BEA WebLogic Network Gatekeeper™

Architectural Overview

Version 3.0™
Document Revised: 14 September 2007

Copyright

Restricted Rights Legend

Trademarks and Service Marks

Contents

3. Software Architecture Overview

4. Introducing Traffic Paths

9. Redundancy, Load Balancing, and High Availability

10. Service Extensibility

11. Backwards Compatibility 2.2 to 3.0

A. Standards and Specifications

B. Connecting to OSA/Parlay Gateways

C. Technical Specifications

CHAPTER 1

Document Roadmap

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

Document Scope and Audience

This document provides a high-level account of WebLogic Network Gatekeeper, its structure and capabilities, consisting of:

- an overview of how it works and what benefits it provides
- an explanation of the interfaces it offers third-party application developers, including a description of the tools it furnishes for development and testing
- a summary view of its internals, including:
  - service capabilities
  - OAM mechanisms
  - policy enforcement
  - security

  - billing capabilities
  - integration with PRM/CRMs and OSS tools
  - product extensibility
- a description of supported hardware architecture and components
- a description of software architecture

The document will be of use to third-party application developers who wish to integrate telephony-based functionality into their products and operator-based system developers who wish to extend the functionality of the WebLogic Network Gatekeeper or to integrate it with PRM and/or OSS tools. It will also be of use to system administrators charged with installing and maintaining WebLogic Network Gatekeeper. Managers, support engineers, and sales and marketing people will also find information of value here.

Guide to This Document

The document contains the following chapters:

- Chapter 1, "Document Roadmap." This chapter
- Chapter 2, "Introducing WebLogic Network Gatekeeper." An overview of the benefits Network Gatekeeper provides both application developers and network operators
- Chapter 3, "Software Architecture Overview." A high level look at Network Gatekeeper's internal architecture
- Chapter 4, "Introducing Traffic Paths." An overview of the traffic path functionality
- Chapter 5, "Developing and Testing Applications." The interfaces offered to third-party developers and the tools available to aid in testing and development
- Chapter 6, "Managing Application Service Providers." An overview of the administration model for third-party application service providers
- Chapter 7, "Managing Network Gatekeeper: OAM."The Network Gatekeeper's application management tool and the main Operation, Administration and Maintenance (OAM) tasks. Integrating with OSS
- Chapter 8, "Charging." Supported charging types. Integrating WebLogic Network Gatekeeper's internal charging mechanism with external billing and settlement systems

- Chapter 9, “Redundancy, Load Balancing, and High Availability.” Fault tolerance, high availability, and load balancing mechanisms from an application and network perspective. Geo-redundancy.
- Chapter 10, “Service Extensibility.” Extending WebLogic Network Gatekeeper by creating modules to support additional application service providers and/or network connectivity interfaces
- Chapter 11, “Backwards Compatibility 2.2 to 3.0.”Overview of the relationship between Network Gatekeeper 2.2 and 3.0
- Appendix A, “Standards and Specifications.” Detailed description of the standards and specifications supported by Network Gatekeeper’s application-facing interfaces, network facing protocols, and security mechanisms
- Appendix B, “Connecting to OSA/Parlay Gateways.” Overview of how WebLogic Network Gatekeeper connects to OSA/Parlay Gateways
- Appendix C, “Technical Specifications.” Detailed description of supported configurations

Terminology

The following terms and acronyms are used in this document:

- Account—A registered application or service provider, associated with an SLA
- Account group—Multiple registered service providers or services which share a common SLA
- Administrative User—Someone who has privileges on the Network Gatekeeper management tool. This person has an administrative user name and password
- Alarm—The result of an unexpected event in the system, often requiring corrective action
- API—Application Programming Interface
- Application—A TCP/IP based, telecom-enabled program accessed from either a telephony terminal or a computer
- Application-facing Interface—The Application Services Provider facing interface
- Application Service Provider—An organization offering application services to end users through a telephony network
- AS—Application Server

- Application User—An Application Service Provider from the perspective of internal Network Gatekeeper administration. An Application User has a user name and password
- CBC—Content Based Charging
- End User—The ultimate consumer of the services that an application provides. An end user can be the same as the network subscriber, as in the case of a prepaid service or they can be a non-subscriber, as in the case of 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
- Event—A trackable, expected occurrence in the system, of interest to the operator
- HA —High Availability
- HTML—Hypertext Markup Language
- IP—Internet Protocol
- JDBC—Java Database Connectivity, the Java API for database access
- Location Uncertainty Shape—A geometric shape surrounding a base point specified in terms of latitude and longitude. It is used in terminal location
- 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
- MM7—A multimedia messaging protocol specified by 3GPP
- MPP—Mobile Positioning Protocol
- Network Plug-in—The WebLogic Network Gatekeeper module that implements the interface to a network node or OSA/Parlay SCS through a specific protocol
- NS—Network Simulator
- OAM —Operation, Administration, and Maintenance
- Operator—The party that manages the Network Gatekeeper. Usually the network operator
- OSA—Open Service Access
- PAP—Push Access Protocol

- Plug-in—See Network Plug-in
- Plug-in Manager—The Network Gatekeeper module charged with routing an application-initiated request to the appropriate network plug-in
- Policy Engine—The Network Gatekeeper module charged with evaluating whether a particular request is acceptable under the rules
- Presence Information—A status indicator that conveys the accessibility and the willingness of a potential communication partner.
- Presentity—A supplier of presence information.
- Quotas—Access rule based on an aggregated number of invocations. See also Rates
- Rates—Access rule based on allowable invocations per time period. See also Quotas
- Rules—The customizable set of criteria - based on SLAs and operator-desired additions - according to which requests are evaluated
- SCF—Service Capability Function or Service Control Function, in the OSA/Parlay sense.
- SCS—Service Capability Server, in the OSA/Parlay sense. WebLogic Network Gatekeeper can interact with these on its network-facing interface
- Service Capability—Support for a specific kind of traffic within WebLogic Network Gatekeeper. Defined in terms of traffic paths
- Service Provider—See Application Service Provider
- SIP—Session Initiation Protocol
- SLA—Service Level Agreement
- 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 signs up for access to an application. The subscriber is charged for the application service usage. See End User
- SQL—Structured Query Language
- TCP—Transmission Control Protocol
- Traffic Path—The data flow of a particular request through WebLogic Network Gatekeeper. Different Service Capabilities use different traffic paths
- USSD—Unstructured Supplementary Service Data
- VAS—Value Added Service
- VLAN—Virtual Local Area Network
- VPN—Virtual Private Network
- Watcher—A consumer of presence information
- WebLogic Network Gatekeeper Core—The container that holds the Core Utilities
- WebLogic Network Gatekeeper Core Utilities—A set of utilities common to all traffic paths
- WSDL —Web Services Definition Language
- XML—Extended Markup Language

Related Documentation

This architectural overview is a part of the WebLogic Network Gatekeeper documentation set. The other documents include:

- *System Administrator's Guide*
- *Handling Alarms*
- *Installation Guide*
- *Integration Guidelines for Partner Relationship Management*
- *Managing Service Providers and Applications*
- *Statement of Compliance*
- *Application Development Guide*

- *SDK User Guide*
- *Extension Toolkit - Developer's Guide*
- *System Backup and Restoration Guide*
- *Licensing*
- *Traffic Path Reference*

Additionally, many documents in the WebLogic Server 9.2 documentation set are of interest to users of WebLogic Network Gatekeeper, including:

- *Introduction to BEA WebLogic Service and BEA WebLogic Express™*
- *WebLogic Server - Installation Guide*
- *Managing Server Startup and Shutdown*
- *Programming Web Services for WebLogic Server*
- *Developing Manageable Application with JMX*
- *Configuring and Using the WebLogic Diagnostics Framework*
- *Using WebLogic Clusters*
- *Securing WebLogic Server*

CHAPTER 2

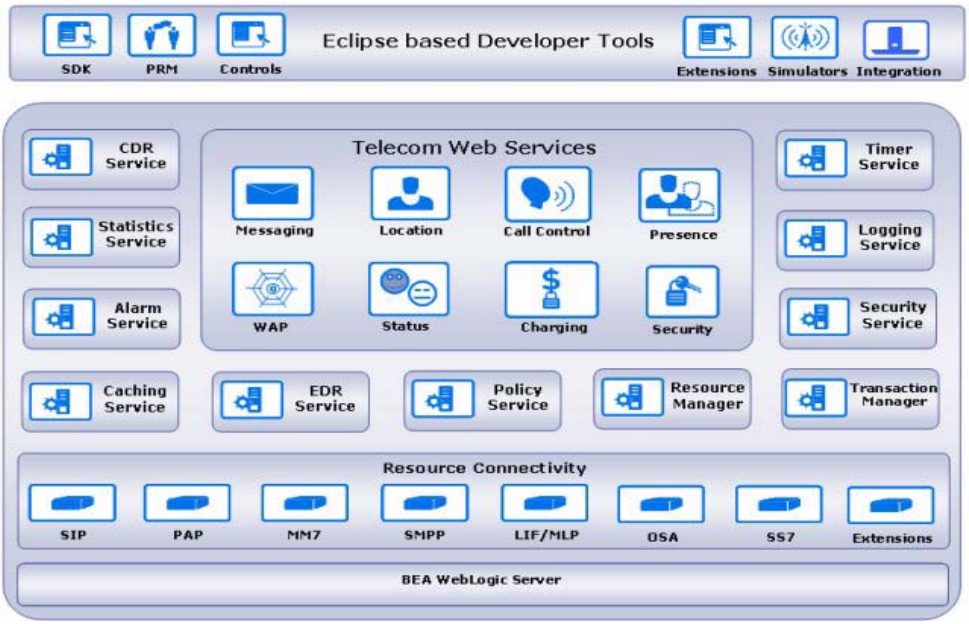
Introducing WebLogic Network Gatekeeper

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

Overview

The worlds of TCP/IP applications and of telephony networks continue to converge. But the relationship between them is often overly complex and difficult to manage. What is needed is a powerful, flexible, secure interface providing, on the one hand, a simple way for application developers to include telephony-based functionality in their software applications and, on the other, features that guarantee the security, stability, and performance required by network operators and demanded by their subscribers. BEA's WebLogic Network Gatekeeper is designed to do exactly this.

Figure 2-1 WebLogic Network Gatekeeper in Context



What Network Gatekeeper Provides

WebLogic Network Gatekeeper offers a host of benefits for both application developers and network operators.

APIs based on well-known Web Services standards

Third-party application developers can access standard network capabilities such as SMS or MMS through a set of Web Services-based interfaces - both Parlay X 2.1 standard and (for WAP Push) WebLogic Network Gatekeeper Extended - tailored to their needs. Because the interfaces are published in standard WSDL files, developers can use their choice of toolsets, increasing ease of production and reducing development time.

Robust security

WebLogic Network Gatekeeper leverages the flexible security framework of WebLogic Server 9.2. Applications can be authenticated using plaintext or digest passwords, X.509 certificates, or

SAML 1.0/1.1 tokens. XML encryption, based on the W3C's standard, can be applied, either to the whole SOAP request, or only to a portion of it. And, to ensure message integrity, requests can be digitally signed, using the W3C XML digital signature standard.

Common access control for both internal and 3rd party applications

A single point of contact providing common authentication, authorization, and access control procedures secures the integrity of the network.

Policy-based execution for flexible application authorization control

Policy-based authorization control built on dynamically customizable Service Level Agreement (SLA) data and other rules can be easily adjusted, shaping application access to fit operators' business models and their security requirements. The power of the Policy Engine can be accessed either as part of a traffic path or through the callable policy API.

Access to many standard telecom network service capabilities

Operators can easily provide applications with access to GSM, cdmaOne, cdma2000, 1xEVDO, WDCDMA, GPRS, SIP, IN, or 3G service nodes either directly through IP-based service nodes such as SMSCs or MMSCs or via OSA/Parlay gateways.

Access to WebLogic SIP server for connectivity to SIP network infrastructure

Calls set up using the Third Party Call traffic path can be routed either through standard OSA/Parlay protocols or through the SIP protocol, using WebLogic SIP server. Call Notification functionality is also supported on SIP. Presence *watchers* (consumers of presence information) can also be established.

Built-in network routing

Routing of service requests to appropriate network nodes can be based on address plans and actual destination address.

Extensible architecture

A flexible architecture using the robust capabilities of WebLogic Server 9.2 means that operators can both extend existing traffic paths to support new network interfaces, for example Unstructured Supplementary Service Data, and create entirely new traffic paths to expose unique network capabilities, should conditions warrant. This makes it easy to create attractive service offerings based on a network's particular features, using the Network Gatekeeper's Extension Toolkit.

Enhanced network protection

Because application service providers are assigned various priority levels, their network access can be managed accordingly, providing:

- Network node access control
- Network node traffic throttling

Integration with Operation Support Systems

All or selected parts of the Network Gatekeeper management application can be integrated with external Operation Support Systems through JMX/JMS or SNMP interfaces, simplifying administrative tasks.

Integration with Billing and Charging Systems

WebLogic Network Gatekeeper can be integrated with existing billing systems (prepaid, postpaid, and rating). The billing systems can then, if desired, be exposed to third party applications through a Payment (Content Based Charging) Web Services API or, if preferred, kept internal to Network Gatekeeper.

Carrier grade and fully scalable architecture

A highly distributed and replicated system design provides carrier grade performance.

Application Development Tools

To assist application developers, Network Gatekeeper provides:

- Web Services WSDL files

- A Developer Guide

In addition, as part of an optional module, Network Gatekeeper can also provide:

- The WebLogic Network Gatekeeper SDK
- The WebLogic Network Gatekeeper Simulator, a graphical test and verification environment for SMS, MMS, Terminal Location, and WAP Push traffic

Partner Relationship Management Interfaces

This optional module provides operators with interfaces to manage large sets of partners. The interfaces 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 interfaces 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.

CHAPTER 3

Software Architecture Overview

The following chapter provides an overview of WebLogic Network Gatekeeper's software architecture, including:

- Overview
- Traffic Paths
- Network Gatekeeper Core and Core Services
- Storage

Overview

The 3.0 release marks a substantial shift in WebLogic Network Gatekeeper's architecture. With this version, Network Gatekeeper has been ported to run on WebLogic Server 9.2, which provides a robust, secure, and highly performant environment for Network Gatekeeper's operation, and its overall structure has been more closely aligned with J2EE standards.

In general, Network Gatekeeper's internal functioning can be divided into three aspects:

- Traffic Paths
- Network Gatekeeper Core and Core Services
- Storage

Traffic Paths

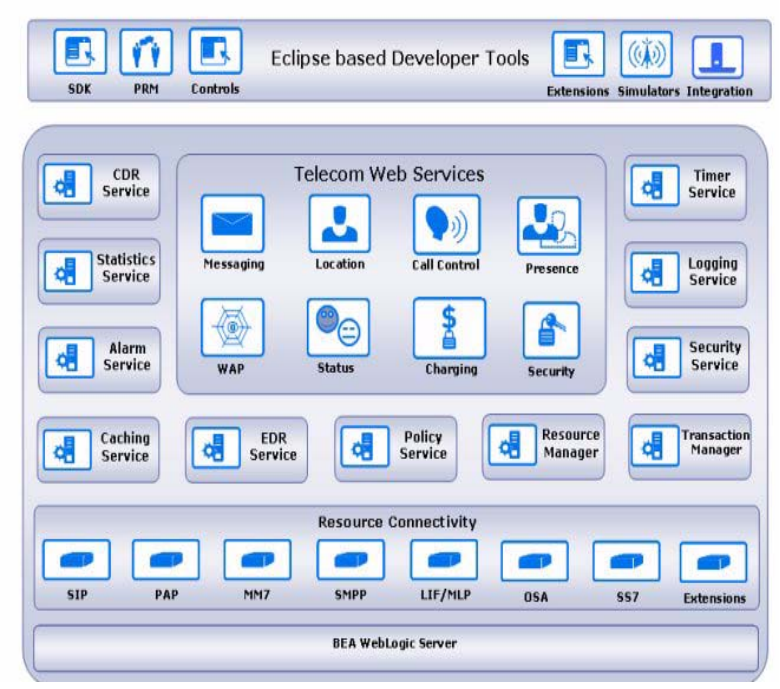
All traffic in Network Gatekeeper flows in traffic paths. A traffic path consists of an application-facing interface, with Web Services Security enforcement, a Service Capability, and a network plug-in, where requests are translated between the application-facing interface and underlying network node protocols. A more detailed description of this flow can be found in Chapter 4, "Introducing Traffic Paths."

