Oracle® Communications Services Gatekeeper

Concepts

Release 6.1

E64619-01

September 2016

Oracle Communications Services Gatekeeper Concepts, Release 6.1

E64619-01

Contents

Preface

This document gives an overview of Oracle Communications Services Gatekeeper.

Audience

This document is intended for anyone who needs an understanding of how Services Gatekeeper works.

This includes:

- System administrators who will install and maintain Services Gatekeeper
- Third-party application developers who integrate telephony-based functionality into their products
- Operator-based system developers who extend the functionality of Services Gatekeeper or integrate it with partner relationship management (PRM) or operations support systems (OSS) tools
- Managers, support engineers, and sales and marketing personnel for network operators and service providers

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For related information, see the following documents:

- *Oracle Communications Services Gatekeeper Getting Started Guide*
- *Oracle Communications Services Gatekeeper System Administrator's Guide*
- *Oracle Communications Services Gatekeeper API Management Guide*
- *Oracle Communications Services Gatekeeper Application Developer's Guide*

- *Oracle Communications Services Gatekeeper Extension Developer's Guide*
- *Oracle Communications Services Gatekeeper Communication Service Reference Guide*
- *Oracle Communications Services Gatekeeper Statement of Compliance*
- *Oracle Communications Services Gatekeeper Licensing Guide*

1 Introducing Oracle Communications Services Gatekeeper

This chapter provides a high-level view of the Oracle Communications Services Gatekeeper product and its most important features.

Understanding Services Gatekeeper

Services Gatekeeper is an API management and governance platform that enables you to provision and monetize web resources quickly and securely using the provided APIs, or the APIs that you or your partners create. Your product can be the APIs, or you can use Services Gatekeeper to create and monetize applications or services based on these APIs. Services Gatekeeper is most useful for productizing the lightweight REST, SOAP, or other XML-based services used in today's world of web services.

Services Gatekeeper also provides a comprehensive collection of communication services (preconfigured APIs) for use with telecom networks to enhance the capability of telecom legacy networks.

You install Services Gatekeeper in one of two ways based on the needs of your implementation. The default or *single-tier* version of Services Gatekeeper comes preconfigured and is designed to install and deploy quickly on a single self-contained system. Single-tier Services Gatekeeper is robust enough to use as a production platform for testing and small to medium-size implementations. You can also cluster single-tier Services Gatekeeper on several systems to take advantage of high availability features. The single-tier Services Gatekeeper includes an integrated Oracle Java DB database to store information that the Services Gatekeeper GUIs and other processes use to store data. You also have the option to use your own MySQL, or Oracle Real Application Cluster (Oracle RAC) database instead of Oracle Java DB.

Single-tier Services Gatekeeper can use all the same features, APIs, applications, communication services, protocols, that multi-tier Services Gatekeeper offers.

This documentation set generally refers to both the single-tier Services Gatekeeper and multi-tier Services Gatekeeper as simply "Services Gatekeeper." It makes the distinction between the two when the difference is important, such as in the installation and configuration documents.

In contrast, the *mult-tier* version of Services Gatekeeper is designed for you to install its various components on different, dedicated hardware systems. You can install multi-tier components on a single system for testing or evaluation, but it is designed to be used on multiple systems. The multi-tier version also comes with the embedded JavaDB database, primarily for testing. Other than JavaDB, the multi-tier Services Gatekeeper expects that you use a database that is installed on its own system.

For information to help you decide which Services Gatekeeper implementation you need, see "When to use Multi-tier Services Gatekeeper".

Multi-tier Services Gatekeeper comes preconfigured to work with the Oracle Enterprise Manager cloud control product. Or you can integrate it with your own cloud management product.

Both Services Gatekeeper configurations are built using a version of Oracle WebLogic Server hardened and extended to support the specialized needs of API management with an emphasis on telecom networks. See “About the Oracle WebLogic Platform” in *Services Gatekeeper System Administrator's Guide* for more information.

