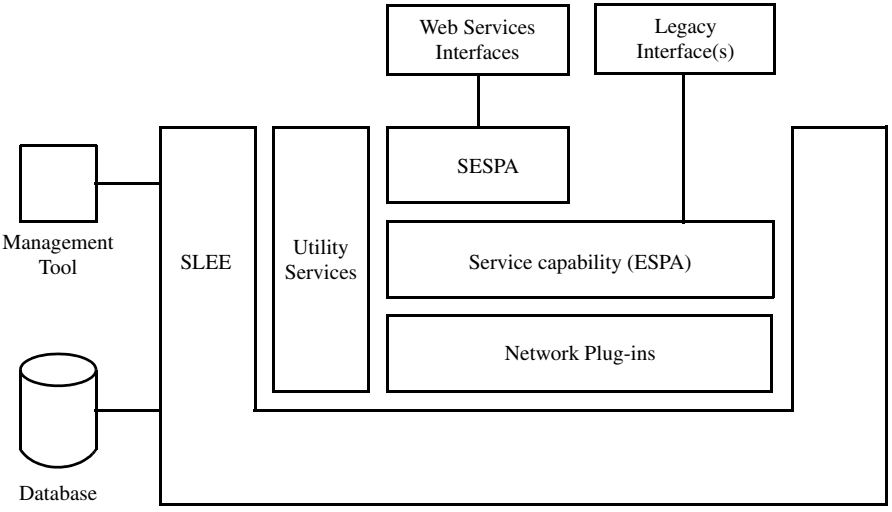


Figure 15-1 WebLogic Network Gatekeeper software architecture overview (Partner management tools SLEE not shown)

Service distribution

All SLEE service types can be distributed to execute in any number of SLEEs in the system. When installing a new SLEE service instance, it is automatically registered in all other SLEEs in the system. Service capability modules are registered in the service capability manager, and the network plug-ins are registered in the plug-in manager. All SLEEs have the utility services installed by default.

Capacity requirements decide the number of instances of each SLEE service to be installed in the system. For high availability reasons, at least two instances of each service shall be installed

Scalability

The capacity of the WebLogic Network Gatekeeper system is changed by changing the number of SLEEs running in the system. When adding a SLEE to an WebLogic Network Gatekeeper, a new server also has to be added.

Software configuration example

The following example describes an WebLogic Network Gatekeeper configuration with four SLEEs, one on each server. Two SLEEs have SESPA and Service capability modules (ESPA), and utility services installed. These SLEEs handle the communication with the applications. The two other SLEEs have network plug-ins and utility services installed. These SLEEs handle the network communication.

The example is shown in Figure 15-2.

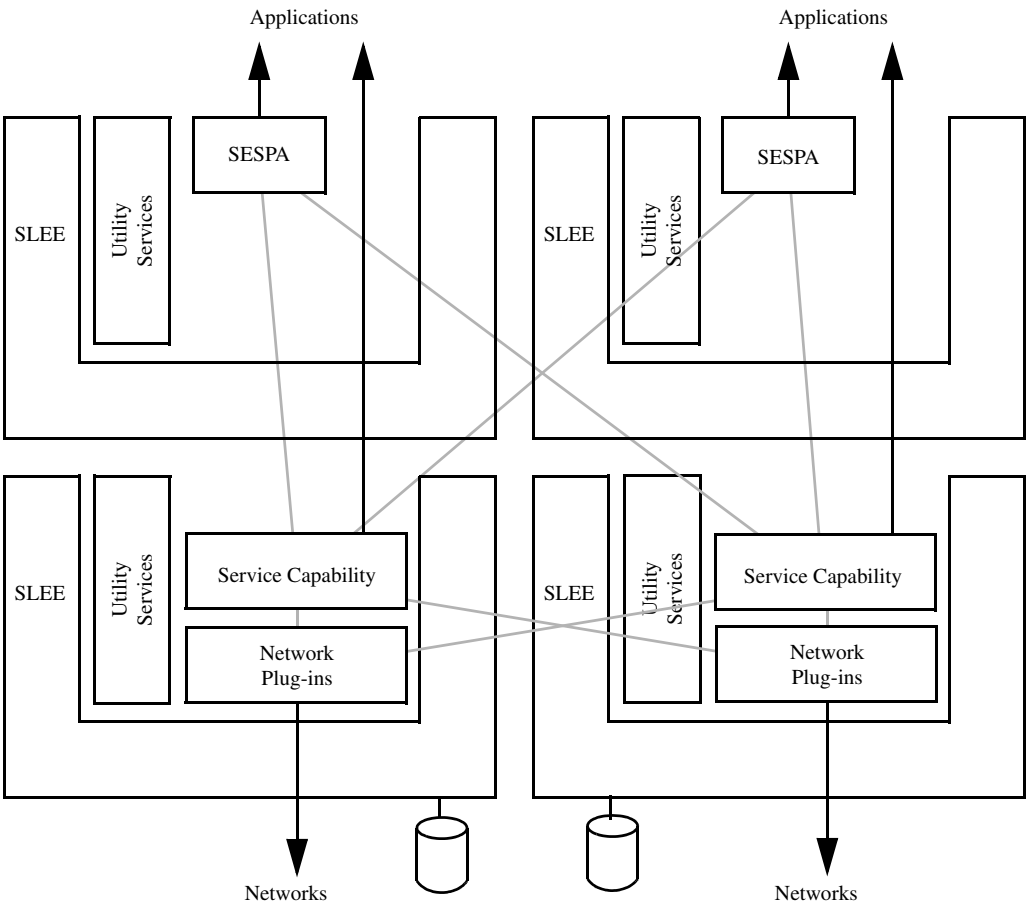


Figure 15-2 Example of an WebLogic Network Gatekeeper with four SLEEs (Partner management tool SLEEs not shown)