Traffic paths are deployed in two clustered tiers, an Access Tier and a Network Tier. In a single physical site installation, this corresponds to a single WLS administration domain.

Note: This description covers the *enhanced* traffic path type. The *backwards compatible* traffic path type is similar, but not identical. For more information on backwards compatibility in Network Gatekeeper generally, see Chapter 11, "Backwards Compatibility 2.2 to 3.0."

Figure 3-1 below gives an overview of the basic traffic path structure.

Figure 3-1 Software architecture overview



Access Tier

The Access Tier handles communication with application service providers through its Web Services interfaces. It also provides authentication, confidentiality, and session management. Using the WLS Web Services Container, incoming SOAP messages are transformed into Java objects, which are then marshalled and sent to the Network Tier using RMI. No state is held in the Access Tier, so any Network Tier instance can use any Access Tier instance to make callbacks to application service provider clients. For added security, the Access and Network Tiers can be separated by a firewall.

Network Tier

The Network Tier provides request routing and protocol translation. It also manages policy enforcement and traffic throttling. Once a request enters the Network Tier, a transaction is started. This transaction mechanism makes sure that state is properly maintained should the request fail.

An application request enters the Network Tier through a Service Capability module, which is a very lightweight proxy. The Service Capability queries the Plug-in Manager for an appropriate plug-in, based on a set of criteria.

The plug-in has both a *north*, or application-facing, interface, and a *south*, or network-facing interface. Tasks common to all traffic paths are completed at these interfaces. For example, policy enforcement is managed at the north interface of the plug-in. As the request enters from the Access Tier, the request's parameters are bundled and sent to the Policy Engine, where they are evaluated according to SLA data and the *rules*. On the other hand, processing network-initiated requests, such as callbacks from underlying network nodes, requires special handling to be associated with the application that originally requested them. In this case code attached to the south interface handles this task.

Once the request enters the plug-in, it is translated into a form appropriate for its target. If a targeted network node requires that state be maintained - as, for example, the call state in the case of a Call Control request - it is maintained in the plug-in. If the underlying node does not require state to be held, Network Gatekeeper holds none. No state is ever held in the Access Tier.

Events and alarms that are raised in the Access Tier occur in the context of standard J2EE Web Applications, and are processed using standard WebLogic Server mechanisms. Because processing in the Network Tier uses the Network Gatekeeper Core and Core Services, Network Gatekeeper specific mechanisms have been introduced to capture such status information that arises in this tier. Event data, alarms, charging data, and usage statistics all begin as events, which are fired when predefined points are encountered or error conditions occur. These events are then sent through filters and delivered to listeners, which are divided by type. Out of the box, these

types include *Alarms*, *Event Data Records* (EDRs), and *Charging Data Records* (CDRs). For more information, see Events, Alarms, and Charging

Note: For the purposes of backwards compatibility, Network Gatekeeper 2.2 style events, alarms, and charging records, generated by the 2.2-based Event, Alarm, and Charging Services, can be published and delivered to 2.2 style, as well as 3.0 style, listeners, but this mechanism is deprecated in version 3.0.

Network Gatekeeper Core and Core Services

Previous versions of Network Gatekeeper ran in a Service Logic Execution Environment (SLEE) which supplied the necessary container-based services. The port of Network Gatekeeper to WebLogic Server 9.2 means that the many standard container services WLS provides can be leveraged in the running of Network Gatekeeper itself. Nonetheless, a number of Network Gatekeeper specific services continue to be required in Network Gatekeeper 3.0. These services make up the Network Gatekeeper Core module, and include such things as the Event Channel Service, Budget Service, Policy Manager, DB Manager, Geo-Redundancy Service, Plug-in Manager, and so on.

Note: Because these services originally existed as SLEE services, some of the names you see in the Web Logic Network Gatekeeper Console User Interface include the string *SLEE*. This is simply a legacy of the older code.

Storage

Modules running in Network Gatekeeper often need to store cluster wide and highly available information in order to function properly. In version 2.2, this was usually done using database tables, but in 3.0 a storage service has been introduced. This service takes care of traffic path storage needs for both short-lived and longer-lived data. For backwards compatibility Network Gatekeeper continues to expose the proprietary configuration store and global store APIs from 2.2.

CHAPTER 4

Introducing Traffic Paths

The following chapter presents an overview of Network Gatekeeper's traffic paths in general, including:

- Overview
  - How it works
  - Typical Application-initiated Traffic Flow
- Platform-wide Functionality
  - Service Level Agreements
  - Policy Enforcement and Policy Decision Points
  - Events, Alarms, and Charging
  - Statistics and Transaction Units (Licensing)
- Short Code Translation
- Traffic Path Types

A separate document, the Network Gatekeeper *Traffic Path Reference*, offers a more detailed look at specific paths.

Overview

All application request data flows through Network Gatekeeper on Traffic Paths. A traffic path consists of a service type (Messaging, User Location, etc.), an application-facing interface (also called a “north” interface), and a network-facing interface (also called “south” interface).

How it works

Network Gatekeeper is deployed into two tiers: the Access Tier, that manages interactions with applications, and the Network Tier, that contains the mechanisms necessary for dealing with the underlying network nodes. For increased security, these tiers can be separated by a firewall.

Applications begin their Network Gatekeeper sessions by logging into the Access Tier, using the Session Manager interface.This interface returns a Session ID. The application must add this Session ID to the header of all its subsequent SOAP requests. Network Gatekeeper uses this value to keep track of all the traffic that an application sends for the duration of the session.

Once the session has been established, the application can begin sending request traffic. Application-initiated requests (also called mobile terminated, or MT) enter through the Access Tier and are then sent on to the Network Tier. The Network Tier manages service authorization, charging, and traffic throttling. The Network Tier then translates the request into a form appropriate for the underlying network node.

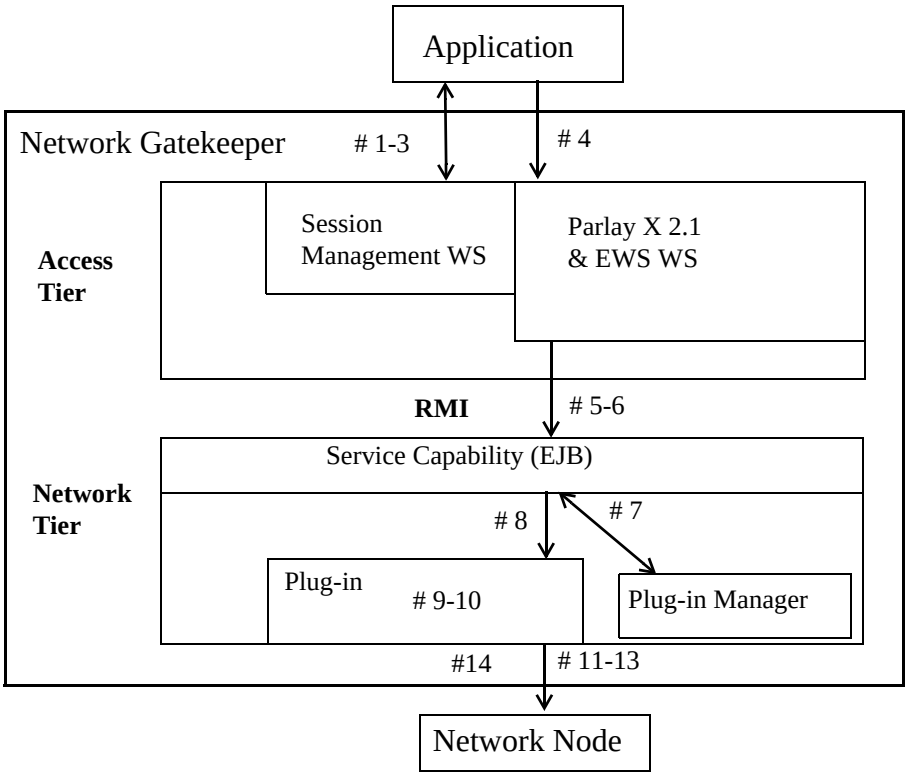
Network-initiated (also called mobile originated, or MO) traffic is also supported by Network Gatekeeper, so that applications can choose to receive data from the telecom network. To do so, the application must first send a request to Network Gatekeeper (or have the operator perform the equivalent task using OAM methods) to register a description of the types of data it is interested in - delivery notifications, incoming messages, etc. - and any criteria that the data must be meet to be acceptable. For example, an application might specify that it is only interested in receiving incoming SMSes that are addressed to the *short code* “12345” and that begin with the string “blue”.

Note: For more on short codes, see Short Code Translation

Typical Application-initiated Traffic Flow

Figure 4-1 below illustrates typical application-initiated traffic flow.

Figure 4-1 Typical Application-initiated Traffic Flow



1. An application establishes a session by using Network Gatekeeper's own Session Management Web Service in the Access Tier. Authentication is managed by WebLogic Server's WS-Security, and supports plaintext or digest passwords, X.509 certificates, or SAML tokens. The particular security requirements of the installation are specified in the WS-Policy section of the published WSDL file, and the request may be created using any toolset the application developer wishes to use.

   Note: It is possible to simply use the standard Parlay X 2.1 WSDL to create requests, but the developer would then be required to ascertain the appropriate security type from the operator, and insert the information manually.

2. Network Gatekeeper verifies that the maximum number of instances specified in the Service Level Agreement (SLA) for this application and its service provider has not been exceeded.
3. A session is established, and the SessionID is returned to the application. Once the application has been established, it may access multiple traffic paths across the cluster transparently.

   Note: The availability of the functionalities of various traffic paths, and the degree to which they are supported depends on the specifics of the installation, the protocols being used, and the types of network nodes to which they are connecting.
4. The session is valid until the application logs out or an operator-established time period has elapsed.
5. A request for a particular operation enters at the application-facing interface - implemented as a J2EE Web Service - in the Access Tier and is pre-processed. The SOAP envelope is removed and the request is transformed into a Java object.
6. The request is passed on to the Network Tier using RMI. The request enters the Network Tier through the Service Capability module, which is implemented as an EJB. The EJB serves as the entrance point for the Network Tier and provides the starting point for application-initiated transactions.
7. The request is evaluated for traffic throttling purposes.
8. The Service Capability queries the Plug-in Manager for an appropriate and available *network plug-in*. The plug-in will complete processing the request.
9. The request is sent to the plug-in the Manager returned. The bulk of the processing that the request undergoes takes place in the plug-in, and, as a result, most configuration tasks focus here.
10. Service authorization takes place in the plug-in, using the policy engine. The policy engine evaluates the request according to the rules. The rules are based on a set of SLAs (service provider and application specific). These rules indicate, for example, whether this particular application or service provider is authorized to use a specific telecom network node. The rules may also include any additional criteria the operator chooses to employ. In some cases the policy engine can also be used to alter method parameters, to add information or to make them compatible with the target network node. For more information, see Figure 4-3
11. The request is translated into the protocol suitable for the underlying network node. Any state information required by the underlying network node is kept within the plug-in.
12. The request is passed to the network.

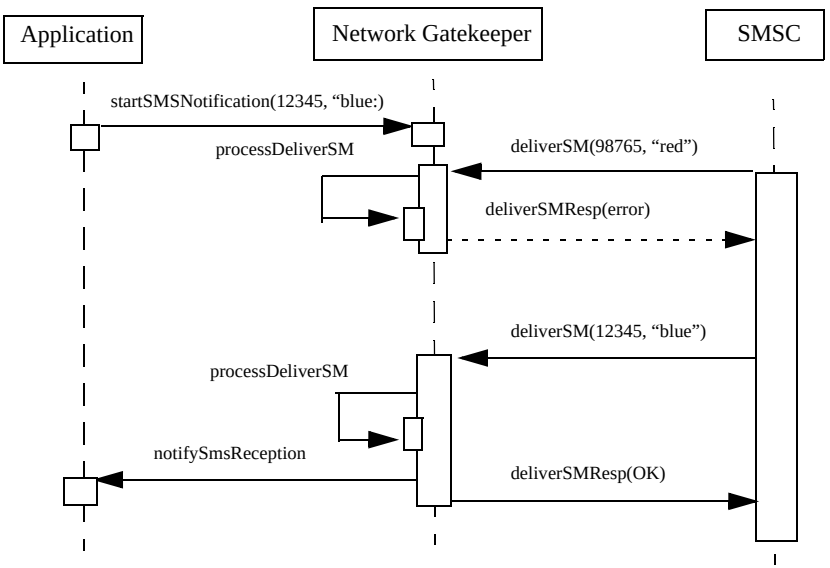
13. When the node acknowledges the request, charging data about the completed request are recorded.

14. The transaction commits.

Typical Network-initiated Traffic Flow

The key difference between application-initiated traffic flow and network-initiated traffic flow (other than the direction) is that the application must first indicate to Network Gatekeeper that it is interested in receiving traffic from the network. It does this by *registering for (*or *subscribing to) notifications*, either by sending a request to Network Gatekeeper or by having the operator set up the notification using OAM methods. In Figure 4-2 below, the application sends Network Gatekeeper a request to begin receiving SMSes from the network, indicating that it is only interested in SMSes that are sent to the address `12345` and that begin with the string `blue`. Not noted in the diagram, but also part of the request, is the URL of the Web Service that the application has implemented to receive these notifications back.

Figure 4-2 Registering for notifications



This registration for notifications is stored in the appropriate network plug-in, which in most cases passes it on to the underlying network node itself (in certain cases the Network Gatekeeper operator must do this manually.) When a matching SMS reaches the plug-in from the network,

the plug-in sends it off to the Policy Engine for evaluation, and then, using RMI, passes the notification, along with the appropriate URL from the registration, to the *callback EJB* in the Access Tier, which sends it to the Web Service implementation for processing, and then on to the application.

Platform-wide Functionality

Some functionality is common to all traffic paths. This functionality includes:

- Service Level Agreements
- Policy Enforcement and Policy Decision Points
- Security
- Events, Alarms, and Charging
- Statistics and Transaction Units (Licensing)

Service Level Agreements

All application access to WebLogic Network Gatekeeper's traffic paths is governed by a set of Service Level Agreements (SLAs) between the application service provider and the Network Gatekeeper operator. Network Gatekeeper uses a two tiered account system to track SLAs: Service Level Agreements are established per service provider group and, within that service provider group, per application group. For more information on the account system, see The Administration Model.

The following levels of data are specified in each type of SLAs:

SLA Level	Criteria
Service Provider	Traffic and charging related data, by service provider group. (For example, the authorized network capabilities, maximum bandwidth available, and so forth by service provider) It also specifies access to charging capabilities and revenue sharing schemas.
Application	Traffic and charging related data, by application group. (For example, the authorized network capabilities, maximum bandwidth available, and so forth by application group) It also specifies access to charging capabilities and revenue sharing schemas.

These SLAs govern a service provider/application's access to a particular traffic path, and regulate any Quality of Service (QoS) agreements into which the operator and the service provider may have entered, such as specifying the guaranteed number of requests a service provider may send through a particular traffic path in a given period of time. Specific enforcement can be tied to Time of Day/Day of Week, with differing Service Level Agreements in force based on the time at which the request is processed. As well, service access can be modulated both by rate (invocations per time period) and quota (aggregated number of invocations). For a detailed look at Service Provider and Application SLA structure, see the "Defining Service Provider Level and Application Level Service Agreements" chapter in *Managing Service Providers and Applications.* For a traffic path-focussed description, see the respective traffic path chapters in the *Traffic Path Reference.* These books are separate documents in the Network Gatekeeper documentation set.