Understanding What Services Gatekeeper Provides

Services Gatekeeper offers these Partner Relationship Management (PRM) GUI tools to create and manage your APIs, and the partners that provide them:

- The Services Gatekeeper Partner and API Management Portal graphical user interface (GUI) tool. You use this tool to:
  - Create and manage APIs, including selecting individual methods to expose to your partners. These elements are displayed as menu selections and input fields in Partner Portal. You also have a robust set of tools available to build, configure, manipulate, translate, and add on-the-fly functionality to your APIs and applications.
  - Create, approve, and manage the partners who create applications and view their statistics using the Services Gatekeeper Partner Portal GUI. You can authorize partners to access Partner Portal.
  - Collect partners into groups, and then apply bandwidth limits to those groups. You can change the bandwidth limits on individual APIs as necessary.
  - Create, approve, and manage the NSS operators who create interfaces that you use in APIs and applications. You can also make them available to your other partners for use in their APIs.
  - Analyze statistics about the APIs your implementation is running. The Partner and API Management Portal interfaces with Oracle Business Intelligence Enterprise Edition release 11.1.1.7.0 (OBIEE) to provide statistics that you use to track API usage, response times, and failure rates. You use these tools to monitor application usage and figure out where to make the necessary adjustments in your products.
- The Services Gatekeeper Partner Portal GUI that your partners use to create and modify applications based on the APIs you make available. You must grant your partners access to the Partner Portal. Once they have access, they can quickly and easily create applications for you to use in the Partner and API Management Portal. They do so by subscribing to and using the APIs previously configured in Partner and API Management Portal and displayed in Partner Portal as menu selections and input fields. A statistics feature is also included for partners to monitor and manage their applications.

  All applications created or updated in Partner Portal are instantly displayed in Partner and API Management Portal for approval by the associated network operator.
- The Network Service Supplier Portal GUI that your network service suppliers use to make create network interfaces, and make them available to you and your partners.

- The Oracle Communications WebLogic Administration Console GUI to configure the back end. You use the Administration Console to configure Services Gatekeeper for your hardware and for administration tasks.
- An expandable self-contained back end to support the portals. The back end includes a single-tier deployment that can contain everything required for Services Gatekeeper on a single host system.
- An API Management API that you can use to customize or replace the PRM Portals.
- An interface to Oracle Business Intelligence Enterprise Edition (OBIEE) that you use to capture statistics on your API usage.

In addition to the PRM portals, Services Gatekeeper includes:

- Built-in API management and governance.
- Comprehensive policy, SLA, and subscriber and network resource management.
- Multiple service facades including support for REST, SOAP, SOA, and native interfaces.
- Extensive list of pre-built network plug-ins.
- Multi-channel authorization and privacy management leveraging the OAuth standard.
- An easily customizable and extensible graphical user interface.

When to Use a Communication Service Instead of an API

If you need to create an API with capabilities that the Partner and API Management Portal does not support, you can use one of the Services Gatekeeper *communication services*, or use the Services Gatekeeper Platform Developer's Studio to create a new communication service. You use communication services to process traffic in much the same ways that APIs do, but they are more flexible. The tradeoff is that you manage them programmatically; you cannot manage them using the Partner and API Management Portal.

You would create a communication service instead of an API if:

- If the API requires a capability that the Partner and API Management Portal does not support. for example if it requires a protocol that the Services Gatekeeper Portals do not support.
- If the API requires simultaneous processing of network nodes. For example is you send requests to three network nodes and want to change processing depending on which of the network nodes responds first.