Scaling the system

If, for example a service capability module becomes overloaded, it is recommended to install a new server and SLEE with the same set of services as the other two SLEEs handling the application communication. This approach, with dedicated SLEEs, simplifies OAM of the system compared to having different sets of SLEE services installed on every server.

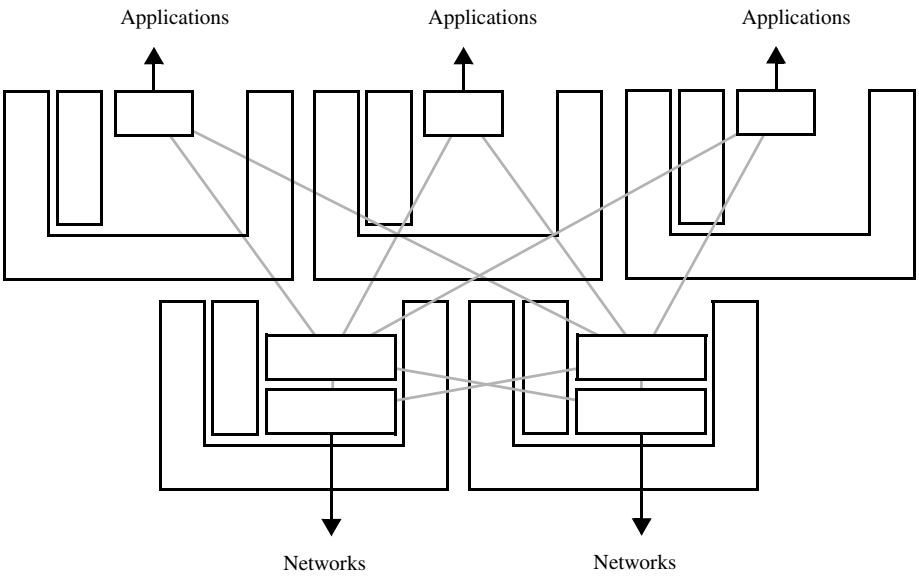


Figure 15-3 The example configuration with an extended application interface

Resource sharing contexts

The purpose of the concept with resource sharing contexts is to separate different types of traffic execution and to separate the traffic related execution from OAM related execution. In addition, resource sharing contexts can be used to separate the system's subnets from each other.

Each resource sharing context has its own ORB and name service. For the resource sharing contexts it is possible to specify:

- the CORBA thread pool size
- the SLEE thread pool size
- the SLEE task queue size

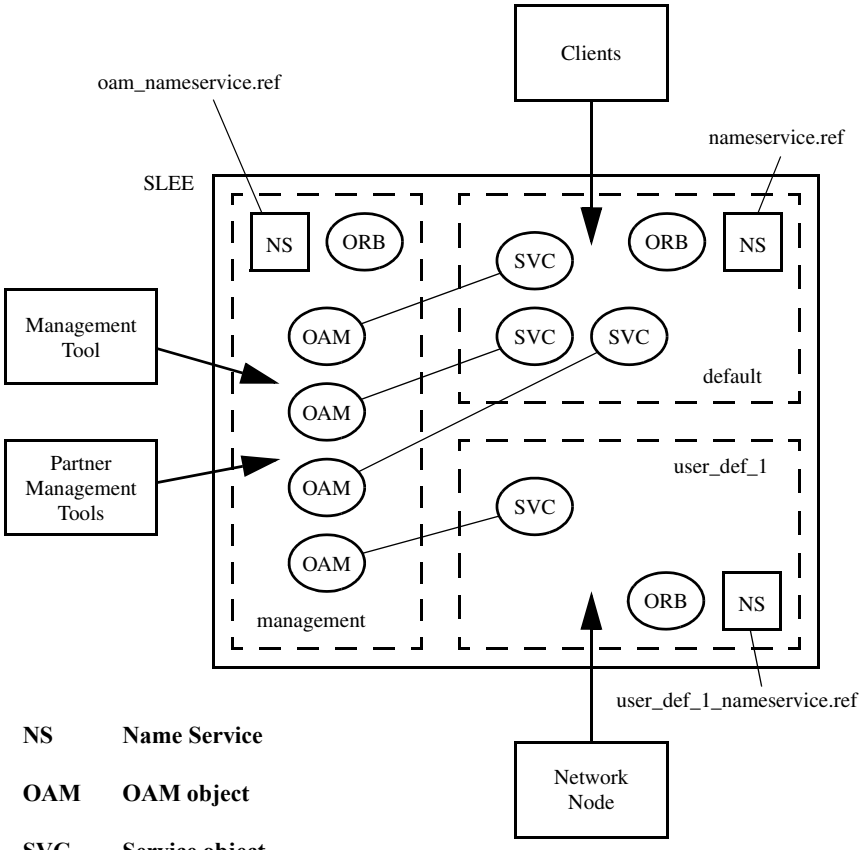


Figure 15-4 Resource sharing contexts (management, default and user def 1)

Figure 15-4, "Resource sharing contexts (management, default and user def 1)," on page 15-7 show a SLEE with three resource sharing context contexts. The *management* and *default* resource sharing contexts are default resource sharing context. Depending on the type of applications, system configuration and so on, more resource sharing contexts can be defined.