In addition, there are also SLAs that help protect the underlying network node by setting priorities for sending requests. If a particular node is overloaded, lower priority traffic can be rejected all together. For more information on these Traffic (sometimes called Node) SLAs, The Administration Model. For more detailed information on these SLAs, see the "Managing and Configuring Routes and Node SLAs" chapter in the *System Administrator's Guide.*

Policy Enforcement and Policy Decision Points

Some SLA enforcement is handled by the traffic path itself, in conjunction with services provided by WebLogic Server. But most SLA enforcement and some additional policy enforcement takes place in Network Gatekeeper's policy engine. As request data enters the Network Plug-in, incoming request parameters are sent off to the policy engine for evaluation. Within the policy engine are Policy Decision Points, or PDPs, where the incoming requests are weighed based on rules based on data defined in the SLAs. Once the request is evaluated, it is returned to the Plug-in. It is also possible to use this process to modify the request's parameters, as, for example, by aliasing subscriber information to protect subscriber privacy or adding a rating to allow for Content Based Charging. All incoming parameters in an application's request are available for manipulation in this way.

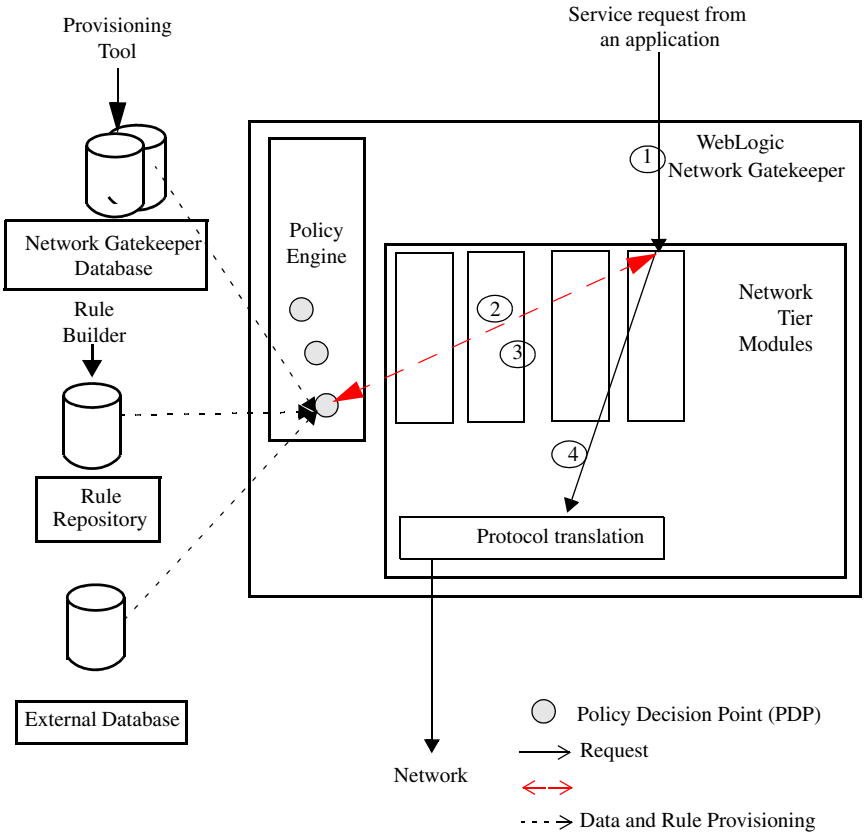
Backwards compatible and enhanced traffic paths present their parameters in slightly different ways, but there is a subset of parameters that all requests *may* include, depending on whether the the parameter has been mapped:

- String `applicationID`: The application ID associated with the request.
- String `serviceProviderID`: The service provider ID associated with the request.

- String `serviceName`: Service name from which the request originates or to which it is destined.
- String `methodName`: Method that triggered the request.
- String `requesterID`: Used only by the Extended Web Services interfaces.
- long `timeStamp`: Time the request was sent to the rules engine for processing. Milliseconds from start of UNIX epoch.

WebLogic Network Gatekeeper also supports the development of custom policies, that is, operator specific policies defined by the operator and implemented by BEA Systems or a selected partner. For more information on PDPs, see Figure 4-3 below.

Figure 4-3 Simplified traffic path policy execution flow



Note: Some network-triggered requests are also evaluated using the Policy Engine.

Security

WebLogic Network Gatekeeper secures the traffic that passes through it in seven ways:

- Authentication
- Authorization and service access
- Confidentiality and integrity

- Auditing and non-repudiation
- Network node authentication
- Database integrity
- Administrative access

Authentication

Network Gatekeeper uses special SOAP headers to authenticate service provider applications. These headers are documented in each interface's WSDL file, in the WS-Policy section. Processing is managed by WebLogic Server's WS-Security, which supports plaintext or digest passwords, X.509 certificates, or SAML tokens for authentication. For more information on WebLogic Server's capabilities, see the "Configuring Security" chapter in *Programming Web Services for WebLogic Server*, a separate document in the WLS set.

Authorization and service access

Access to a particular traffic path is based on the two types of SLAs discussed in Service Level Agreements.

Confidentiality and integrity

To guarantee the confidentiality of communication between WebLogic Network Gatekeeper and the application, all traffic can be encrypted - fully or partially - using W3C's standard XML encryption. Message integrity can be assured using the W3C XML digital signature standard. Again, the WS-Policy section of the published WSDL for each interface describes if and how either of these standards is being used.

Auditing and non-repudiation

Both successful and unsuccessful login attempts generate events/EDRs. All transactions are stored as CDRs in the database. Alarms are generated when service requests are denied due to SLA violations. As well, standard J2EE artifacts such as HTTP logs are generated.

Network node authentication

If the underlying network node provides an authentication interface, WebLogic Network Gatekeeper protocol plug-ins can register and be authenticated with it, making the request's transfer to the network secure.

Note: This is highly dependent on the protocol and the specific implementation in the node and the plug-in.

Database integrity

Access to WebLogic Network Gatekeeper database is protected by username and password combinations.

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

Administrative access

WebLogic Network Gatekeeper is administered through the WebLogic Server Console and the Network Gatekeeper extension within that console. Using the console requires a username and password, which are encrypted before being stored in the WebLogic Network Gatekeeper database. 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 administrative 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 operator’s OAM organization. One group consists of one or more logically related software modules.

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

Events, Alarms, and Charging

All WebLogic Network Gatekeeper modules can produce general events, alarms and charging events. General events are expected system occurrences that are of importance to the operator, but do not need corrective action. Alarms are system occurrences that are unexpected and may

require corrective action. Charging events are the basis for CDRs, the records that provide the information needed to charge for services.

Event handling in the Access Tier

The Access Tier runs in the WebLogic Server's Web Services Container, so events or alarms that are raised there can be monitored through standard JMX mechanisms or by using the WebLogic Diagnostics Framework.

See *Developing Manageable Applications with JMX* and *Configuring and Using the WebLogic Diagnostics Framework* for more information on how this works.

Event handling in the Network Tier

In the Network Tier, much of the functionality comes from the interaction between traffic paths and the Network Gatekeeper Core and Core Services. To capture this specialized level of information, Network Gatekeeper has developed specific mechanisms to record the data.

In enhanced traffic paths, all status information generated by the Network Tier - events, alarms, charging data, and usage statistics - begins as an event, which is fired whenever designated methods are called or exceptions are thrown. These events are then sent to the EDR Service. In the EDR Service, events are processed through XML-based filters, which provide the criteria by which the events are classified into types. The filters can also be used to transform the data in the original event, including adding other useful information. Once the information has been processed by the filters, it is delivered to type-specific listeners. Out of the box, there are three types of filters (edr.xml, cdr.xml, alarm.xml) that produce three distinct types of data: *Event Data Records* (EDRs), *Charging Data Records* (CDRs), and *Alarms*. All three of these filters can be customized as desired. These filters can also deliver desired event-based information to external JMS-based listeners. These listeners are set up as standard JMS topic subscribers and can be anywhere on the network.

Note: For the purposes of backwards compatibility, Network Gatekeeper 2.2 style events, alarms, and charging records, generated by the 2.2-based Event, Alarm, and Charging Services, can be published and delivered to 2.2 style, as well as 3.0 style, listeners, but this mechanism is deprecated in version 3.0.

See the *WebLogic Network Gatekeeper - System Administrator's Guide* for more information on setting up these filters.

Each 3.0 style EDR always includes:

EdrId	The type of EDR
ServiceName	The service type (SMS, Call Handling, etc.) that produced the event
ServerName	The name of the WLS host
Timestamp	The time at which the event was triggered (in milliseconds from midnight 1 January 1970)
ContainterTransactionID	The transaction ID from WebLogic Server, if available. This identifies the thread on which the request is executed
Class	The name of the class that logged the event
Method	The name of the method that logged the event
Source	The kind of event. There are two possible values for this field: • Method: the event was fired in relation to a method call • Exception: the event was fired in relation to an exception being thrown

In addition, most events include:

Direction	The direction in which the request is traveling. There are two possible values for this field: • **South**: traveling toward the network node • **North:** traveling toward the application
Position	The position of the EDR relative to the method that logged the EDR. There are two possible values for this field: • **Before**: the event occurred before the method • **After**: the event occurred after the method

Interface	The interface at which the EDR is logged. There are three possible values for this field: • **North**: the event was logged at the north plug-in interface • **South**: the event was logged at the south plug-in interface • **Other**: the event was logged someplace other than the north or south interfaces
Exception	The name of the exception that triggered the EDR
SessionId	The application's session identifier
ServiceProviderId	The service provider account identifier
ApplicationId	The application account identifier
AppInstanceGroupId	The authentication user name of the Application Account. This is a string that is equivalent to the 2.2 value: Application Instance Group ID
DestAddress	The destination address. If this is a send list, the first address will be listed here. Additional addresses are stored in the AdditionalInfo field.
AdditionalInfo	Variable information depending on the traffic path. Stored as "key=value\n" pairs.

Alarm handling

Network Tier alarms are those events that are of immediate interest to the operator. These are either 3.0 style EDRs that are defined via filters created in the internal alarm configuration file or 2.2 style alarms from the deprecated Alarm Service. While each 3.0 style alarm begins as an EDR, not all the information available in the EDR is stored when the alarm is written to the database (although that information can be retrieved using an external listener). Each alarm entry in the database includes the following information:

alarm_id	A unique sequential identifier
source	The name of the software module that raised the alarm and the IP address of the server in which the module runs. This is *not* the same as the Source field in the event
timestamp	The time at which the event was triggered (in milliseconds from midnight 1 January 1970)
severity	The importance of the alarm. There are four possible values for this field: • warning • minor • major • critical
identifier	The alarm type
alarm_info	Information provided by the module that raised the alarm
additional_info	3.0 style alarms only. This field includes: • Service Provider ID • Application ID • Application Instance Group ID • Other information depending on context

Management integration

Network Gatekeeper supports integration of its alarm and event mechanisms with external management tools.

OSS

An Operation Support System (OSS) can integrate with WebLogic Network Gatekeeper alarm and event services through the creation of external JMS listeners. As well, integration can be managed via OAM scripts through the use of JMX-based tools.

SNMP

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

Charging Data Records

Charging Data Records can originate either as filtered 3.0 style EDRs or as 2.2 style CDRs from the deprecated Charging Service. While each 3.0 style CDR begins as an EDR, not all the information available in the EDR is stored when the CDR is written to the database (although that information can be retrieved using an external listener). Each CDR entry in the database includes the following information:

transaction_id	The Network Gatekeeper transaction sequence number
service_name	The traffic path whose use is being tracked
service_provider	The Service Provider ID
application_id	The Application ID
application_instance_id	The login user name of the Application Account. This is a string that is equivalent to the 2.2 value: Application Instance Group ID
container_transaction_id	3.0 style CDRs only.The transaction ID from WebLogic Server, if available. This identifies the thread on which the request is executed
server_name	3.0 style CDRs only. The name of the server in which the CDR was generated
timestamp	3.0 style CDRs only. The time at which the event was triggered (in milliseconds from midnight 1 January 1970)
service_correlation_ID	3.0 style CDRs only. An identifier that allows the usage of multiple service types to be correlated into a single charging unit
charing_session_id	An ID correlating related transactions within a service capability module that belong to one charging session. For example, a call containing three call-legs will produce three separate transactions within the same session. Not necessarily the same as the SessionId, the application session identifier, although the traffic path could choose to use the SessionId for this purpose.

start_of_usage	The date and time the request began to use the services of the underlying network.
connect_time	The date and time the destination party responded. Used for Call Control traffic only.
end_of_usage	The date and time the request stopped using the services of the underlying network.
duration_of_usage	The total time the request used the services of the underlying network.
amount_of_usage	The used amount. Used when the charging is not time dependent, as in, for example, flat rate services.
originating_party	The originating party's address.
destination_party	The destination party's address. This is the first address in the case of send lists, with all additional addresses placed in the additional_info field.
charging_info	A service code added by the application or by the policy service.
additional_info	If the traffic path supports send lists, all destination addresses other than the first, under the key “destinationParty”. Any information from 2.2 style additional_info, under the key “oldInfo”. In addition any additional information provided by the traffic path
revenue_share_percentage	2.2 style CDRs only. Kept for backwards compatibility and deprecated for 3.0
party_to_charge	2.2 style CDRs only. Kept for backwards compatibility and deprecated for 3.0
slee_instance	2.2 style CDRs only. Kept for backwards compatibility and deprecated for 3.0
network_transaction_id	2.2 style CDRs only. Kept for backwards compatibility and deprecated for 3.0
network_plugin_id	2.2 style CDRs only. Kept for backwards compatibility and deprecated for 3.0

transaction_part_number	2.2 style CDRs only. Kept for backwards compatibility and deprecated for 3.0
completion_status	2.2 style CDRs only. Kept for backwards compatibility and deprecated for 3.0

Statistics and Transaction Units (Licensing)

Licensing for WebLogic Network Gatekeeper is based on a maximum allowed rate (measured in *transaction units per second* or TUPS) during a specific time period per 24-hour interval. Two TUPS rates are measured: Base Platform - the more general rate - and BEA Module - which covers only Network Gatekeeper-supplied traffic paths. For more information on how these rates are calculated, see *WebLogic Network Gatekeeper - Licensing*, a separate document in this set.

Short Code Translation

A common feature of Messaging capable networks is the use of short codes and message prefixes to help route traffic and to make access to certain features easier for the end user. Instead of having to use the entire address, users can enter shorter sequences when they dial, which are then mapped to the full address in the network. WebLogic Network Gatekeeper supports short codes and message prefixes, which allow the same short code to be mapped to multiple addresses, based on what is prepended to the enclosed message.

Traffic Path Types

This release of WebLogic Network Gatekeeper marks a significant change in the basic architecture of the core Network Gatekeeper product, including a port to WebLogic Server. As a result of these changes, some traffic paths now exist in a somewhat reformulated, but backwards compatible version of themselves while others have been completely rewritten and exist in fully Network Gatekeeper 3.0, enhanced versions. In terms of processing traffic, the two types are essentially the same, but there are some small differences in OAM procedures. See the *WebLogic Network Gatekeeper - System Administrator's Guide* for more information.

Network Gatekeeper operators who have created extension traffic paths or network plug-ins will need to make certain changes in their code to allow those extensions to run in the new environment. The necessary changes are covered in some detail in the *Extension Toolkit Developer's Guide*, a separate document in the Toolkit set. New extensions should be written using the enhanced architecture.

Parameter Tunneling (MP01)

Beginning with version 3.0 MP01, a mechanism has been put in place to allow applications to include parameters that are not specified in the application-facing interface in their requests. This functionality can be used to supply additional information to the underlying network. The mechanism involves adding the additional information to the header of the SOAP request. The traffic path retrieves the tunneled parameters and inserts them into the request before it goes to the network.