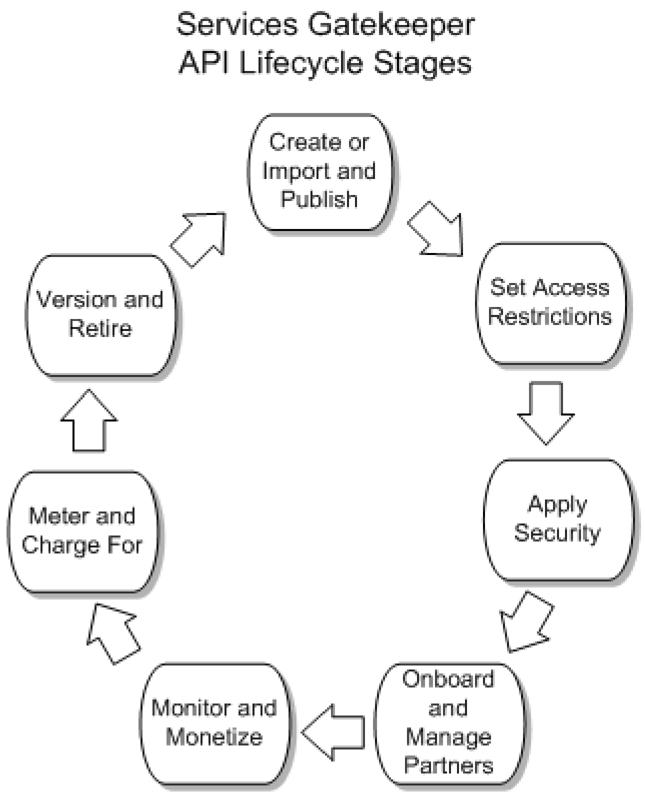
Services Gatekeeper provides a robust collection of existing telecom-based communication services out of the box for you to use if your implementation requires it. See "Using Services Gatekeeper with a Traditional Telecom Network" for a list of the pre-packaged Services Gatekeeper communication services. These communication services wrap the plug-ins in various telecom protocols.

Understanding Services Gatekeeper API Management and Governance

Services Gatekeeper installation provides a cloud-ready API management implementation that uses the Partner and API Management GUIs to create, protect, manage, and monetize your APIs.

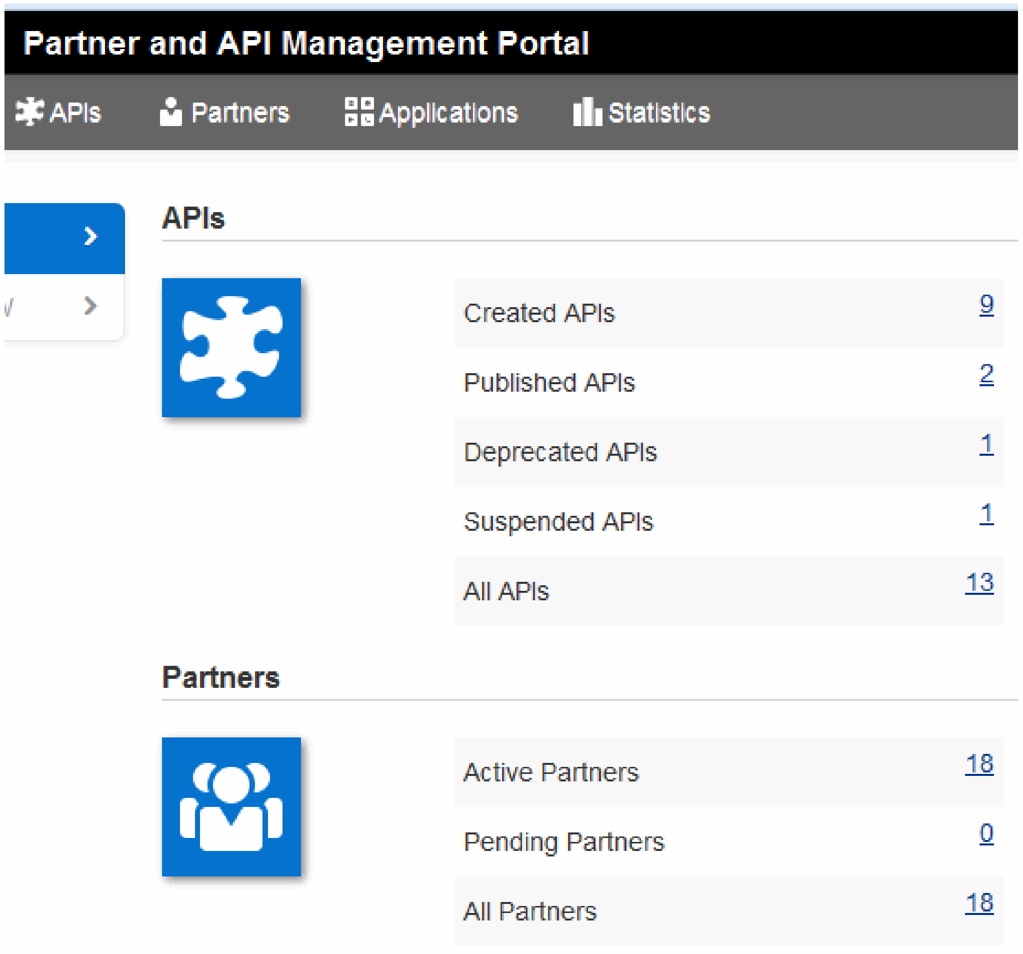
Figure 1–1 shows the Services Gatekeeper API life cycle stages, and illustrates how APIs are created, protected, managed, and monetized. These life cycle stages are explained in sections that follow.