CHAPTER 16

Software Module Descriptions

The following sections describe WebLogic Network Gatekeeper software modules:

SLEE

The SLEE is a CORBA based service logic execution environment where all SLEE services are executed. By using CORBA also for management, the SLEE and its services can be managed remotely.

The SLEE supports version handling of services without restarting the SLEE process. When a new service version is installed, the new version takes care of all new traffic involving the service. The old version has to finish the traffic it is currently handling before it can be removed. When the service is removed, all files belonging to that service are automatically deleted from the system.

The SLEE process is supervised by the SLEE agent. The agent restarts the SLEE process if it terminates unexpectedly. When the SLEE is restarted (by the SLEE agent or manually), the services' restart order and previous operating states are retrieved from the database.

SLEE services

All software modules installed and run in the SLEE are regarded as SLEE services. SLEE services can have different behaviour in relation to the SLEE and the other SLEE services:

- A service that is possible to install and run in the SLEE is *deployable*. This is a mandatory requirement on a SLEE service.
- If it is possible to manage the service through the WebLogic Network Gatekeeper management tool it is also *manageable*.
- Finally, if it is possible to communicate with the service through CORBA, the service is *accessible*.

The SLEE gets information about a service's behaviour at service installation when it reads the service's deployment descriptor. The XML based deployment descriptor also describes other SLEE service characteristics such as service name and default settings.

An installed SLEE service can have one of the following states:

State	Description
Installed	The service software is installed in the SLEE.
Started	The service is started. If the service is manageable, it will also be available in the WebLogic Network Gatekeeper management tool.
Activated	The service is activated, that is, in its normal running state and accepts CORBA requests through its accessible interface.
Suspended	A sub-state between *Activated* and *Started* that is used for graceful shutdown of services. The service does not accept new requests, but it will finish all current assignments.
Error	The service has raised too many critical alarms and has been taken out of service by the SLEE. The allowed number of critical alarms is configured at service installation.

All service states are automatically restored in case of a SLEE restart.

SLEE service load balancing

There are two versions of load balancing. In the first case, the overloaded service instance passes on the request to another service instance that is not overloaded.

In the other case, the severely overloaded service informs the requesting service that it is overloaded and rejects the request. The requesting service takes care of finding another instance of the service. In the following sections, load balancing for each SLEE service type will be further explained.

SLEE utility services

Utility services are SLEE services that provide common support functions that can be used by all other SLEE services, see Table 16-1.

A utility service is called when another SLEE service wants to use the function the utility service provides.

Table 16-1 Utility Services

Utility Service	Provided Function
Alarm	Receives alarms and stores them in the database. Makes the alarms available to registered listeners through CORBA.
Charging	Receives CDRs and stores them in the database. Makes the CDRs available to registered listeners through CORBA.
Event log	Receives events and stores them in the database. Makes the events available to registered listeners through CORBA.
Global counters	Handles counters that are used by more than one SLEE instance.
Plug-in manager	Keeps track of the installed network plug-ins and related routes. Handles load balancing and high availability from the service capability modules towards the plug-ins.
Policy	Keeps track of all registered policies and makes policy decisions. For more information about the Policy service, see Chapter 7, "Policy".
Service capability manager	Keeps track of the installed service capability modules. Handles load balancing and high availability from the plug-ins towards the service capability modules.
Statistics	Recieves usage information and stores it in the database.
Task manager	Handles thread pools and task queues.
Time	Provides timers and timestamps when requested.
Trace	Makes it possible to retrieve trace information from a service that is suspected to be faulty. Saves the trace information to a file.

Service capability modules

Service capability modules provide CORBA based service capability interfaces. These interfaces are used by SESPA (see "SESPA modules" on page 16-7).

This section describes the functionality provided by a service capability module independently of the service capability it implements. For information about the service capabilities supported by WebLogic Network Gatekeeper, see Chapter 3, "Service Descriptions".

The main tasks performed by the service capability modules are:

- Service provisioning
- Charging (or service usage) data generation
- Policy enforcement
- Plug-in selection
- Load balancing towards the plug-ins

Figure 16-1 shows a service capability module connecting an application to two different networks through two network plug-ins. Also, the service capability module's interfaces to the charging, policy, plug-in manager and service capability manager utility services are shown.