To use parameter tunneling, the traffic path must be created specifically to support it. The only out-of-the-box traffic path in Network Gatekeeper 3.0 MP01 that has this capability is the SMS to SMPP traffic path (see the *Traffic Path Reference* for detailed information). Extension traffic path creators, however, can add this functionality as the needs of their system require.

CHAPTER 5

Developing and Testing Applications

The following sections introduce developing applications to interact with WebLogic Network Gatekeeper:

- Overview
- References
- Tools

Overview

Network Gatekeeper provides application developers with two types of easy to use Web Services APIs, those based on the Parlay X 2.1 standard and an additional one to cover WAP Push functionality, which is not supported by Parlay X. These interfaces include:

- Third Party Call
  Using this traffic path, an application can set up a call between two parties (the caller and the callee), poll for the status of the call, and end the call.
- Audio Call
  Using this traffic path, an application can set up a call to a telephone subscriber and then, when the subscriber answers, play an audio message, such as a meeting reminder.
- Call Notification
  Using this traffic path, an application can set up and end notifications on call events, such as the callee in a third party call attempt is busy. In addition, in some cases the application can then reroute the call to another party.

- Call Handling

  Using this traffic path, an application can establish rules that will automatically handle calls that meet certain criteria. These rules might establish, for example, that calls from a particular number are always blocked, or are always forwarded if the initial callee is busy. In addition, the application can retrieve rules that are currently in place.

- Short Messaging

  Using this traffic path, an application can send SMS text messages, ringtones, or logos to one or multiple addresses, set up and receive notifications for final delivery receipts of those sent items, and arrange to receive SMSes meeting particular criteria from the network.

- Multimedia Messaging

  Using this traffic path, an application can send Multimedia Messages to one or multiple addresses, set up and receive notifications for final delivery receipts of those sent items, and arrange to receive MMSes meeting particular criteria from the network.

- Terminal Status

  Using this traffic path, an application can request the status (reachable, unreachable, or busy) of one or more terminals and set up and receive notifications for a change in status for particular terminals.

- Terminal Location

  Using this traffic path, an application can request the position of one or more terminals or the distance between a given position and a terminal. It can also set up and receive notifications based on geographic location or time intervals.

- Presence

  Using this traffic path, an application can be a *watcher* for presence information published by a *presentity*, an end user who has agreed to have certain data, such as current activity, available communication means, and contact addresses made available to others. So a presentity might say that at this moment he is in the office and prefers to be contacted by SMS at this number. Before the watcher can receive this information, it must subscribe and be approved by the presentity. Once this is done, the watcher can either poll for specific presentity information, or set up status notifications based on a wide range of criteria published by the presentity.

- Payment

  Using this traffic path, an application can communicate charging information to an operator in situations where the cost of the service is based on the nature of the content delivered and not on connect time. For example, an end user could request the download of a music video, which costs a specific amount. The application can notify the operator that the user should be charged a particular amount or be refunded a particular amount. In

the case of pre-paid accounts, it can also reserve a certain amount of the user's available funds and then charge or release the reservation depending, say, on whether or not the download was successful.

- WAP Push

  The application-facing interface of this traffic path, unlike the previous ten, is not based on the Parlay X 2.1 specification. Many elements within it, however, are based on widely distributed standards. Using this traffic path, an application can send a WAP Push message, send a replacement WAP Push message, or set up status notifications about previously sent messages.

References

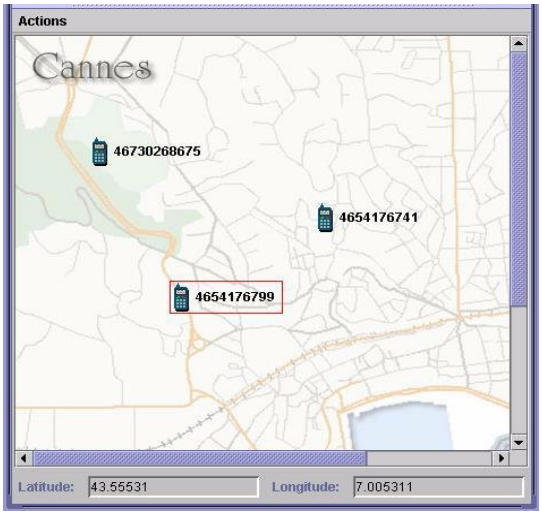
WebLogic Network Gatekeeper ships with the *WebLogic Network Gatekeeper - Application Development Guide*, which covers both the APIs themselves and some additional information an application developer needs to create applications that work with Network Gatekeeper. Because all of the APIs are Web Services based, applications can be developed using any environment that the developer chooses.

Tools

As an option, application developers for WebLogic Network Gatekeeper can also access a set of tools created to ease the development process, the WebLogic Network Gatekeeper SDK, including the Network Gatekeeper Simulator, a GUI-based testing environment for MMS, SMS, WAP Push, and Terminal Location applications.

The WebLogic Network Gatekeeper Simulator is an interactive, graphical test environment in which telecom-enabled applications designed to use WebLogic Network Gatekeeper can be tested. The current version supports MMS, SMS, WAP Push, and Terminal Location applications. Applications use the interfaces in the test environment just as they would the interfaces on Network Gatekeeper.

Figure 5-1 WebLogic Network Gatekeeper Simulator



The testing interface of the Simulator consists of a GUI which displays a map. The map can be changed to represent different geographical areas. Mobile terminals representing the application's end users are added to the map and given a phone number. These terminals can then be used as testing targets, sending and receiving messages, and querying for location. Once the terminals have been defined, they can be moved to different locations on the map.

Also included with the Simulator is a developer's copy of WebLogic Server, the environment in which the Simulator runs. For more on what is needed to use the SDK and run the Simulator, see the *SDK User Guide*, a separate volume in this document set.

CHAPTER 6

Managing Application Service Providers

The following sections describe the framework for managing service providers and applications:

- Overview
- The Administration Model
- Partner Relationship Management Interfaces
- Other Tasks Associated with Administering Service Providers

Overview

Managing partner relationships is key to the successful convergence of third-party application services and telecom network operations. WebLogic Network Gatekeeper provides a partner administration model to help operators handle the needs and demands of their partners in a flexible and powerful way:

- Application service providers are registered with WebLogic Network Gatekeeper, by service provider account and application account.
- Each account type is associated with a group that is tied to a Service Level Agreement that defines its access to both Network Gatekeeper and underlying network nodes.

The service provider and application registration are performed either internally through the WebLogic Network Gatekeeper Management Console or through external management systems integrated with WebLogic Network Gatekeeper using the Network Gatekeeper Partner Relationship Management Interfaces.

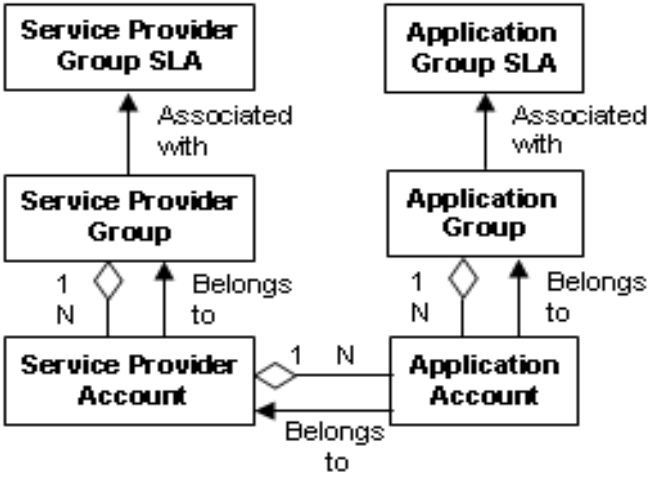
The Administration Model

The WebLogic Network Gatekeeper administration model allows operators to manage application service provider access at increasingly granular levels of control. An application service provider registers with WebLogic Network Gatekeeper and is given a service provider account. To support tiering, service provider accounts are associated together into account groups. These groups are then associated with their own Network Gatekeeper Service Level Agreements.

Within a service provider account are individual application accounts, registered on their respective service provider accounts. As in the case of service provider accounts, these application accounts are grouped together into account groups, each of which is associated with its own SLA.

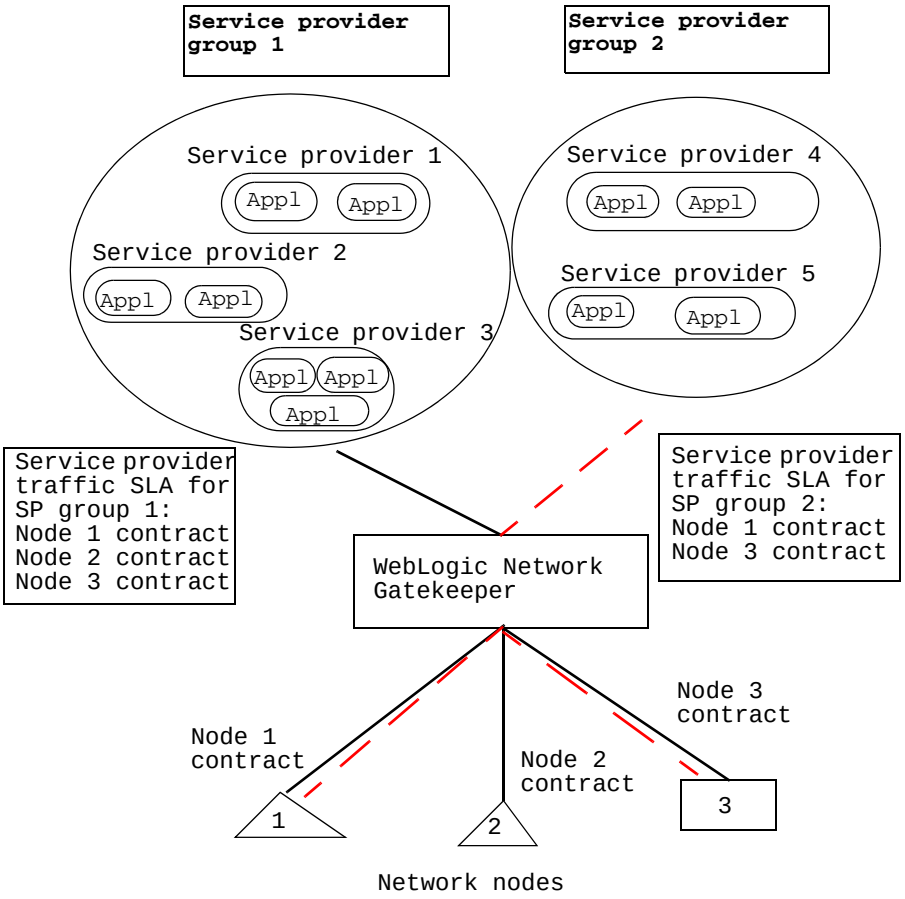
Network Gatekeeper SLAs on the service provider and application level regulate, for example, the type of service capability made available, the maximum bandwidth use allowed and the number of concurrent sessions supported. They may also specify access to charging capabilities and revenue sharing schemas. See Figure 6-1 for more information.

Figure 6-1 Service Provider and Application Administration Model



In addition to Network Gatekeeper SLAs, WebLogic Network Gatekeeper supports two types of traffic SLAs, contracts designed to protect the underlying telecom network. Service provider traffic SLAs regulate the relationship between a service provider group and the network nodes to which it has access. See Figure 6-2 for more information.

Figure 6-2 Service Provider Traffic SLAs



In Figure 6-2 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.

The second type of traffic SLA, the total traffic SLA, regulates the relationship between the Network Gatekeeper itself and the underlying nodes.

Partner Relationship Management Interfaces

The WebLogic Network Gatekeeper Partner Relationship Management Interfaces provide support for the automation of the traditionally work intensive tasks related to service provider and application administration, including supporting workflows, with a request/approve model. Most of the work of registration can be shifted to the service provider, as the operator's role changes from that of entering registration data to that of approving registration data. Large numbers of service provider and application accounts can be managed without increasing administration overhead. Service providers are also provided with a defined and structured channel to communicate desired account changes and to retrieve usage statistics for the accounts.

For a detailed description of the Partner Relationship Management Interfaces, see the document *Integration Guidelines for Partner Relationship Management for WebLogic Network Gatekeeper*.

As a part of an integration project, the Partner Relationship Management Interfaces can also be integrated with back end systems and network nodes, such as SMSCs, MMSCs, and pre-paid systems for creating and updating accounts.

Other Tasks Associated with Administering Service Providers

For an application to use content based charging or messaging service capabilities, mechanisms internal to Network Gatekeeper must be set up. To use user interaction service, announcements must be recorded and installed in the network. For more information on these areas, see *WebLogic Network Gatekeeper - System Administrator's Guide*

CHAPTER 7

Managing Network Gatekeeper: OAM

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

- Overview
- The WLS and Network Gatekeeper Management Console
- OAM Tasks Overview
- OSS Integration

Overview

WebLogic Network Gatekeeper is usually controlled through the WebLogic Network Gatekeeper Management Console, a specialized extension of the general WebLogic Server Console. The Console is a web-based tool, and can be run in any environment that supports appropriate web browsers. For general information on the WLS Console, see the "Overview of the Administration Console" chapter of the *Introduction to BEA WebLogic Server and BEA WebLogic Express*™. For some tasks, you can also use scripts that run in the Web Logic Scripting Tool. For general information, see *WebLogic Scripting Tool.* In addition, all or selected parts of the management application can be integrated with external Operation Support Systems (OSS) using JMX/JMS and alarms can be distributed using SNMP traps. And the application service provider management tool functionality can be integrated with PRM and CRM systems using the Network Gatekeeper Partner Relationship Management Interfaces.

Administrative users can be divided into user groups with access to different aspects of the administrative functionality. Within user groups, individual users can have differing levels of access. See *WebLogic Network Gatekeeper - System Administrator's Guide* for more information.

The WLS and Network Gatekeeper Management Console

The BEA WebLogic Network Gatekeeper Management Console is a Web browser-based, graphical user interface that you use to manage a WebLogic Server domain. A standard production installation for WebLogic Network Gatekeeper consists of at least one WebLogic Server domain.

One instance of WebLogic Server in each domain is configured as an Administration Server. The Administration Server provides a central point for managing a WebLogic Network Gatekeeper domain. All other server instances in the domain are called Managed Servers. In Network Gatekeeper, they are divided into Access Tiers and Network Tiers. In a domain with only a single WebLogic Server instance, that server functions both as Administration Server and both Managed Servers. The Administration Server hosts the Administration Console, which is a Web application accessible from any supported Web browser with network access to the Administration Server. To access the console, use the following URL:

```
http://hostname:port/console
```

where `hostname` is the DNS name or IP address of the Administration Server and `port` is the listen port on which the Administration Server is listening for requests.

OAM Tasks Overview

Use the Administration Console to:

- Configure, start, and stop Network Gatekeeper instances
- Configure Network Gatekeeper clusters
- Configure Network Gatekeeper services, such as database connectivity (JDBC) and messaging (JMS)
- Configure security parameters, including managing users, groups, and roles
- Monitor server and application performance
- View server and domain log files

- View application deployment descriptors
- Edit selected runtime application deployment descriptor elements

Use the WebLogic Network Gatekeeper specific section (accessed through the Domain Structure tree on the left side of the Administration Console) to:

- Configure Network Gatekeeper traffic paths
- Provision Application Service Providers
- Monitor alarms

Tasks performed outside the Console

- Extend Network Gatekeeper's functionality
- Backup and restore the system
- Upgrade the system

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

OSS Integration

All or selected parts of the management application can also be integrated with external Operation Support Systems (OSS) through secured JMX/JMS interfaces. For more information on working with JMX, see *Developing Manageable Applications* and *Configuring and Using the WebLogic Diagnostic Framework.* Alarm supervision systems can set up external JMS listeners to receive user definable types of event-based data, including standard alarms. For more information on using JMS listeners, see Events, Alarms, and Charging. SNMP traps are sent to any registered SNMP managers.