Figure 1–1 API Lifecycle Stages



You use the Services Gatekeeper Partner and API Management Portal shown in Figure 1–2 for most of the API management and governance tasks.

Figure 1–2 Partner and API Management Portal Example



Create or Import and Publish APIs

If you already have created web services, you can use the Partner and API Management Portal wizard to create APIs from those web services and publish the APIs for your partners and customers. You can create an API from an existing web service URI, a REST-based WADL file, a SOAP-based WSDL file, an existing registered network service, or one of the pre-packaged Services Gatekeeper plug-ins. See "Developing Telephony Applications for Services Gatekeeper" in for a list of the plug-ins.

You expose the API using one of the interface combinations listed in Table 1–1.

Table 1–1 Services Gatekeeper API Interface Translation Options

Web Service Interface (Payload Format)	**Network Interface (Payload Format)**
SOAP (XML)	SOAP (XML)
SOAP (XML)	REST (XML/JSON)
REST (XML/JSON)	REST (XML/JSON)
REST (XML/JSON)	SOAP (XML)

Services Gatekeeper also enables you to interface with XML remote procedure calls on both the web service and network interfaces.

Dynamically Apply Actions to API Traffic

Once you have created an API, you use the Partner and API Management Portal to apply SLA and QoS restrictions, and perform a variety of *actions* on the HTTP requests and responses in API traffic. The tasks you can perform on API traffic include:

- Changing the throttling (quota) settings.
- Restricting access to a specific API endpoint.
- Using Cross-reference Origin Resource Support (CORS) to provide secure access to third-party resources.
- Authenticating an application (as an alternative to other authentication mechanisms)
- Converting request/response body data from XML to JSON and JSON to XML.
- Using an XLST on body of the request/response.
- Validating your API Schema.
- Adding REST-based callouts.
- Performing any othermessage manipulation tasks that your implementation requires, using a Groovy programming language script.

You can create and apply actions to an API when you are creating it, and then update them dynamically afterward, as long as the action is still in a PUBLISHED or DEPRICATED state. These dynamic changes take effect immediately; no other action is required.

These actions replace the Services Gatekeeper service intercepter features for HTTP to HTTP communication. The Services Gatekeeper communication services that use non-HTTP protocols must use the services interceptor features for these capabilities.

Apply Security to the API

You use the Network Security options on the Partner and API Management GUI to apply network security to an API. You can configure the security strategy for the APIs to be one of the following:

- None
- Text-based
- OAuth-based

During API creation, the partner manager has the option to make the API *public* (available to all partners) or *private* and only accessible to specific chosen partners.

Onboard and Manage Partners and Partner Groups

You use the Partner and API Management GUI to create partners and collect them into groups. You group partners together that require the same access to a set of APIs. However, you can specify different throttling parameters for each API, and for each partner in the group. A partner could be within your organization, a separate organization or entity, or even a single person.