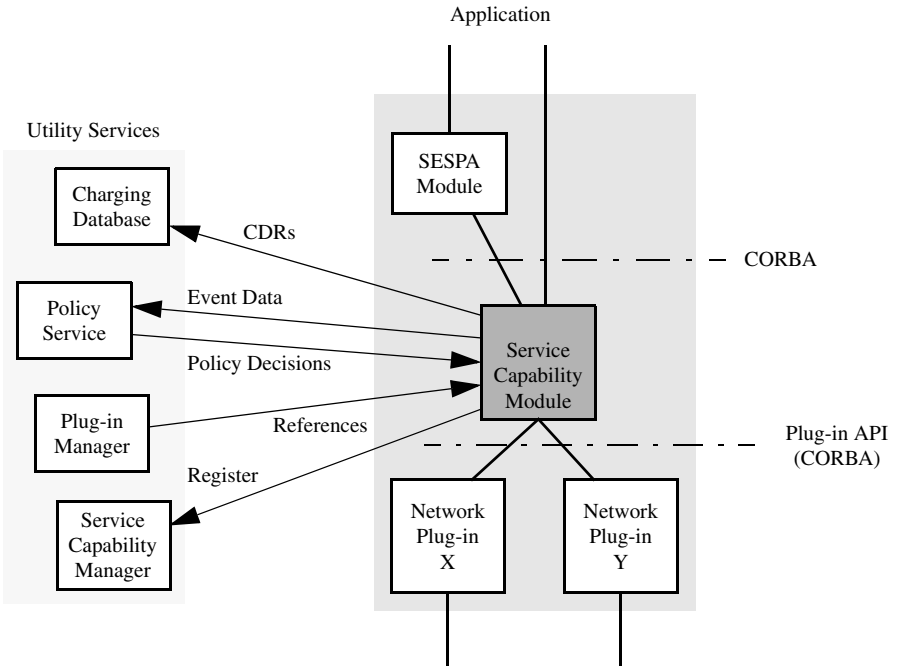


Figure 16-1 An service capability module connecting an application to two different networks through two network plug-ins

Charging data generation

The service capability modules are responsible for collection and storage of charging data, see "CDR based charging" on page 8-2.

Plug-in selection

The applications' requests are routed to different network plug-ins depending on the destination address plan and address range. A plug-in manager included in each SLEE provides the routing functionality. All routing data is available in the database Routes to individual plug-ins are specified using regular expressions that match the addresses to be routed to the connected networks.

Policy enforcement

The service capability modules enforce the policies defined in the service provider,application, and node SLAs. The policies are enforced in the Policy Enforcement Points (PEPs). For more information on policy, see "Policy" on page 7-1.

Load balancing

An overloaded service capability module can ask the SLEE if there are other instances of the service capability module having normal load levels. If there are, the request to use the service capability module is forwarded to one of the other service capability module instances.

For network initiated events, the service capability module manager handles load balancing towards service capability module instances according to a round robin schema.

High availability

From the plug-in perspective, the service capability module manager has a list of references to all service capability module instances of each type. When an application registers a listener for a network triggered event, a reference to that application is stored in the internal database. If a service capability module instance becomes unavailable, the plug-in gets a new service capability module instance reference from the service capability module manager and forwards the network triggered event to the new service capability module instance. The new service capability module instance uses the reference in the database to notify the application.

SESPA modules

SESPA modules provide stateless interfaces for the service capabilities. The states are stored persistently in the database and requests within one service session can be distributed across multiple SLEE instances for load balancing and high availability.

The SESPA modules are used for the Web Service access. The switches handle the initial HTTP access high availability and load distribution among individual SESPA module instances.

Load balancing

The SESPA modules balance the load between available service capability modules of the same type.

High availability

The SESPA modules provides high availability between available service capability modules of the same type.

Network plug-ins

A network plug-in is a SLEE service connecting a service capability module to a service node, a third party OSA/Parlay SCS, or other system in the underlying networks. For a list of available core network plug-ins, see "Supported network protocols" on page C-5.

A plug-in consists of two parts, a service capability module specific part implementing the service capability module's plug-in interface and a network specific part interfacing the network. The plug-in converts the service capability module's CORBA based plug-in interface to fit the interface exposed by the network. New protocol/network support can be added without affecting the service capability module. New plug-ins are added in run-time. Even multiple versions of a network protocol can be handled using multiple plug-ins.

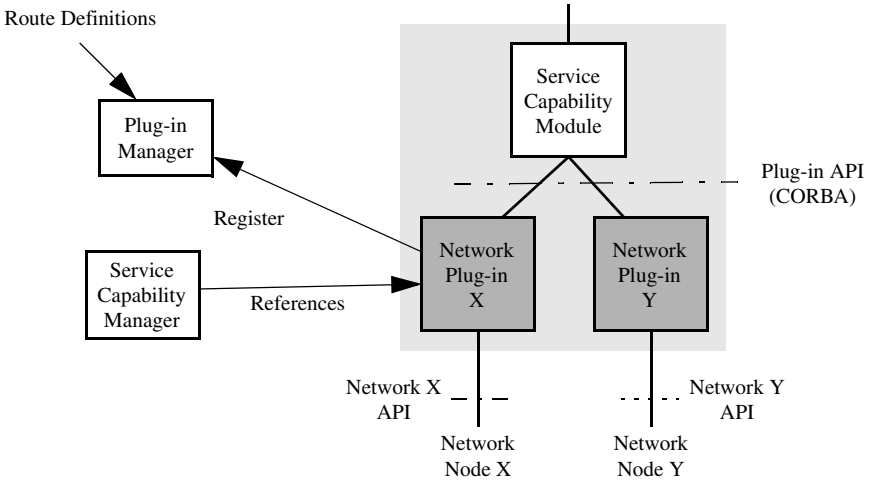


Figure 16-2 Two plug-ins connecting to two different networks

When a network plug-in is installed in the WebLogic Network Gatekeeper, it registers itself in the plug-in manager instance of the SLEE it is being installed in. The SLEE will then notify the other SLEEs in the system that the new plug-in is available for use. The plug-in manager's routing function routes service requests to different plug-ins depending on the destination party's address format (supported address plan and address range). The routes are specified through the plug-in manager.