CHAPTER 8

Charging

The following sections describe WebLogic Network Gatekeeper charging functionality:

- Overview
- CDR-based Charging
- Content Based Charging and Accounting
- Billing System Integration

Overview

WebLogic Network Gatekeeper makes it possible to tailor the type of charging associated with each application service. An application can use one or more of the following alternatives:

- Charging based on time used or per-use services (CDR based)
- Charging based on the content or value of the used service (CBC)

CDR-based Charging

CDRs are used for charging based either on time used or on access to certain per-use services. Charging based on time used is typically employed for calls. Per-use might be employed, for example, to charge for a positioning service.

CDR data can be stored in WebLogic Network Gatekeeper's internal charging database or retrieved in real-time by billing and post processing systems through a billing gateway (this requires integration with the billing gateway: see Billing System Integration).

Data Generation

Charging data is generated every time an application uses a traffic path. The charging data is recorded by the traffic path during the period the application interacts with the network. When the interaction is closed, the traffic path stores the charging data as a CDR in the Network Gatekeeper's database. (If WebLogic Network Gatekeeper is integrated with a billing gateway, the charging data is sent directly to the billing gateway.) For an overview of what is stored in a CDR in the database, see Charging Data Records

Content Based Charging and Accounting

Content or value based charging makes it is possible to charge a user or subscriber based on the variable value of a used service rather than on time used or flat rates. This can be used, for example, when down loading music video clips or in m-commerce applications. WebLogic Network Gatekeeper supports both pre-paid and post-paid end-user accounts using various mechanisms.

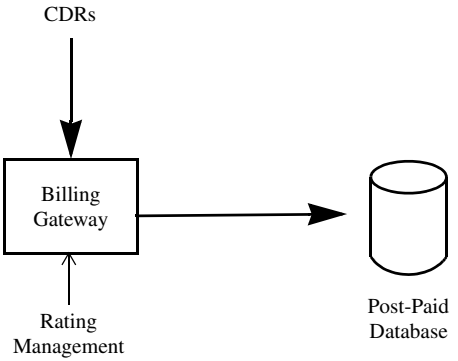
Billing System Integration

Network Gatekeeper can be integrated with external billing systems, either those that receive charging data directly or those that automatically retrieve information from Network Gatekeeper’s database. CDRs can be customized to fit the requirements of these systems, both in terms of format and behavior.

Billing gateways

Real-time settlement of pre-paid accounts using CDR based charging requires integration through a billing gateway. This method can also be used to support post-paid services.

Figure 8-1 Billing integration through billing gateway

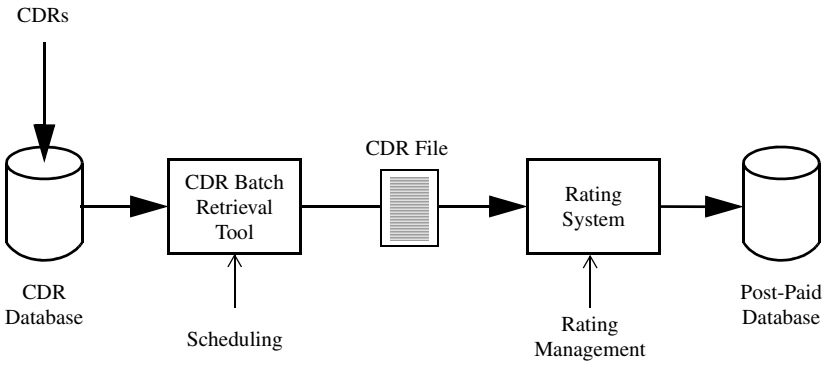


When integrating through a billing gateway, the billing gateway retrieves the CDRs in real-time through an external JMS-based charging listener. Rating, rating management, billing information storage, and pre-paid accounts settlement are handled by the billing gateway. The flow is shown above in Figure 8-1.

CDR database

In the case of applications that use post-paid accounts, it is possible to integrate billing by retrieving CDRs that have been stored in the Network Gatekeeper database.

Figure 8-2 Billing integration using the database



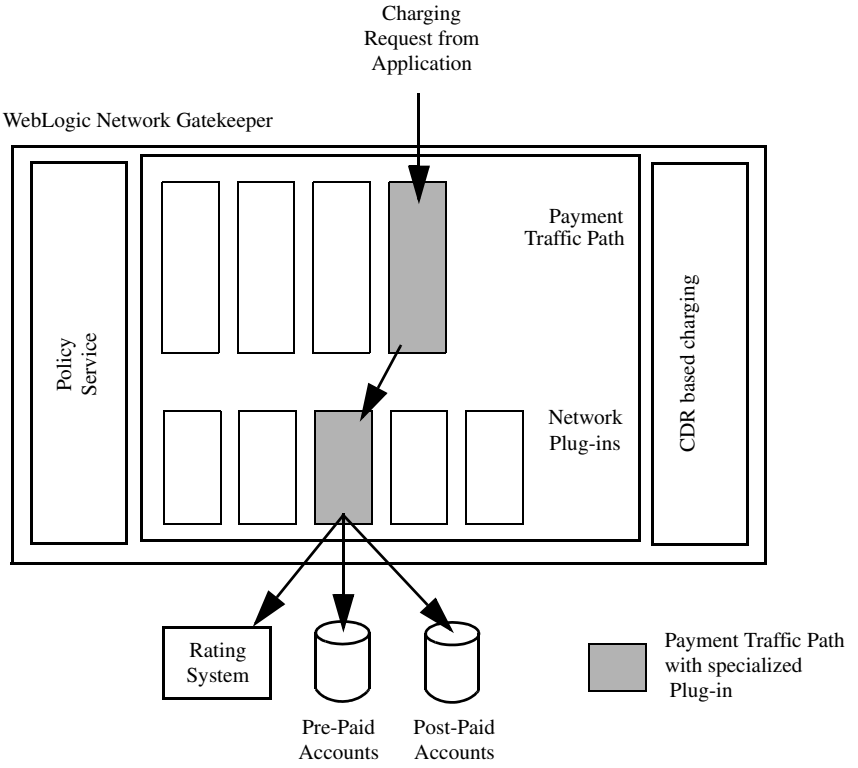
When integrating using this method, a CDR batch retrieval tool retrieves the CDRs from the database and stores them in a file format. The CDR file is processed by a rating system that transforms it into billing information and then stores it in a post-paid accounts database. The flow is shown above in Figure 8-2.

Payment plug-in

Applications using the Payment traffic path (for Content Based Charging) need to have a connection with an accounts database. The integration between the Payment traffic path and the database is made via a customized plug-in, developed to fit the characteristics of the accounts database. Both pre-paid and post-paid accounts can be handled using this method. For example, it is possible to check to make sure that there are funds available in a pre-paid account before a call is set up.

The flow is shown below in Figure 8-3

Figure 8-3 Billing integration through a Payment plug-in.



CHAPTER 9

Redundancy, Load Balancing, and High Availability

Redundancy, load balancing and high availability are essential for true carrier grade performance. WebLogic Network Gatekeeper uses both software and hardware components to achieve these important ends:

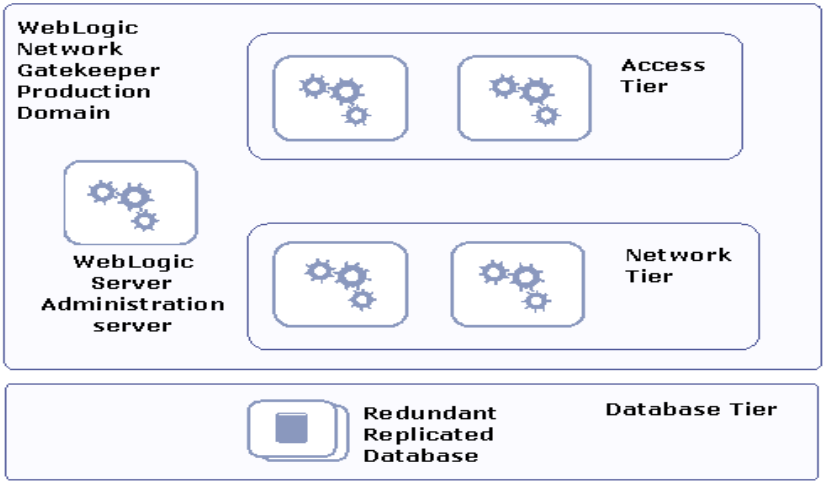
- Tiering
- Traffic Management Inside Network Gatekeeper
  - Application-initiated Traffic
  - Network-triggered Traffic
- Registering Notifications with Network Nodes
  - Network Node Supports Primary and Secondary Notification
  - Network Node Supports Only Single Notification
- Network Configuration
- Geographic Redundancy

WebLogic Network Gatekeeper's high availability mechanisms are supported by the clustering mechanisms made available by its container, WebLogic Server. For general information about WebLogic Server and clustering, see *Using WebLogic Server Clusters*.

Tiering

For both high availability and security reasons, Network Gatekeeper is split into two tiers: the Access Tier and the Network Tier. Each tier consists of a cluster, with at least two server instances per cluster, and all server instances run in active mode, independently of each other. The servers in both clusters are, in the context of WebLogic Server, *managed* servers. Together the clusters make up a single WebLogic Server administration domain, controlled through an administration server.

Figure 9-1 Example Production Domain



Communication between the Access Tier and the Network Tier takes place using Java RMI. Application requests are load-balanced between the Access Tier and the Network Tier and failover mechanisms are present between the two. See Traffic Management Inside Network Gatekeeper for more information on these mechanisms in application-initiated and network-triggered traffic flows.

There is an additional tier containing the database. Within the cluster, data is made highly available using a cluster-aware storage service which ensures that session state data is made available among Network Tier instances since multiple invocations can relate to the same session.

Traffic Management Inside Network Gatekeeper

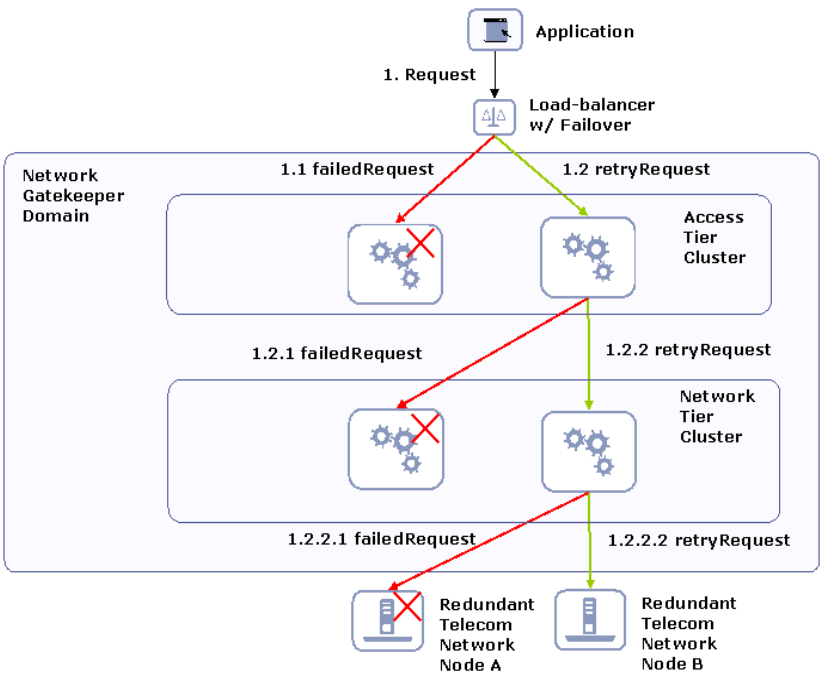
Potential failure is possible at many stages along the path that traffic follows as it moves through Network Gatekeeper. The following sections detail, tier by tier, how Network Gatekeeper deals with problems that might arise in both application-initiated and network-triggered traffic.

Application-initiated Traffic

Application-initiated traffic consists of all requests that travel from applications through Network Gatekeeper to underlying network nodes.

The example below follows the worst-case scenario for application-initiated traffic as it passes through Network Gatekeeper, and the failover mechanisms that attempt to keep the request alive.

Figure 9-2 Failover mechanisms in application-initiated traffic



1. The application sends a request to Network Gatekeeper. In a production environment, this request is routed through a hardware load balancer, usually protocol-aware. If the request towards the initial Access Tier server fails (1.1), either a time-out or a failure is reported. The load-balancer, or the application itself, is responsible for retrying the request.
2. The request is retried on a second server in the cluster (1.2) and it succeeds. It then attempts to send the request on to the Network Tier.
3. The request either fails to reach the Network Tier or fails during the process of marshalling/unmarshalling the request as it travels to the Network Tier server (1.2.1).
4. A fail-over mechanism in the Access Tier sends the request to a different server in the Network Tier cluster and it succeeds (1.2.2). It then attempts to send the request on to the network node.

   Note: If the request fails *within* the Network Tier, failover does not occur. In this case, an exception is thrown to the application, which can then re-send the request.
5. The attempt to send the request to the telecom network node fails (1.2.2.1).
6. If a redundant pair of network nodes exists, the request is forwarded to the redundant node (1.2.2.2). If this request fails, the failure is reported to the application.

Network-triggered Traffic

Network-triggered traffic can consist of the following:

- Requests that contain a payload, such as terminal location or an SMS
- Acknowledgements from the underlying network node that an application-initiated request has been processed by the network node itself. A typical example might indicate that an SMS has reached the SMSC. From an application's perspective, this is normally processed as part of a synchronous request, although it may be asynchronous from the point of view of the network
- Acknowledgements from the underlying network node that the request has been processed by the destination end-user terminal; for example, an SMS delivery receipt indicating that the SMS has been delivered to the end-user terminal. From an application's perspective, this is normally handled as a incoming notification

For network-triggered traffic, Network Gatekeeper relies heavily on the telecom network node, or other external artifacts such as load-balancers with failover capabilities, to do failover.

In the case of network nodes that can handle the registration of multiple callback interfaces, such as a Parlay Gateway, Network Gatekeeper registers one primary and one secondary callback

interface. If the Parlay Gateway is unable to send a request to the network plug-in registered as the primary callback interface, the Parlay Gateway is responsible for retrying the request, sending it to the plug-in that is registered as the secondary callback interface. This secondary callback interface is found in a network plug-in residing in another Network Tier instance. The plug-ins are responsible for communicating with each other and making sure that both callback interfaces are registered. See Network Node Supports Primary and Secondary Notification below for more information.

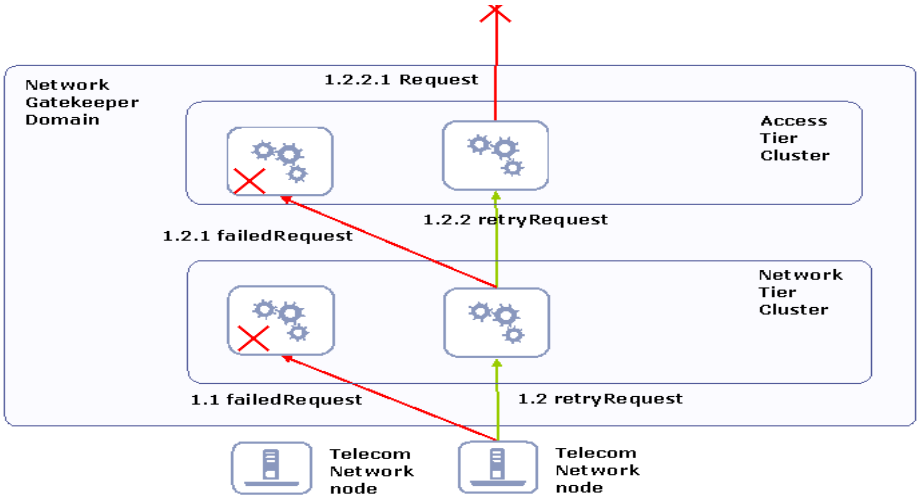
For HTTP-based protocols, such as MM7, MLP, and PAP, Network Gatekeeper relies on an HTTP load balancer with failover functionality between the telecom network node and Network Gatekeeper. See Network Node Supports Only Single Notification below for more information.