Meter and Charge for Your APIs and Applications

The most common ways that Services Gatekeeper customers charge for the services they provide are *content-based* charging and *call data record* (CDR) charging. Content-based charging allows external billing systems to probe your Services Gatekeeper database and bill customers based on information that matches certain parameters. Charging by CDR is appropriate for online services for which you charge for services as they are used. For example, charging for access to services, or by duration of service.

Version and Retire APIs

You can revise your APIs as needed and make newer versions available to your customers, partners, and network service suppliers (NSSs). There can only be one active API at time with the same name, but you have the option to deprecate or suspend an API and replace it.

Understanding the Multi-tier Services Gatekeeper Configuration

The multi-tier version of Services Gatekeeper includes all of the features of a single-tier Services Gatekeeper, but can be configured into a more robust and flexible environment.

When to use Multi-tier Services Gatekeeper

As previously mentioned, the single-tier Services Gatekeeper installation is appropriate for small to medium size API management implementations. As your implementation grows, and you find that you need increase performance by separating the various Services Gatekeeper components on dedicates systems, you can convert your system to a multi-tier version. A multi-tier Services Gatekeeper implementation is more flexible, but more complex. The multi-tier implementation uses the same GUI portals and administration console GUI tools as the default implementation, but spreads the back-end work across multiple physical servers. The difference is how the Services Gatekeeper back end is configured.

The multi-tier installation is appropriate for implementations that:

- Process enough traffic through the Application Tier (AT) and Network Tier (NT) that they require separate physical servers.
- Require the Application Test Environment tool to test multi-tier Services Gatekeeper telecom traffic with your applications.
- Require the Platform Test Environment to test your Services Gatekeeper implementation is performing as you expect it to.
- Require large amounts of data that are more efficiently accessed using a robust database (such as Oracle RAC) on a dedicated hardware system. Although the Oracle JavaDB database is supported for testing or small implementations.

  The single-tier Services Gatekeeper usually uses the embedded and easy-to-use Oracle Java DB database, which is appropriate for test small production implementations.

Access to Oracle Communications Converged Application Server for Connectivity to SIP Network Infrastructure

In addition to providing access to traditional telecom network functionality, Services Gatekeeper can also connect application services to SIP-based functionality, using

Oracle Communications Converged Application Server product (Converged Application Server). Calls set up using the Parlay X 2.1 or RESTful Third Party Call communication services can be routed through SIP. Parlay X 2.1 or RESTful Call Notifications can be established using SIP and Parlay X 2.1 or RESTful Presence *watchers* (consumers of presence information) and *presentities* (providers of presence information) can be set up.

Simplifying and Automating Partner Management Using Web Services

Managing numerous services, particularly when the providers are third-party partners, can be time and effort consuming. As the market expands, you use the Partner and API Management Portal interfaces to handle processes such as partner registration, service activation, and provisioning. These web service interfaces support automating a wide range of partner-related tasks and provide partners with easily available access to information about their accounts. The interface also enables operators to create groups of partners sharing sets of data, which can be used for tiering or segmentation of partners. Operators can then focus their administrative and partner management resources on their most rewarding partners.

Secure Access Control for Both Internal and Third-Party Applications

Services Gatekeeper can function as a single point of contact for access to the functionality of the underlying network, providing common authentication, authorization, and access control procedures for all applications, both internal and third-party based. For SOAP-based interfaces, Services Gatekeeper leverages the flexible security framework of Oracle WebLogic Server to provide robust system protection. Applications can be authenticated using plaintext or digest passwords, x.509 certificates, or SAML 1.0/1.1 tokens. Service requests can use XML encryption, based on the W3C standard, for either the whole request message or specific parts of it. And, to ensure message integrity, requests can be digitally signed, using the W3C XML digital signature standard. For RESTful interfaces, Services Gatekeeper uses HTTP basic, OAuth 2.0, or security assertion markup language (SAML) authentication of user name/password and SSL.