For a network triggered event, the service capability module manager provides the plug-in with a service capability module instances reference according to a round robin schema.

Load balancing

The plug-in manager divides the requests from the service capability module according to a round robin schema. The more installed plug-ins of the same type supporting the same address type and routing, the lower the load on the individual instances.

High availability

If a plug-in is unavailable, the service capability module gets a new plug-in instance reference from the plug-in manager according to a round robin schema.

Database

Storage and replication

The WebLogic Network Gatekeeper database is a replicated SQL database accessed through JDBC. The two sides of the database run on different servers. If a SLEE service wants to use an external database, this is handled through JDBC.

A service that wants to access the database requests a database connection from the SLEE. The SLEE then tries to get a connection to the master database. In case of failure, the SLEE tries to get a connection to the slave. If the SLEE gets a connection to the slave but not to the master, the SLEE changes the slave to master and raises an alarm. The requesting service will not notice any of this as long as it gets its database connection. Services will only be informed if the SLEE fails to provide a database connection.

For information about database security, see "Database security" on page 12-4.

CHAPTER 17

Redundancy and Load Balancing

The following sections describe WebLogic Network Gatekeeper high availability and load balancing features:

Applications to network nodes fault tolerance

Fault tolerance downstream is defined as the ability of the WebLogic Network Gatekeeper to hide that an WebLogic Network Gatekeeper software module or server has become unavailable, and seamlessly switch over the service execution to other software module instances.

Fault tolerance upstream is defined as the ability for the WebLogic Network Gatekeeper to provide network triggered events from the a service capability in the network to a subscribing application even if an WebLogic Network Gatekeeper software module or server becomes unavailable.

Three areas are identified as key areas when it comes to provide fault tolerance:

- Preservation of listeners, states and sessions.
- High availability and load balancing between software modules.
- Overall platform robustness

Preservation of listeners, states and sessions

The WebLogic Network Gatekeeper implements different levels of fault tolerance for different service capabilities. Below follows a description of these mechanisms for the affected service capabilities.

• Call	All notification listeners are preserved.
• Charging	A charging session that has been lost can be can be retrieved as long as the ID, given when the session was created, is known.
• Messaging	All open mailboxes are preserved. Likewise are the states of all folders preserved. All operations defined for the manager, the mailboxes and the folders are available and the applications can use these seamlessly. All notification listeners are preserved.

High availability and load balancing between software modules

The WebLogic Network Gatekeeper provides high availability and load balancing between software modules in all layers, see Figure 17-1.

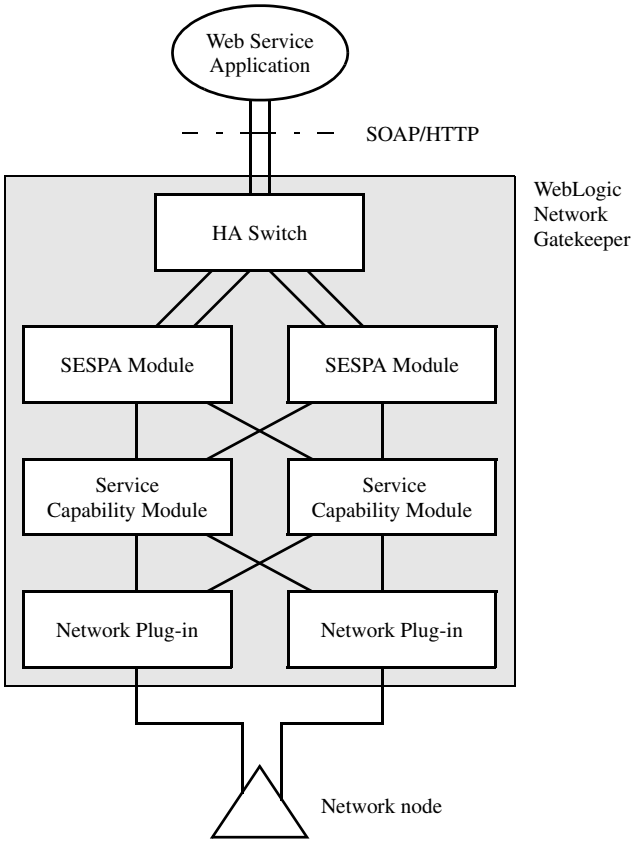


Figure 17-1 End-to-end high availability and load balancing through WebLogic Network Gatekeeper

The load balancing and high availability mechanisms implemented in each software module are described in Chapter 16, "Software Module Descriptions".

Platform robustness

The WebLogic Network Gatekeeper itself is built on a carrier grade platform in order to allow high availability for the executing applications. For more information, see Chapter 14, "Hardware Architecture".