If a telecom network protocol does not support load balancing and high availability, a single point of failure is unavoidable. In this case, all traffic associated with a specific application is routed through the same Network Tier server and each plug-in has one single connection to one telecom network node.

The worst-case scenario for network triggered traffic for medium life span notifications using a network node that supports primary and secondary callback interfaces is described below.

Note: For more information on life spans, see Registering Notifications with Network Nodes.

Figure 9-3 Failover mechanisms in network-triggered traffic



1. A telecom network node sends a request to the Network Gatekeeper network plug-in that has been registered as the primary. It fails (1.1) due to either a communication or server failure.
2. The telecom network node resends the request, this time to the plug-in that is registered as the secondary call-back interface. This plug-in is in a different server instance within the Network Tier cluster.
3. The Network Tier attempts to send the message to the callback EJB in the Access Tier. It fails (1.2.1)
4. If the request fails to reach the Access Tier, or failure occurs during the marshalling/unmarshalling process, the Network Tier retries, targeting another server in the Access Tier. It succeeds (1.2.2). If, however, the failure occurs after processing has begun in the Access Tier, failover does not occur and an error is reported to the network node.
5. The callback EJB in the Access Tier attempts to send the request to the application (1.2.2.1). If the application is unreachable or does not respond, the request is considered failed, and an error is reported to the network node.

Registering Notifications with Network Nodes

Before applications can receive network-triggered traffic, or notifications, they must register their interest in doing so with Network Gatekeeper, either by sending a request or having the operator set the notification up using OAM methods. In turn these notifications must be registered with the underlying network node that will be supplying them. The form of this registration is dependent on the capabilities of that node.

If registration for notifications is supported by the underlying network node protocol, the traffic path's network plug-in is responsible for performing it, whether the registration is the result of an application-initiated registration request or an on-line provisioning step in Network Gatekeeper. For example, all OSA/Parlay Gateway interfaces support such registration for notifications.

Some network protocols may not support all registration types. For example, in MM7 an application can register to receive notifications for delivery reports on messages is sent *from* the application, but not to receive notifications on messages sent *to* the application from the network. In this case, registration for such notifications can be done as an off-line provisioning step in the MMSC.

Network Gatekeeper is responsible for correlating all network-triggered traffic with its corresponding application, whether the original registration for notification was completed using a request from the application or OAM methods.

There are three categories for such registrations, based on the expected life span of the notification. These categories determine the failover strategies used:

- Short life span

  These notifications are very short-lived, with an expected life span of a few seconds. Typically these are delivery acknowledgements for hand-off of the request to the network node, where the response to the request is reported asynchronously. For this category, a single plug-in, the originating one, is deemed sufficient to handle the response from the network node.

- Medium life span

  These notifications are neither short- nor long-lived, with an expected life span of minutes up to a few days. Typically these are delivery acknowledgements for message delivery to an end-user terminal. For this category, the delivery notification criteria that have been registered are replicated to exactly one additional instance of the network protocol plug-in. The plug-in that receives the notification is responsible for registering a secondary notification with the network node, if possible.

- Long life span

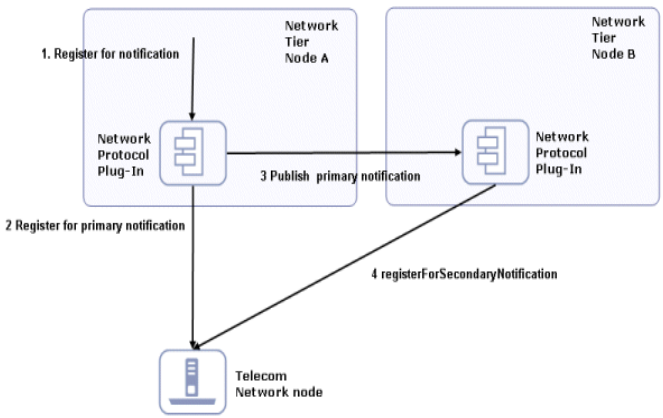
These notifications are long-lived, with an expected life span of more than a a few days. Typically these are registrations for notifications for network-triggered SMS and MMS messages or calls that need to be forwarded to an application. For this category, the delivery notification criteria are replicated to all instances of the network plug-in. Each plug-in that receives the notification is responsible for registering an interface with the network node.

Network Node Supports Primary and Secondary Notification

Figure below illustrates how Network Gatekeeper registers both primary and secondary notifications with network nodes that support it. This capability must be supported both by the network protocol in the abstract, and by the implementation of the protocol as it exists in both the network node and the traffic path's network plug-in.

Note: The scenario assumes that the network node supports registration for notifications with overlapping criteria (primary/secondary).

Figure 9-4 Registration flow with primary/secondary notifications



1. The request to register for notifications enters the network protocol plug-in from the application.

2. The primary notification is registered with the telecom network node.
3. The notification information is propagated to another instance of the network protocol plug-in.
4. The secondary notification is registered with the telecom network node.

Note: The concept of primary/secondary notification is not necessarily ordered. The most recently registered notification may, for example, be designated the primary notification.

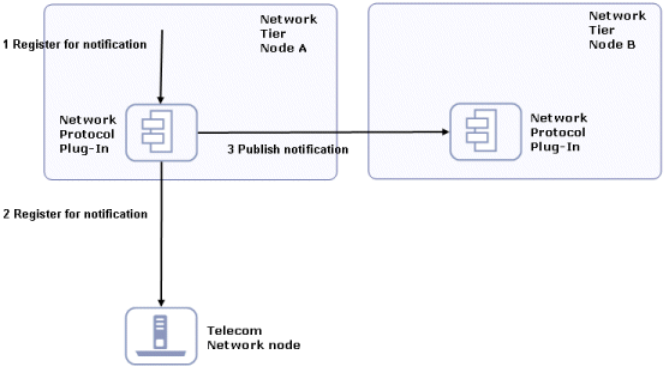
When a network-triggered request that matches the criteria in a previously registered notification reaches the telecom network node, the node first tries the network plug-in that registered the primary notification. If that request fails, the network node has the responsibility of retrying, using the plug-in that registered the secondary notification. The secondary plug-in will have all necessary information to propagate the request through Network Gatekeeper and on to the correct application.

Network Node Supports Only Single Notification

Figure 9-5 below illustrates the registration step in Network Gatekeeper if the underlying network node does not support primary/secondary notification registration.

Note: The scenario assumes that the network node does not support registration for notifications with overlapping criteria. Only one notification for a given criteria is allowed.

Figure 9-5 Registration flow with single notification node

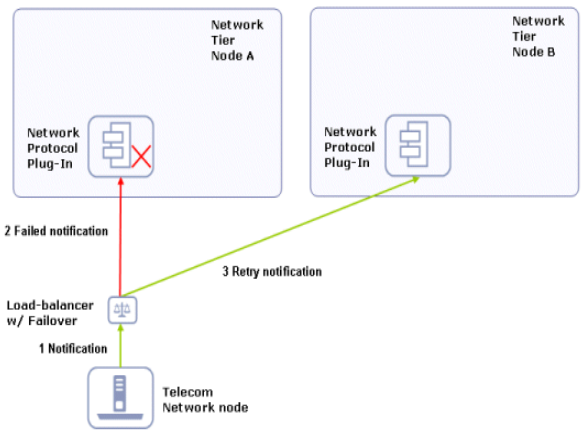


1. The request to register for notifications enters the network protocol plug-in from the application.
2. The primary notification is registered with the telecom network node.
3. The notification information (matching criteria, target URL, etc.) is propagated to another instance of the network protocol plug-in. The plug-in makes the necessary arrangements to be able to receive notifications.

As is clear from the above illustration, in this situation the underlying network node has a callback interface to only a single network plug-in. In order to achieve high-availability and load-balancing a load balancer with fail-over support must be introduced between the network protocol plug-in and the network node, as in Figure 9-6 below.

Note: Whether of not this is possible depends on the network protocol, as the load-balancer must be protocol-aware.

Figure 9-6 Traffic flow with single notification node



Network Configuration

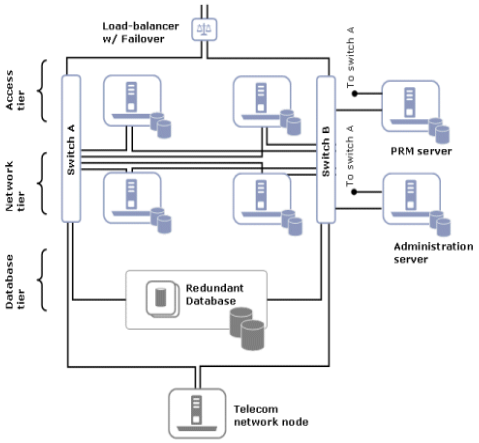
In addition to the specific hardware components listed above, the general structure of a Network Gatekeeper installation is designed to support redundancy and high availability. A typical installation consists of a number of UNIX/Linux servers connected through duplicated switches. Each server has redundant network cards connected to separate switches. The servers are organized into clusters, with the number of servers in the cluster determined by the needed capacity.

As described previously, Network Gatekeeper is divided into an Access Tier, which manages connections to applications and a Network Tier, which manages connections to the underlying telecom network. For security, the Network Tier is usually connected only to Access Tier servers, the appropriate underlying network nodes, and the WebLogic Server administration server, which manages the domain. A third tier hosts the database. This tier should be hosted on dedicated, redundant servers. For physical storage, a Network Attached Storage via fibre channel controller cards is an option.

Because the different tiers perform different tasks, their servers should be optimized with different physical profiles, including amount of RAM, disk-types, and CPUs. Each tier scales individually, so the number of servers in a specific layer tier can be increased without affecting the other tiers.

A sample configuration is shown in Figure 9-7. Smaller systems in which the Access Tier and the Network Tier are co-located in the same physical servers are possible, but only for non-production systems,. Particular hardware configurations depend on the specific deployment requirements, and are worked out in the dimensioning and capacity planning stage.

Figure 9-7 Sample hardware configuration



In high availability mode, all hardware components are duplicated, eliminating single point of failure. This means that there are at least two servers executing the same software modules, that each server has two network cards, and that each server has a fault-tolerant disk system, for example RAID.

The administration server may have duplicate network cards, connected to each switch. The optional PRM servers should run on separate, dedicated servers.

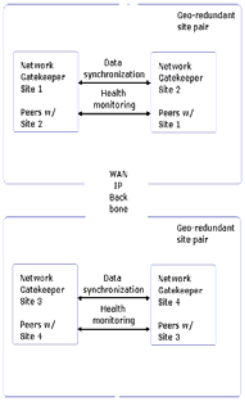
For security reasons, the servers used for the Access Tier can be separated from the Network Tier servers using firewalls. The Access Tier servers reside in a Demilitarized Zone (DMZ) while the Network Tier servers are in a trusted environment.

Geographic Redundancy

All Network Gatekeeper modules in production systems are deployed in clusters to ensure high availability. This prevents single points of failure in general usage. To prevent service failure in the face of catastrophic events - natural disasters or massive system outages like power failures -

Network Gatekeeper can also be deployed at two geographically distant sites as site pairs. Each site, which is a Network Gatekeeper domain, has a site peer. See Figure 9-8 for more information.

Figure 9-8 Overview of geographically redundant site pairs



Note: The geographic distribution of the sites is *not* transparent to the applications accessing Network Gatekeeper. There is no single sign-on mechanism across sites and an application must establish a session with each site it intends to use. In case of site failure, an application must manually fail-over to a different site. Provisioning for each site must be performed individually.

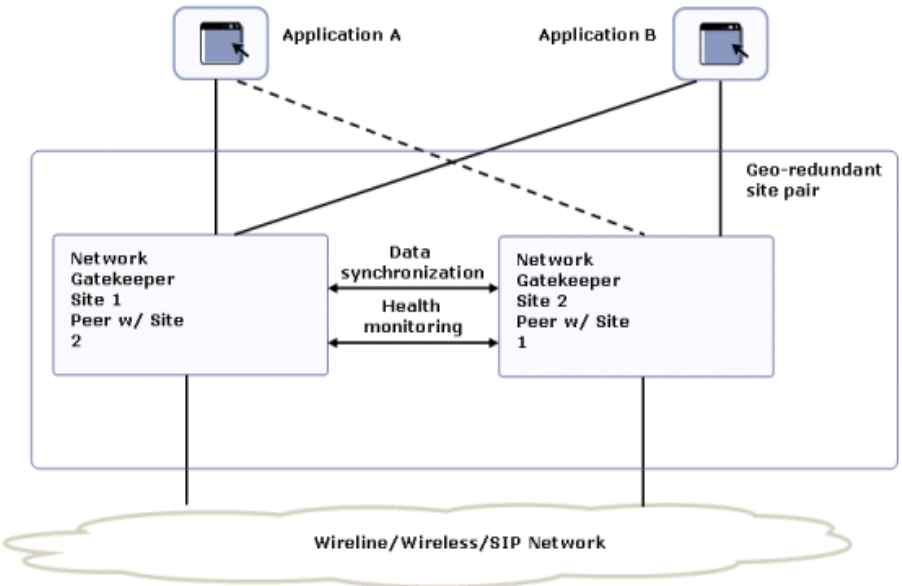
SLA enforcement is synchronized across geographic sites and SLAs are enforced across predefined pairs. Each site is configured to have a reference to its peer site. A subset of all SLAs for a given site is designated as being enforceable across sites. Exactly which parts are selected depends on particular applications and their usage patterns.

Each site maintains a designated hub node that responsible for accounting and the enforcement of SLAs at that site. The service executing on the hub node is highly available and is migrated to another server should server failure occur. Cross-site enforcement is accomplished through hub-to-hub synchronization of global usage counts. The accuracy of enforcement across site pairs

is configurable through an accuracy factor, which is translated into a synchronization interval based on, among other settings, the number of servers.

Applications that normally use only a single site for their traffic can failover to their peer site while maintaining ongoing SLA enforcement. This scenario is particularly relevant for SLA aspects that have longer term impact such as quotas.

Figure 9-9 Geographically redundant site pairs and applications



The geographic redundancy design does not maintain state for *ongoing conversations*. Conversations in this sense are defined in terms of the correlation identifiers that are returned to the applications by Network Gatekeeper or passed into Network Gatekeeper from the applications. Any state associated with a correlation identifier exists on only a single geographic site and is lost in the event of a site-wide disaster. Conversational state includes, but is not limited

to, call state and registration for network triggered notifications. This type of state is considered volatile, or transient, and is not replicated at the site level.

By implication, therefore, conversations must be conducted and complete on their site of origin. If an application wishes to maintain conversational state cross-site - for example, a registration for network-triggered traffic - it must register with each site individually.

On the other hand, this type of affinity does not prevent load balancing between sites for different or new conversations. An example might be sending an SMS message. Because each such a request constitutes a new conversation, sending SMS messages could be balanced between the sites.

Below is a high-level outline of the redundancy functionality:

- The SLA format supports contracts being able to be enforced across geographic site domains. There is an option to configure the system to enforce the SLA across domains. This can be configured per SLA on both on the service provider group and application group level. By default, SLAs are not enforced across sites.
- Connection lost alarms will be raised whenever the peer sites fail to establish connection a certain number of times. The number of retries is configurable.
- The Network Gatekeeper Budget Service is used to enforce SLA request and quota limits. The budget state is distributed to the other geographic sites. Network Gatekeeper automatically determines the optimal synchronization interval.
- These are the categories of data that are candidates for replication to other sites depending on the restrictions on application load balancing and fail-over between sites you choose to support.
  - SLA quota counters

    Quotas that span over longer period of time are persisted in the database to avoid losing state information during server or site failures. The replication is performed at the level of the Network Gatekeeper as a whole as opposed to relying on the databases to do the replication.
  - SLA request limits

    Request limits that span over longer period of time are persisted, in a manner similar to that of quota counters.
- Alarms are generated if a site does not have identical service provider group and application group level SLA configuration between peer sites.
- Alarms are generated if site A treats site B as a peer, but site B does not recognize site A as a peer.