Flexible Authorization Control Based on Fine-Grained Policy Decisions

The Services Gatekeeper includes a powerful and responsive policy enforcement mechanism uses service level agreements (SLAs) to regulate service provider and application access to particular communication service functionality down to the level of supported operations and parameters. It also supports a range of quality-of-service (QoS) guarantees that you can modulate by Time of Day/Day of Week, Rates, and Quotas. You can add more rules to limit access. SLA management and maintenance can be simplified by organizing service provider and application accounts into groups. You can also create custom SLA versions to enhance the set of broadly comprehensive SLAs provided by Services Gatekeeper.

You control subscriber permissions and preferences in a separate Subscriber SLA, created by the operator or an integrator using tools available in the Platform Development Studio. Subscribers can indicate, for example, that they want to allow service provider X to query for the location of their mobile terminals, but not service provider Y.

Enhanced Network Protection

In addition to the service level agreements that cover access to functionality within Services Gatekeeper itself, other SLAs explicitly define service provider access to underlying network nodes. Services Gatekeeper throttles and shapes traffic to protect the underlying network during periods of heavy load, using node SLAs.

Built-in Network Routing

Services Gatekeeper provides a fine-grained internal system for routing service requests directly to appropriate network nodes based on various parameters, including the sending application, the destination address, or any other request parameter. Services Gatekeeper supports in-production deployment of multiple instances of most network protocol plug-ins (the module that interacts most directly with the underlying nodes) on an as needed basis.

Carrier Grade and Fully Scalable Architecture

The Services Gatekeeper architecture is based on the performance and clustering features provided by Oracle WebLogic Server, including:

- Tiering

  Services Gatekeeper is deployed in two tiers, an application tier (AT), and a network tier (NT). You can separate these tiers by a firewall for increased security. State is held only in the network-facing tier, and each tier can be built out independently of the other.

- High availability and failover

  Services Gatekeeper is designed throughout to ensure multi-level protection against single points of failure.

- Geographic-redundancy

  To protect the system in the face of catastrophic failure, geographically distant sites can be set up as site pairs. **Service Provider** and **Application Group** SLA enforcement is synchronized across geographic sites and SLAs are enforced between the site pairs. Any changes in account configuration information are also replicated across sites.

- Storage Service

  All traffic that passes through Services Gatekeeper is transactionally wrapped. Maintaining state consistently and durably in clustered and high performance environments is traditionally difficult, but the Services Gatekeeper Storage Service uses a sophisticated strategy of optimizing storage based on state access patterns. An in-memory store distributed among all the nodes serves as the entrance to data access. Reading from disk, and its attendant overhead, is reduced because the disk-based database functions as an archive rather than as a system of first use. The reduced overhead has two important benefits:

  – Speed: Because the data is available in memory, data access is extremely fast.

  – Scalability: As a system scales out, relying exclusively on disk-based database access often becomes a performance bottleneck. Because the data in Services Gatekeeper is distributed among the network tier nodes, adding more servers to the network tier actually increases data availability.

  In addition, the Storage Service optimizes access to exactly the kinds of data that matter most in telecom traffic processing. Designed as a POJO java.util.Map -based API, client access is simplified for both storing data and making retrieval queries.

Coherence is used as the storage provider for configuration, core services, and a set of communication services.

OSS and Billing System Integration

All or selected parts of the Services Gatekeeper management mechanism can be integrated with your external Operation Support Systems (OSS) through JMX/JMS or SNMP interfaces. The tasks associated with administering current service providers and adding new ones can be folded into existing systems.

The Services Gatekeeper internal charging mechanisms can also be integrated with your existing billing systems. Offline and online (using the Parlay X 3.0 Payment API) Diameter-based charging is supported.

Subscriber Personalization and Protection

Using Services Gatekeeper, operators can customize their application functionality for individual subscriber by accessing subscriber profile information stored on network LDAP servers. At the same time, operators can protect subscriber privacy by using filters based on those same profiles to regulate the access that applications have, limiting the information that applications can acquire to what the subscriber wants to make available.