APPENDIX A

Charging Data

The following sections provide a reference to WebLogic Network Gatekeeper charging data:

Charging data format

The CDR data is stored in the MySQL database in the following format:

Field	Format
transaction_id	bigint
session_id	bigint
service_name	varchar
user_id	varchar
start_of_usage	bigint
connect_time	bigint
end_of_usage	bigint
duration_of_usage	bigint
amount_of_usage	bigint
originating_party	varchar
destination_party	varchar
transaction_part_number	int
completion_status	int
additional_info	blob
revenue_share_percentage	float
charging_info	varchar
party_to_charge	varchar
service_provider	varchar
slee_instance	varchar

All services that produce charging data do not use all fields. See the section below on details of which fields that are relevant for the different services.

All times given, except duration of usage, are relative to the UNIX era, that is number of milliseconds since midnight, January 1, 1970, Greenwich Mean Time, GMT.

Duration of usage is given in seconds.

Charging

Field Name	Description
transaction_id	A unique ID for the transaction.
session_id	An ID used to connect related transactions.
service_name	The service capability module (Charging) that has created the CDR.
user_id	The application that used the service capability module.
completion_status	Indicates the completion status of the transaction. 0 - failed 1 - completed
additional_info (in XML format)	Possible data: • <merchant_id> </merchant_id> The merchant ID, for instance the ID of the organisation or company that provided the content. • <account_id> </account_id> The account the charging operation is performed on. • <user_id> </user_id> The ID of the user that used the service. • <method_name> </method_name> Provides the type of charging performed.

Field Name	Description
charging_info	Service code added by the application or by the policy service. Can also be used to provide the actual price.
party_to_charge	The ID of the party that initiated the request that resulted in the transaction.
service_provider	The ID of the service provider that hosts the application.
slee_instance	The SLEE instance hosting the service capability module that generated the CDR.

Call

Field Name	Description
transaction_id	A unique ID for the transaction.
session_id	An ID used to connect related transactions.
service_name	The service capability module (Call control) that has created the CDR.
user_id	The application that used the service capability module.
start_of_usage	The date and time the service capability module started to use services in the telecom network.
connect_time	The date and time the destination party responded.
end_of_usage	The date and time the service capability module stopped to use services in the telecom network.
duration_of_usage	The total time the service capability module used the network services. Value is 0 (zero) if no answer or route fail.
originating_party	The address of the originating party.
destination_party	The address of the destination party.

Field Name	Description
transaction_part_ number	Always equal to 1.
completion_status	Indicates the completion status of the transaction. 0 - failed 1 - completed
charging_info	Service code added by the application or by the policy service. Can also be used to provide the actual price.
party_to_charge	The ID of the party that initiated the request that resulted in the transaction.
service_provider	The ID of the service provider that hosts the application.
slee_instance	The SLEE instance hosting the service capability module that generated the CDR.

Messaging

Field Name	Description
transaction_id	A unique ID for the transaction.
session_id	An ID used to connect related transactions.
service_name	The service capability module (Messaging) that has created the CDR.
user_id	The application that used the service capability module.
start_of_usage	The date and time the service capability module started to use services in the telecom network.
amount_of_usage	Always equal to 1.
originating_party	The address of the originating party.
destination_party	The address of the destination party.

Field Name	Description
transaction_part_ number	Always equal to 1.
completion_status	Indicates the completion status of the transaction. Outgoing message: 0 - failed 1 - completed 2 - partial (indicates that SEND_RESULT=OK, another CDR with status completed or failed will be created after DEVLIVERY_ACK has been recieved) Incomig message: 0 - failed 1 - completed 2 - partial (indicates that the message has been successfully stored in the database, but there is no listener registered) 3 - completed but notfication failed (indicates that the message has been successfully stored in the database, but the notification towards the application failed)
additional_info (in XML format)	Possible data: • <method> </method> The type of operation performed [DELIVERY_ACK | SEND_RESULT | MESS_ARRIVED] Possible data if method=DELIVERY_ACK: • <sendListSize> </sendListSize> - no. of addresses in send list. • <byteSize> </byteSize> - size of message in bytes. • <mbox></mbox> - mailbox address • <msg_type></msg_type> SMS or MMS Possible data if method=SEND_RESULT: • <sendListSize> </sendListSize> - no. of addresses in send list. Possible data if method=MESS_ARRIVED: • <byteSize> </byteSize> - size of message in bytes. • <msg_type></msg_type> SMS or MMS

Field Name	Description
charging_info	Service code added by the application or by the policy service. Can also be used to provide the actual price.
party_to_charge	The ID of the party that initiated the request that resulted in the transaction.
service_provider	The ID of the service provider that hosts the application.
slee_instance	The SLEE instance hosting the service capability module that generated the CDR.

Subscriber profile

Field Name	Description
transaction_id	A unique ID for the transaction.
service_name	The service capability module (Subscriber profile) that has created the CDR.
user_id	The application that used the service capability module.
start_of_usage	The date and time the service capability module started to use services in the telecom network.
amount_of_usage	The used amount. Used when charging is not time dependent. For example, flat rate services.
destination_party	The address of the destination party. That is, the subscriber the profile belongs to.
transaction_part_number	Always equal to 1.
completion_status	Indicates the completion status of the transaction. 0 - failed 1 - completed 3 - completed but callback to application failed

Field Name	Description
additional_info (in XML format)	Possible data: • <op> </op> Provides the type of subscriber profile operation performed [get | set]
charging_info	Service code added by the application or by the policy service. Can also be used to provide the actual price.
party_to_charge	The ID of the party that initiated the request that resulted in the transaction.
service_provider	The ID of the service provider that hosts the application.
slee_instance	The SLEE instance hosting the service capability module that generated the CDR.