Limitations:

- Provisioning accounts
  Service provider, application group (including SLA) and account data are not replicated across sites. Provisioning must be performed at each individual site.
- Notifications
  Applications are expected to either register for notifications from all the sites or to re-register for notifications upon site failure.
- Application sessions
  If application requests are to be load balanced across sites, the applications must establish sessions with each site separately.
- Fail-back
  If an application fails over to the back-up site, Network Gatekeeper does *not* support fail-back to the original site.
- SLA Overrides
  SLAs may use overrides that, for example, set traffic levels based on time-of-day. Overrides are not enforced across sites, even if Network Gatekeeper is otherwise configured to enforce SLAs across sites. If overrides are present in these SLAs, alarms are emitted.

CHAPTER 10

Service Extensibility

The following sections describes how to extend the WebLogic Network Gatekeeper functionality, including:

Overview

Networks change. Existing functionality is parsed in new ways to support new features. New nodes with new or modified abilities are added. Because of WebLogic Network Gatekeeper's highly modular design, exposing these new features to partners is a straightforward proposition. There are three ways to extend Network Gatekeeper:

- Entirely new traffic paths
- New network plug-ins that can work with existing application facing interfaces
- New and/or extended policy rules and SLA data.

The Extension Toolkit

To help operators and systems integrators, WebLogic Network Gatekeeper ships with the Web Logic Network Gatekeeper Extension Toolkit. The Toolkit comprises the following features:

- An installer

The Toolkit is available as an install option in the Network Gatekeeper installer. It creates a directory hierarchy that parallels that of the Network Gatekeeper hierarchy.

- An Eclipse plug-in
  The developer supplies information to an Eclipse plug-in wizard, which automatically sets up an Extension Project. Included within this project can be a substantial amount of generated code, including:
  - The entire Access Tier, with the Web Service implementation and any callback EJBs that are necessary

    Note: The Extension Toolkit only supports building traffic paths based on Web Services application-facing interfaces.
  - Most of the Service Capability EJB layer of the Network Tier
  - A skeleton of the code required for the Network plug-in layer of the Network Tier
- A complete sample traffic path
- A component library including code for commonly used functionality in network plug-ins.
- A testing tools suite, including:
  - An Ant task for automatically generating Web Service client stubs
  - A component library including functionality for logging in and maintaining a Network Gatekeeper session
  - General testing utilities
  - Support for adding automatic configuration of Network Gatekeeper to tests
  - A sample test case
- Policy and SLA examples to aid the developer in developing new rules and SLA data.

Backwards Compatibility for 2.2 Extensions

The Extension Toolkit can also be used to create much of the code required to port Network Gatekeeper 2.2 style extensions and plug-ins to the new version 3.0 framework.

CHAPTER 11

Backwards Compatibility 2.2 to 3.0

Version 3.0 represents a substantial re-working of the basic architecture of WebLogic Network Gatekeeper based upon the Java Enterprise Edition. Nonetheless, significant work has been done to insure key forms of backwards compatibility with code created by end-users to run in version 2.2, including:

- Web Services-based Application Clients
- External Listeners
- Extension Traffic Paths and Plug-ins

The following provides a high-level description of the mechanisms by means of which older code can run in the context of the new architecture.

Note: Any code based on the 2.2 model is supported for 3.0, but deprecated.

In addition, there is a section that gives an overview of the internal structure of backwards compatible traffic paths, which may be of use to those administrators who are charged with configuring those paths.

Web Services-based Application Clients

Service Provider application clients *may* need to make changes in two aspects of their code:

- Interfaces
- Authentication

Interfaces

The Web Services-based supplied traffic paths in previous versions of Network Gatekeeper used interfaces based on multiple sources - the Parlay X 1.0 standard, the Parlay X 2.1 draft standard, and Extended Web Services, Network Gatekeeper's own in-house interface set. With the release of 3.0, all interfaces for the supplied traffic paths have been upgraded to the Parlay X 2.1 standard, with the exception of WAP Push interface, which continues to use the Extended Web Services set. Service Provider applications that wish to interact with these supplied traffic paths must update to support these standards. Service Provider applications that wish to interact with operator provided extension traffic paths that were created in the 2.2 framework should be able to continue to use the same interfaces.

Authentication

Authentication in Network Gatekeeper 2.2 was based on obtaining a login ticket from the Access Web Service. This mode of authentication continues to be supported in Network Gatekeeper 3.0, but it is deprecated. The preferred mode of authentication is based on WS-Policy, and uses the Session Management Web Service instead of the Access Web Service. For more information on the difference between these two modes, see the *Application Development Guide*, another document in this set.

External Listeners

Network Gatekeeper 2.2 provided a mechanism for developing CORBA based listeners for events, alarms, and charging. This mechanism continues to be supported in 3.0, but is deprecated. The actual data the 2.2 style listeners receive is identical to the data they received in 2.2, which may not be exactly the same data propagated to 3.0 style listeners. Because 3.0 style enhanced traffic paths may generate slightly different data than 2.2 style paths, some fields in the data these paths deliver to 2.2 listeners may be empty. These fields are deprecated going forward. For more information on the new event/alarm/charging model, see Events, Alarms, and Charging .

Extension Traffic Paths and Plug-ins

Some operators may have customized 2.2 style traffic paths and plug-ins that they have created to support specialized features of their networks and/or needs of their service providers. These traffic paths can be upgraded to run within the context of the 3.0 architecture using tools provided in the Network Gatekeeper Extension Toolkit. Very little additional code needs to be written to

make the necessary changes. For more information, see the *Extension Toolkit - Developer's Guide*, a separate document in this set.

Overview of the Internal Structure of Backwards Compatible Traffic Paths

The backwards compatible traffic paths provided with Network Gatekeeper 3.0 retain in a modified form some of the layered architectural features of version 2.2 traffic paths. Each of these layers may require some configuration, depending on the needs of the traffic path, and they use a CORBA-based connection model to connect to each other rather than the Java-based model used elsewhere in v3.0.

APPENDIX A

Standards and Specifications

The following appendix provides a description of the specific standards that WebLogic Network Gatekeeper supports, along with, where possible, links to the actual specifications. A detailed statement of compliance is available upon request. This detailed statement of compliance is provided only under the terms of a non-disclosure agreement.

Application-facing interfaces

Parlay X 2.1

The Network Gatekeeper application-facing interfaces support the following parts of the Parlay X 2.1 specification.

Note: See http://parlay.org/en/specifications/pxws.asp for links to the specifications.

- **Common**, ETSI ES 202 391-1 V1.2.1 (2006-12) Open Service Access (OSA); Parlay X Web Services; Part 1: Common (Parlay X 2).
- **Third Party Call**, ETSI ES 202 391-2 V1.2.1 (2006-12) Open Service Access (OSA); Parlay X Web Services; Part 2: Third Party Call (Parlay X 2).
- **Call Notification**, ETSI ES 202 391-3 V1.2.1 (2006-12) Open Service Access (OSA); Parlay X Web Services; Part 3: Call Notification (Parlay X 2).
- **Short Messaging**, ETSI ES 202 391-4 V1.2.1 (2006-12) Open Service Access (OSA); Parlay X Web Services; Part 4: Short Messaging (Parlay X 2).

- **Multimedia Messaging**, ETSI ES 202 391-5 V1.2.1 (2006-12) Open Service Access (OSA); Parlay X Web Services; Part 5: Multimedia Messaging (Parlay X 2).
- **Payment**, ETSI ES 202 391-6 V1.2.1 (2006-12), Open Service Access (OSA); Parlay X Web Services; Part 6: Payment (Parlay X 2).
- **Terminal Status**, ETSI ES 202 391-8 V1.2.1 (2006-12), Open Service Access (OSA); Parlay X Web Services; Part 8: Terminal Status (Parlay X 2).
- **Terminal Location**, ETSI ES 202 391-9 V1.2.1 (2006-12), Open Service Access (OSA); Parlay X Web Services; Part 9: Terminal Location (Parlay X 2).
- **Call Handling**, ETSI ES 202 391-10 V1.2.1 (2006-12), Open Service Access (OSA); Parlay X Web Services; Part 10: Call Handling (Parlay X 2).
- **Audio Call**, ETSI ES 202 391-11 V1.2.1 (2006-12), Open Service Access (OSA); Parlay X Web Services; Part 11: Audio Call (Parlay X 2).
- **Presence**, ETSI ES 202 391-14 V1.2.1 (2006-12), Open Service Access (OSA); Parlay X Web Services; Part 14: Presence (Parlay X 2).

Extended Web Services

The Extended Web Services are Network Gatekeeper's proprietary application-facing interfaces. These interfaces are implementations of commonly requested functionality, including, in this release, WAP Push. Although the interfaces themselves are not standardized, they often use standardized elements.

Note: Below is a list of such standardized elements. See http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html for links to the specifications.

The payload of a WAP Push message shall adhere to:

- **WAP Service Indication Specification**, as specified in Service Indication Version 31-July-2001 Wireless Application Protocol WAP-167-ServiceInd-20010731-a.
- **WAP Service Loading Specification**, as specified in Service Loading Version 31-Jul-2001 Wireless Application Protocol WAP-168-ServiceLoad-20010731-a.
- **WAP Cache Operation Specification**, as specified in Cache Operation Version 31-Jul-2001 Wireless Application Protocol WAP-175-CacheOp-20010731-a.

Note: The Extended Web Services WAP Push traffic path does not verify the payload. It simply passes it on to the underlying network node.

Network protocol plug-ins

Off-the shelf, Network Gatekeeper supports the network protocols listed in Table 11-1 through the use of network protocol plug-ins. Although each plug-in is a part of a given traffic path, certain protocols can be used by multiple traffic paths for different purposes. In these cases there may be multiple implementations of the same protocol for use in different traffic paths.

Below is a list of supported network protocols organized per traffic path.

Table 11-1 Network plug-ins organized per traffic path.

Traffic Path	Network protocol plug-in	Specification
Parlay X 2.1 Third Party Call	Parlay 3.3 MultiParty Call Control	ETSI ES 201 915-4 V1.4.1 (2003-07), Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control SCF (Parlay 3), part MultiParty Call Control Service. Section MultiParty Call Control Service. http://parlay.org/en/specifications/apis_archives.asp
	SIP	RFC 3261. http://www.ietf.org/rfc/rfc3261.txt
Parlay X 2.1 Call Notification	Parlay 3.3 MultiParty Call Control	ETSI ES 201 915-4 V1.4.1 (2003-07), Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control SCF (Parlay 3), part MultiParty Call Control Service. Section MultiParty Call Control Service. http://parlay.org/en/specifications/apis_archives.asp
	SIP	RFC 3261. http://www.ietf.org/rfc/rfc3261.txt

Table 11-1 Network plug-ins organized per traffic path.

Traffic Path	Network protocol plug-in	Specification
Parlay X 2.1 Short Messaging	Parlay 5.0 Multimedia Messaging	ETSI ES 203 915-15 V1.1.1 (2005-04), Open Service Access (OSA); Application Programming Interface (API); Part 15: Multi-Media Messaging SCF (Parlay 5). http://parlay.org/en/specifications/apis_archives.asp
	SMPP v3.4	Short Message Peer to Peer, Protocol Specification v3.4, Document Version:- 12-Oct-1999 Issue 1.2. http://smsforum.net/
Parlay X 2.1 Multimedia Messaging:	Parlay 5.0 Multimedia Messaging	ETSI ES 203 915-15 V1.1.1 (2005-04), Open Service Access (OSA); Application Programming Interface (API); Part 15: Multi-Media Messaging SCF (Parlay 5). http://parlay.org/en/specifications/apis_archives.asp
	MM7 v 5.3.0 Ericsson MM7 1.0 Ericsson MM7 R2.0 ACA 03 Ericsson MM7 R2.5 ACA 04 **Note:** Only one of the above listed protocols can be used at the same moment for a given node in a domain.	3rd Generation Partnership Project; Technical Specification Group Terminals; Multimedia Messaging Service (MMS); Functional description; Stage 2 (Release 5), 3GPP TS 23.140 V5.3.0. Messages are compliant with XSD schemes defined with name space http://www.3gpp.org/ftp/Specs/html-info/23140.htm

Table 11-1 Network plug-ins organized per traffic path.

Traffic Path	Network protocol plug-in	Specification
Parlay X 2.1 Payment	Parlay 3.3 Charging	ETSI ES 201 915-12 V1.4.1 (2003-07), Open Service Access (OSA); Application Programming Interface (API); Part 12: Charging SCF (Parlay 3). http://parlay.org/en/specifications/apis_archives.asp
Parlay X 2.1 Terminal Status	Parlay 3.3 Mobility, User Status	ETSI ES 201 915-6 V1.4.1 (2003-07), Open Service Access (OSA); Application Programming Interface (API); Part 6: Mobility SCF (Parlay 3). User Status part. http://parlay.org/en/specifications/apis_archives.asp
Parlay X 2.1 Terminal Location	Parlay 3.3 Mobility, User Location	ETSI ES 201 915-6 V1.4.1 (2003-07), Open Service Access (OSA); Application Programming Interface (API); Part 6: Mobility SCF (Parlay 3). User Location part. http://parlay.org/en/specifications/apis_archives.asp
	MLP 3.0 MLP 3.2 **Note:** Only one of the above listed protocols can be used at the same moment for a given node in a domain.	Location Inter-operability Forum (LIF) Mobile Location Protocol, LIF TS 101 Specification Version 3.0.0 and Mobile Location Protocol 3.2 Draft Version 3.2 Open Mobile Alliance, OMA-TS-MLP-V3_2-20050914-D. MLP 3.0: http://www.openmobilealliance.org/tech/affiliates/lif/lifindex.html MLP 3.2: http://www.openmobilealliance.org/release_program/mls_v1_0.html

Table 11-1 Network plug-ins organized per traffic path.

Traffic Path	Network protocol plug-in	Specification
Parlay X 2.1 Call Handling	Parlay 3.3 Call User Interaction and Parlay 3.3 MultiParty Call Control	Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control SCF, ETSI ES 201 915-4 V1.4.1 section MultiParty Call Control Service Open Service Access (OSA); Application Programming Interface (API); Part 5: User Interaction SCF (Parlay 3), ETSI ES 201 915-5 V1.4.1. http://parlay.org/en/specifications/apis_archives.asp
Parlay X 2.1 Audio Call	Parlay 3.3 Call User Interaction and Parlay 3.3 MultiParty Call Control	ETSI ES 201 915-4 V1.4.1 (2003-07), Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control SCF (Parlay 3). Section MultiParty Call Control Service ETSI ES 201 915-5 V1.4.1 (2003-07), Open Service Access (OSA); Application Programming Interface (API); Part 5: User Interaction SCF (Parlay 3). Call user interaction parts. http://parlay.org/en/specifications/apis_archives.asp
Parlay X 2.1 Presence	SIP	RFC 3261. http://www.ietf.org/rfc/rfc3261.txt

Table 11-1 Network plug-ins organized per traffic path.