In addition, operators can optionally create Subscriber SLAs, which create service provider groupings called *service classes* that can be associated with individual subscriber URIs. Operators or integrators can do this using the Profile Provider service provider interface (SPI) included in the Platform Development Studio. Using Subscriber SLAs allows subscribers to customize their interactions with application service providers and at the same time keeping all their subscriber data within the confines of your domain.

Extensible Architecture

A flexible architecture using the robust capabilities of Oracle WebLogic Server means that operators can extend existing communication services to support new network interfaces, for example, Unstructured Supplementary Service Data. They can also create entirely new communication services to allow application service developers access to the unique feature of their networks using the Services Gatekeeper Platform Development Studio.

About Developing Applications

Services Gatekeeper includes various documentation and tools to assist application developers:

- *Services Gatekeeper Application Developer's Guide*
- *Services Gatekeeper Java API Reference*
- *Services Gatekeeper OAM Java API Reference*
- *Services Gatekeeper Actions Java API Reference*
- The *Application Test Environment* graphical user interface (in the Services Gatekeeper SDK) that you use to test applications on a simulation of Services Gatekeeper.
- The *Platform Test Environment* graphical user interface that you use to test your application against network and application simulators.

- Web services WSDL and WADL files

Developing New Applications

The Services Gatekeeper documentation set includes the *Services Gatekeeper Application Developer's Guide* which describes the SOAP, REST, Native, and OneAPI interfaces and includes additional information that application developers need. Because the SOAP and Restful interfaces are Web services based, applications can be developed using any environment that the developer chooses.

Using the Application Test Environment GUI to Test Applications

Application developers can use the Services Gatekeeper Software Developer Kit (SDK) to test their applications. The SDK consists of the Application Test Environment (ATE), GUI which is a lightweight tool that simulates an instance of Services Gatekeeper called the Virtual Communication Service (VCS). The VCS exposes both SOAP and RESTful interfaces.

You can perform functional tests on tasks involving communication with Services Gatekeeper, such as opening sessions, sending and receiving messages, and examining delivery reports through the simulator. You can perform these tests without having to connect to an actual Services Gatekeeper implementation. When the time comes to test and later deploy the application with an actual Services Gatekeeper installation, you only need to change a few URLs in the application.

Application developers configure their applications to interface with the ATE. They then monitor the application's activity in the ATE interface, by checking whether messages have been received, and by examining information returned to the application. The ATE GUI includes a map in which developers can delineate geographic regions, insert and move terminals, and send messages to and receive messages from the terminals as if they were mobile devices in an actual network.

For more information, see "Testing Applications with the Application Test Environment" in *Services Gatekeeper Application Developer's Guide*.

Using the Platform Test Environment GUI to Test Features

Services Gatekeeper includes the Platform Test Environment (PTE), a graphical user interface (GUI) that you use to test default Services Gatekeeper features and your own custom communication services. The PTE simulates both the application-facing interfaces and network-facing protocols so you can test Services Gatekeeper behavior.

You use the PTE to configure an actual running Services Gatekeeper implementation using MBeans and specify access and budgets with service-level agreements (SLAs) just as you would in a production environment. The PTE includes clients, simulators, tools, and plug-ins to mimic most of the default Services Gatekeeper features. The PTE is a complete telecom testing environment. You create actual applications within the PTE to use as application-facing interfaces. Each client module interacts with a simulated network protocol on the network-facing part of the PTE. The client modules interact with simulated network protocols just as they would in a production environment. You can also extend the PTE to test your own custom features.

For more information see "Understanding the Platform Test Environment" in *Services Gatekeeper Platform Test Environment User's Guide* for details.

Using Services Gatekeeper with a Traditional Telecom Network

The Services Gatekeeper API management features are directed mainly at HTTP-to-HTTP communication. Services Gatekeeper has also traditionally provided a variety of preconfigured capabilities to use with network that communicate using traditional dedicated telephony protocols.