User interaction

Field Name	Description
transaction_id	A unique ID for the transaction.
session_id	An ID used to connect related transactions.
service_name	The service capability module (User interaction) that has created the CDR.
user_id	The application that used the service capability module.
completion_status	Indicates the completion status of the transaction. 0 - failed 1 - completed 3 - completed but callback to application failed

Field Name	Description
additional_info (in XML format)	Possible data: • <method name> </method name> The type of operation performed [abortActionErr | abortActionRes | deleteMessageErr | deleteMessageRes | recordMessageErr | recordMessageRes | sendInfoAndCollectErr | sendInfoAndCollectRes | sendInfoErr | sendInfoRes | userInteractionFaultDetected] • <assignmentID> </assignmentID> Provides the assignment ID. • <infoID> </infoID> Provides the announcement's infoID. Used for call based UI only. • <serviceCode> </serviceCode> Provides the service code used to identify the application. Used for network initiated UI requests (UI notification) or sendInfoAndCollect responses. • <resourceType> </resourceType> Provides the plug-in type [SMS | USSD | GUI].
charging_info	Service code added by the application or by the policy service. Can also be used to provide the actual price.
party_to_charge	The ID of the party that initiated the request that resulted in the transaction.
service_provider	The ID of the service provider that hosts the application.
slee_instance	The SLEE instance hosting the service capability module that generated the CDR.

User location

Field Name	Description
transaction_id	A unique ID for the transaction.
session_id	An ID used to connect related transactions.
service_name	The service capability module (User location) that has created the CDR.
user_id	The application that used the service capability module.
start_of_usage	The date and time the service capability module started to use services in the telecom network.
amount_of_usage	Always equal to 1.
destination_party	The address of the destination party.
transaction_part_ number	Always equal to 1.
completion_status	Indicates the completion status of the transaction. 0 - failed 1 - completed 3 - completed but callback to application failed
additional_info (in XML format)	Possible data: • <location_type> </location_type> The type of operation performed [standard | extended | periodic] • <cause> </cause> Provides an indication on the possible cause in case of an unsuccesful status request.

Field Name	Description
charging_info	Service code added by the application or by the policy service. Can also be used to provide the actual price.
party_to_charge	The ID of the party that initiated the request that resulted in the transaction.
service_provider	The ID of the service provider that hosts the application.
slee_instance	The SLEE instance hosting the service capability module that generated the CDR.

User status

Field Name	Description
transaction_id	A unique ID for the transaction.
session_id	An ID used to connect related transactions.
service_name	The service capability module (User status) that has created the CDR.
user_id	The application that used the service capability module.
start_of_usage	The date and time the service capability module started to use resources in the telecom network.
amount_of_usage	The used amount. Used when charging is not time dependent. For example, flat rate services.
destination_party	The address of the destination party.
transaction_part_ number	Always equal to 1.
completion_status	Indicates the completion status of the transaction. 0 - failed 1 - completed 3 - completed but callback to application failed

Field Name	Description
additional_info (in XML format)	Possible data: • <cause> </cause> Provides an indication on the possible cause in case of an unsuccesful status request.
charging_info	Service code added by the application or by the policy service. Can also be used to provide the actual price.
party_to_charge	The ID of the party that initiated the request that resulted in the transaction.
service_provider	The ID of the service provider that hosts the application.
slee_instance	The SLEE instance hosting the service capability module that generated the CDR.

APPENDIX B

Policy Data

The following sections provide a reference to WebLogic Network Gatekeeper policy data:

General policy data

All policy rules can use the following general policy data:

- Application ID
- Service capability module name
- Service Level Agreement (SLA) data - For more information about the SLA data, see User's Guide - WebLogic Network Gatekeeper.
- Date
- Time
- Service code
- Service provider ID

Request specific policy data

Request specific data is provided with the service requests. This data can be combined with the general policy data when creating custom policies.

Charging

Request	Available Data
Create charging session	• Merchant ID • Account ID • User address
Credit amount	• Merchant ID • Account ID • User address • Number • Exponent Number and Exponent are used to define the credited amount as: Amount = Number x $10^{Exponent}$

Request	Available Data
Credit unit	See request: Credit amount.
Debit amount	See request: Credit amount.
Debit unit	See request: Credit amount.
Direct credit amount	See request: Credit amount.
Direct credit unit	See request: Credit amount.
Direct debit amount	See request: Credit amount.
Direct debit unit	See request: Credit amount.
Get charging session	-
Reserve amount	• Merchant ID • Account ID • User address • Preferred number • Preferred exponent • Minimum number • Minimum exponent Number and Exponent are used to define the preferred and minimum reserved amounts as: Amount = Number x $10^{Exponent}$
Reserve unit	See request: Credit amount.