Traffic Path	Network protocol plug-in	Specification
Extended Web Services WAP Push	PAP 2.0	Push Access Protocol, WAP Forum™, WAP-247-PAP-20010429-a. http://www.openmobilealliance.org
	Parlay 5.0 Multimedia Messaging	ETSI ES 203 915-15 V1.1.1 (2005-04), Open Service Access (OSA); Application Programming Interface (API); Part 15: Multi-Media Messaging SCF (Parlay 5). http://parlay.org/en/specifications/apis_archives.asp
Not applicable	Parlay 3.3 Framework **Note:** This is a network-facing protocol that does not belong to a certain traffic path. It is used by all Parlay plug-ins.	ETSI ES 201 915-3 V1.4.1 (2003-07), Open Service Access (OSA); Application Programming Interface (API); Part 3: Framework (Parlay 3). http://parlay.org/en/specifications/apis_archives.asp

Security

Network Gatekeeper supports the security standards listed below. The security standards are applicable for the application-facing interfaces. Network Gatekeeper leverage Web Services Security mechanisms provided by WebLogic Server. For more information, see *Understanding WebLogic Security* and *Programming Web Services for WebLogic Server*,

Note: See http://www.oasis-open.org/specs/index.php for links to the specifications.

WS-Security Core Specification 1.0, as described in Web Services Security: SOAP Message Security 1.1 (WS-Security 2004) OASIS Standard Specification, 1 February 2006.

UsernameToken Profile 1.0, as specified in Web Services Security UsernameToken Profile 1.0 OASIS Standard 200401, March 2004.

X.509 Certificate Token Profile, as specified in Web Services Security X.509 Certificate Token Profile OASIS Standard 200401, March 2004.

SAML Token Profile 1.1, as specified in Web Services Security: SAML Token Profile 1.1 OASIS Standard, 1 February 2006.

SOAP Message Security 1.0 (WS-Security 2004), as specified in Web Services Security: SOAP Message Security 1.0 (WS-Security 2004) OASIS Standard 200401, March 2004.

Transport-level security mechanisms such as SSL or VPN tunneling can be used for the PRM interfaces.

APPENDIX B

Connecting to OSA/Parlay Gateways

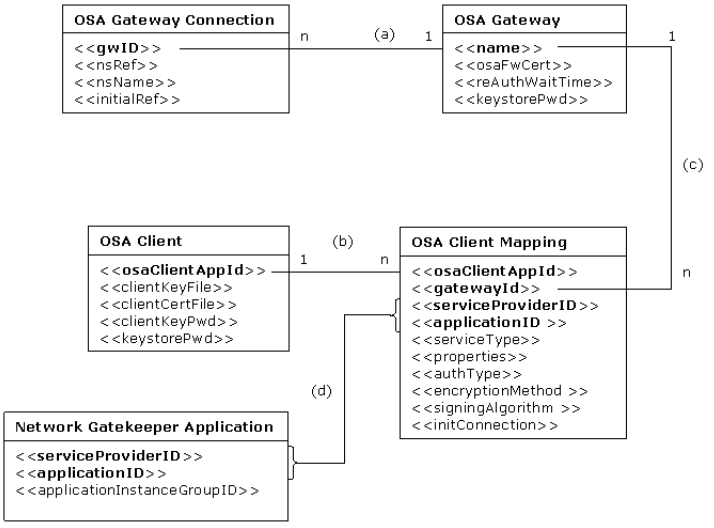
In some cases Network Gatekeeper does not connect directly to an underlying telecom network node. Instead it connects to the network using an OSA/Parlay Gateway. For some traffic paths there may even be multiple OSA/Parlay Gateways that can carry traffic to the network. For example, the Parlay X 2.1 Multimedia Messaging traffic path can be configured, off the shelf, to connect to one or more Parlay 5.0 Multimedia Messaging SCSes and/or one or more Parlay 3.3 User Interaction SCSes. From the point of view of Network Gatekeeper, an OSA/Parlay Gateway is a network node, while the Gateway sees Network Gatekeeper as an OSA/Parlay application.

Because the OSA/Parlay Gateway sees Network Gatekeeper as an OSA/Parlay application, certain parameters defining the connection must be set up in Network Gatekeeper using the Management Console before a connection can be made. This chapter provides a high level overview of this type of connection. For more information on the specific OAM methods used to set up a given connection, see the “Managing and Configuring OSA/Parlay Gateway Connections” chapter in the *System Administrator’s Guide*, a separate document in this set.

Defining Connections

The modules involved in defining a connection between Network Gatekeeper and an OSA/Parlay Gateway are shown in Figure 11-1 below.

Figure 11-1 OSA Gateway Connection Model



(a) One per OSA Framework in OSA Gateway. Keyed by <<name>> and <<gwID>>
(b) One per SCS used by the plug-in. Keyed by <<osaClientAppId>>
(c) <<name>> and <<gatewayID>> are linked
(d) Relation between Network Gatekeeper Application and OSA application

These modules include:

- An **OSA Gateway,** which represents the actual OSA Gateway. Each OSA Gateway (more than one are possible) that is available is registered in Network Gatekeeper
- Each OSA Gateway has one or more **OSA Gateway Connections**. Multiple connections are used if the actual OSA Gateway contains more than one OSA Framework.
- Because the OSA Gateway needs Network Gatekeeper to be an OSA Client, an **OSA Client** module (more than one are possible) represents the user credentials part of a standard OSA Client that are not normally a part of the information that Network Gatekeeper sends to the network.

- An **OSA Client Mapping** (more than one are possible) maps the normal credentials that the application supplies when it logs into Network Gatekeeper together with the OSA Gateway specific credentials stored in the OSA Client module. There must be (at least) one Client Mapping per OSA SCS. If the traffic path uses *n* OSA SCSs, *n* Client Mappings must be defined.

Note: Wildcard mechanisms can be used in the Client Mapping, as described below.

a. The client mapping may be set up per application level, so there is a one to one mapping between a Network Gatekeeper application and the equivalent OSA Client. This means that every transaction originating from a specific application results in a transaction in the OSA Gateway that is traceable to that specific application.

b. The client mapping can use a wildcard for the application level, but specify the service provider, so multiple Network Gatekeeper applications that originate from a common service provider are mapped to a single OSA Client. In this case, the transactions in the OSA/Gateway are traceable only to the service provider.

c. The client mapping can use wildcards for both the service provider and the application level, so all applications from all service providers are mapped to a single OSA Client. In this case, transactions in the OSA/Gateway are traceable only to Network Gatekeeper.

Defining the OSA Client mapping is normally part of the provisioning chain in setting up service provider and application accounts. If the authentication method used between the Network Gatekeeper and an OSA/Parlay Gateway requires certificates and keys, these are set up when establishing user mapping details.

Connection lookup

Network Gatekeeper traffic paths use an internal service, the OSA Access service, to manage all connections with OSA/Parlay Gateways. The traffic path asks the OSA Access service for a connection, and the service handles all of the details of Authentication, Service Discovery, and Load Management towards the OSA/Parlay Framework before returning the handle for the SCS to the traffic path.

Note: A connection can be configured to be initialized either when the OSA Client is first created or when the first request involving the OSA Client is sent to the OSA/Parlay Gateway.

The OSA Access service also uses the Load Management functionality provided by the OSA/Parlay Framework to monitor the SCS. When the handle for the SCS is retrieved, the OSA Access service caches the handle. The OSA Access service is thus responsible for both load balancing and failover towards the OSA Gateway.

Technical Specifications

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

Supported Configurations

The following sections describe the supported configurations for WebLogic Network Gatekeeper.

Overview of Network Gatekeeper Base platform

Below is a summary of the operating systems and hardware platforms for WebLogic Network Gatekeeper:

- HP-UX 11.23 on Intel Itanium2
- Linux Redhat AS4 on Intel Xeon
- Solaris 9 or Solaris 10 on Sun UltraSPARC

The next sections describe the configuration requirements for access tier, network tier and database tier servers. For a description of the different tiers, see Network Configuration

Common configuration requirements

All servers in the cluster building up the Network Gatekeeper must be dedicated servers.

The directory in which the software is installed must reside on the server's local file system.

There must be at least 1 GB of diskspace available under `/user/local`.

HP-UX 11.23 on Intel Itanium2

Configuration requirements for Access tier servers

Table 11-2 Requirements for WebLogic Network Gatekeeper access tier servers on HP-UX 11.23 on Intel Itanium2

Operating System Version and Patches	HP-UX 11.23 with HP-UX patches for Java™ See http://www.hp.com/products1/unix/java/patches/index.html.
Chip Architecture and Minimum Processor Speed	Intel Itanium2 (1.5 GHz)
JDK	HP-UX JDK for the Java 2 Standard Edition platform version 5.0.03 with Java HotSpotTM Server VM (32-bit) and all later JDK 5.0.* service packs for development and production deployment on HP-UX
RAM	1 GB required; 2 GB recommended
Disk	2 x 36 GB
Network cards	2 x LAN interface card

Configuration requirements for Network tier servers

Table 11-3 Requirements for WebLogic Network Gatekeeper network tier servers on HP-UX 11.23 on Intel Itanium2

Operating System Version and Patches	HP-UX 11.23 with HP-UX patches for Java™ See http://www.hp.com/products1/unix/java/patches/index.html.
Chip Architecture and Minimum Processor Speed	Intel Itanium2 (1.5 GHz)

Table 11-3 Requirements for WebLogic Network Gatekeeper network tier servers on HP-UX 11.23 on Intel Itanium2

JDK	HP-UX JDK for the Java 2 Standard Edition platform version 5.0.03 with Java HotSpotTM Server VM (32-bit) and all later JDK 5.0.* service packs for development and production deployment on HP-UX
RAM	1 GB required; 2 GB recommended
Disk	2 x 36 GB
Network cards	2 x LAN interface card

Configuration requirements for Database tier servers

Table 11-4 Requirements for WebLogic Network Gatekeeper database tier servers on HP-UX 11.23 on Intel Itanium2

Operating System Version and Patches	HP-UX 11.23 with HP-UX patches for Java™ See http://www.hp.com/products1/unix/java/patches/index.html.
Chip Architecture and Minimum Processor Speed	Intel Itanium2 (1.5 GHz)
JDK	HP-UX JDK for the Java 2 Standard Edition platform version 5.0.03 with Java HotSpotTM Server VM (32-bit) and all later JDK 5.0.* service packs for development and production deployment on HP-UX
RAM	2 GB required; >6 GB recommended
Disk	2 x 36 GB
Network cards	2 x LAN interface card
RDBMS	See Supported databases
Database storage system	Network attached storage using fibre channel interface.

Linux Redhat AS4 on Intel Xeon

Configuration requirements Access tier servers

Table 11-5 Requirements for WebLogic Network Gatekeeper access tier servers on Linux Redhat AS4 on Intel Xeon

Operating System Version and Patches	Red Hat Enterprise Linux AS release 4 (Nahant Update 2) Kernel version 2.6.9-22.Elsmp glibc-2.3.4-2.13 and later updates and errata levels
Chip Architecture and Minimum Processor Speed	Intel Xeon (3.4 GHz)
JVM	Sun: Version 1.5.0_10 and all later JDK 5.0.* service packs JRockit: Version 1.5.0_11, build R27.3.1
RAM	1 GB required; 2 GB recommended
Disk	2 x 36 GB
Network cards	2 x LAN interface card

Configuration requirements for Network tier servers

Table 11-6 Requirements for WebLogic Network Gatekeeper network tier servers on Linux Redhat AS4 on Intel Xeon

Operating System Version and Patches	Red Hat Enterprise Linux AS release 4 (Nahant Update 2) Kernel version 2.6.9-22.Elsmp glibc-2.3.4-2.13 and later updates and errata levels
Chip Architecture and Minimum Processor Speed	Intel Xeon (3.4 GHz)
JVM	Sun: Version 1.5.0_10 and all later JDK 5.0.* service packs JRockit: Version 1.5.0_11, build R27.3.1
RAM	1 GB required; 2 GB recommended

Table 11-6 Requirements for WebLogic Network Gatekeeper network tier servers on Linux Redhat AS4 on Intel Xeon

Disk	2 x 36 GB
Network cards	2 x LAN interface card

Configuration requirements for Database tier servers

Table 11-7 Requirements for WebLogic Network Gatekeeper database tier servers on Linux Redhat AS4 on Intel Xeon

Operating System Version and Patches	Red Hat Enterprise Linux AS release 4 (Nahant Update 2) Kernel version 2.6.9-22.Elsmp glibc-2.3.4-2.13 and later updates and errata levels
Chip Architecture and Minimum Processor Speed	Intel Xeon (3.4 GHz)
JVM	Sun: Version 1.5.0_10 and all later JDK 5.0.* service packs JRockit: Version 1.5.0_11, build R27.3.1
RAM	2 GB required; >6GB recommended
Disk	2 x 36 GB
Network cards	2 x LAN interface card
RDBMS	See Supported databases
Database storage system	Network attached storage using fibre channel interface.

Solaris 9 or Solaris 10 on Sun UltraSPARC

Configuration requirements Access tier servers

Table 11-8 Requirements for WebLogic Network Gatekeeper access tier servers on Solaris 9/10 on UltraSPARC

Operating System Version and Patches	Solaris 9/10
Chip Architecture and Minimum Processor Speed	UltraSPARC IIIi (1.5 Ghz)
JVM	Sun - Version 1.5.0_10 and all later JDK 5.0.* service packs
RAM	1 GB required; 2 GB recommended
Disk	2 x 35 GB
Network cards	2 x LAN interface card

Configuration requirements for Network tier servers

Table 11-9 Requirements for WebLogic Network Gatekeeper network tier servers on Solaris 9/10 on UltraSPARC

Operating System Version and Patches	Solaris 9/10
Chip Architecture and Minimum Processor Speed	UltraSPARC IIIi (1.5 Ghz)
JVM	Sun - Version 1.5.0_10 and all later JDK 5.0.* service packs
RAM	1 GB required; 2 GB recommended
Disk	2 x 35 GB
Network cards	2 x LAN interface card

Configuration requirements for Database tier servers

Table 11-10 Requirements for WebLogic Network Gatekeeper database tier servers on Solaris 9/10 on UltraSPARC

Operating System Version and Patches	Solaris 9/10
Chip Architecture and Minimum Processor Speed	UltraSPARC IIIi (1.5 Ghz)
JVM	Sun - Version 1.5.0_10 and all later JDK 5.0.* service packs
RAM	2 GB required; >6 GB recommended
Disk	2 x 36 GB
Network cards	2 x LAN interface card
RDBMS	See Supported databases
Database storage system	Network attached storage using fibre channel interface.

Supported databases

Oracle 10g RAC	Full DB Failover and Fault Tolerance
Oracle 10g Single Instance	
MySQL 4.1.22 Single Instance	

Load balancer and tier 3 switches

Per customer preference.

Firewall

Optional. Per customer preferences.

Disc storage

While disc storage can be an ordinary disk system, for performance and high availability reasons, a RAID system should be used.

General characteristics

CORBA version	CORBA 2.5
Java version	JRE 1.5, JDBC 3.0
Database	Oracle, Single Instance or RAC: • Oracle 10g R1 (Oracle 10.1.0.4 and later patch sets of 10.1.x) • Oracle 10g R2 (Oracle 10.2.0.1 and later patch sets of 10.2.x) MySQL 4.1.22 Single Instance only
ORB	Orbacus 4.3
Parlay X	2.1
Rule engine	JRules 6.5.2
SNMP version	v1, v2
SOAP version	1.1, 1.2
SOAP engine	WLS

Programmable Interfaces

Interface	Description
Plug-in interfaces for: • Audio Call • Third Party Call • Call Handling • Call Notification • Short Messaging • Multimedia Messaging • Terminal Location • Terminal Status • Presence • Payment • WAP Push	Makes it possible to add new network plug-ins for extended network/protocol support.
Parlay X 2.1 based interfaces: • Audio Call • Third Party Call • Call Handling • Call Notification • Short Messaging • Multimedia Messaging • Terminal Location • Terminal Status • Presence • Payment	Provides high level telecom Web Services APIs
Extended API based interfaces for: • Access/Session Management • WAP Push	Provides high level telecom Web Service APIs

Interface	Description
API for Callable Policy	Provides access to the Policy Engine via Web Service API
Utility service interfaces for: • Alarm handling • Charging • Event handling • Time • Trace	Facilitates development by providing support functions.