Messaging

Request	Available Data
Enable messaging notification	• Mailbox address • Event name
Open mailbox	• Mailbox address

Request	Available Data
Put message	• Sent to address(es) • Sent from address • Priority
Put multimedia message	See request: Put message
Get message	• Message ID
Messaging event notify	• Mailbox address • Message ID • Sent to address • Sent from address
Message result	• Message ID • Sent to address • Sent from address • Delivery status

Call

Request	Available Data
Create call	• Number of active calls
Create call leg	• Number of current call legs in the call
Enable notification	• Destination address range • Originating address range
Route	• Destination address • Originating address

Request	Available Data
Report notification	• Destination address • Originating address • Call event type • Monitor mode
Network initiated call	• Destination address

Subscriber profile

Request	Available Data
Get information	• Destination address
Set information	• Destination address

User interaction

Request	Available Data
Create user interaction	-
Send information and collect user input	• Specification on how the information is sent to the user. This information can be one of the following: - an infoID, identifying pre-defined information to be send (announcement and/or text); - a string, defining the text to be sent; - a URL, identifying pre-defined information or data to be sent to or downloaded into the terminal. • Specification of which language the information is given in. • Specification if a response is required from the call user interaction service, and any action the service should take. • Destination address.

Request	Available Data
Send information	• Specification on how the information is sent to the user. This information can be one of the following: - an infoID, identifying pre-defined information to be send (announcement and/or text); - a string, defining the text to be sent; - a URL, identifying pre-defined information or data to be sent to or downloaded into the terminal. • Specification of which language the information is given in. • Specification if a response is required from the call user interaction service, and any action the service should take. • Definition of how many times the information shall be sent to the end-user. • Destination address.
User interaction event notify	• Sent to address • Sent from address • Service code Defines a 2-digit code which indicates the user interaction to be triggered. The value is operator specific. • Info data type This information specifies which of the following data types the user interaction uses: - binary data; - text; - unspecified
Send Info result	• Message ID • Sent to address • Sent from address • Delivery status

User location

Request	Available Data
User location report	• Destination address(es) • Allowed destination address(es)
Extended user location report	• Destination address(es) • Priority (high, low) • Accuracy (in meters) • Allowed destination address(es)
Periodic user location report	• Destination address(es) • Priority (high, low) • Accuracy (in meters) • Reporting interval (in seconds) • Allowed destination address(es)

User status

Request	Available Data
User status report	• Destination address(es) • Allowed destination address(es)

APPENDIX C

Technical Specification

The following sections summarize the technical specifications of WebLogic Network Gatekeeper:

Supported Configurations

Network Gatekeeper Base platform

Intel Itanium2

Operating system: HP-UX 11.23

Java: HP-UX 1.4.2 SDK

Intel Xeon

Operating system: Linux Redhat AS3

Java: JRockit 1.4.2 SDK

Network Gatekeeper Application Test Environment

Intel Pentium

Operating system: Windows XP

Java: Sun 1.4.2 SDK

Database included

MySQL 4.0

General characteristics

CORBA version	CORBA 2.5
Database	MySQL 4.0
Java version	JRE 1.4, JDBC 3.0
ORB	Orbacus 4.1.2
Parlay X	1.0
Rule engine	JRules 4.5
SNMP version	v1, v2
SOAP version	1.1, 1.2
SOAP engine	Axis 1.1

Programmable interfaces

Interface	Description
Plug-in interfaces for: • Charging • Call control • Messaging SMS • Messaging MMS • Subscriber profile • User interaction call based • User interaction message based • User location • User status	Makes it possible to add new network plug-ins for extended network/protocol support.
Extended API based interfaces for: • Access • Call • Charging • Messaging • Subscriber profile • User interaction • User location • User status	Provides high level telecom Web Services APIs.

Interface	Description
Parlay X based interfaces for: • Multimedia message • Network initiated third party call • Payment • SMS • Terminal location • Third party call • User status	Provides high level telecom Web Services APIs.
Utility service interfaces for: • Alarm handling • Charging • Event handling • High availability • Time • Trace	Facilitates development by providing support functions.

Supported network protocols

Service Capability	Supported Protocols
Call	Call control: • Parlay 3.3 (OSA rel 4) Multiparty call control: • Parlay 3.3 (OSA rel 4)
Charging	Parlay 3.3 (OSA rel 4)
Messaging	• CIMD2 • Ericsson MM7 R2.0 • MM7 rel 5 • Nokia EAIF • SMPP version 3.4 • Parlay 3.3 (OSA rel 4) SMS and MMS

Service Capability	Supported Protocols
Subscriber Profile	A network plug-in for connection with a database in the WebLogic Network Gatekeeper is available. Customized plug-ins needed for integration with external databases.
User Interaction	Call user interaction: • Parlay 3.3 (OSA rel 4) Generic user interaction: • CIMD2 • SMPP version 3.4 • Parlay 3.3 (OSA rel 4)
User Location	• MLP (LIF) 3.0.0 • MPP version 3.0 • Parlay 3.3 (OSA rel 4)
User Status	• MPP version 3.0 • Parlay 3.3 (OSA rel 4)

APPENDIX D

References

User's Guide - WebLogic Network Gatekeeper

Application Developer's Guide - Parlay X for WebLogic Network Gatekeeper

User's Guide - WebLogic Network Gatekeeper Application Test environment

API Descriptions - Extended Web Services for WebLogic Network Gatekeeper

API Descriptions - Parlay X for WebLogic Network Gatekeeper