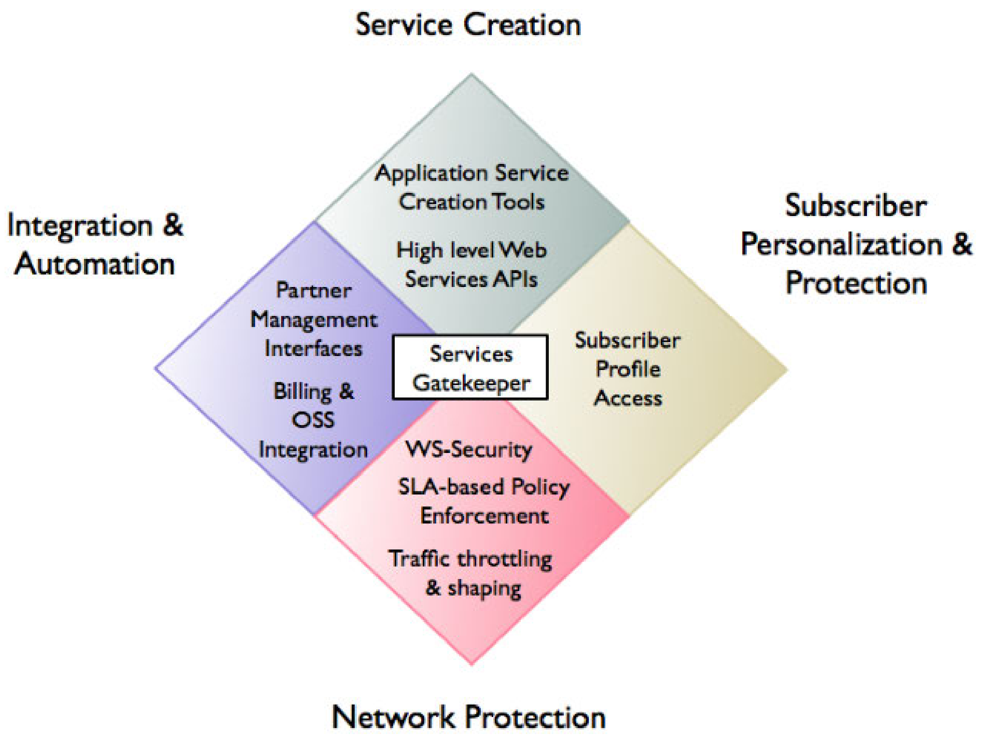
Subscribers continue to require services that provide them with functionality and flexibility that cross the traditional boundaries between the world of the Internet and the world of their phones. Operators want to be responsive to the desires of their subscribers, and to provide services that satisfy subscriber demands, promote subscriber loyalty, increase average revenue per user (ARPU), and drive traffic to their networks.

The telephony-based Services Gatekeeper features allow operators to:

- Offer simplified access to their network's capabilities, for both internal developers and external partners
- Provide tooling and support for application service development and testing
- Manage external partners efficiently
- Protect the security and stability of the underlying network
- Integrate new services with their existing operations and management facilities
- Protect subscriber privacy and control
- Support flexibility of access as networks change and grow
- Do all this in a way that scales and meets the performance needs subscribers have come to expect

With the help of Services Gatekeeper, operations can effectively reduce the overhead of creating the applications that provide required services and enable a wider ranging development community to contribute to a better subscriber experience.

Figure 1–3 Oracle Communications Services Gatekeeper in Context


Service Creation
Integration & Automation
Subscriber Personalization & Protection
Network Protection
Application Service Creation Tools
High level Web Services APIs
Partner Management Interfaces
Billing & OSS Integration
Services Gatekeeper
Subscriber Profile Access
WS-Security
SLA-based Policy Enforcement
Traffic throttling & shaping

Access to Telecom Network Service Capabilities Using APIs Based on Well-Known Standards

The protocols required by underlying telecom network capabilities are often complex, and the learning curve associated with using them is steep. To help application service developers, Services Gatekeeper includes standard network capabilities such as Short Message Service (SMS), Multimedia Messaging Service (MMS) or Call Control through a set of easy-to-use interfaces (called *facades* in Services Gatekeeper).

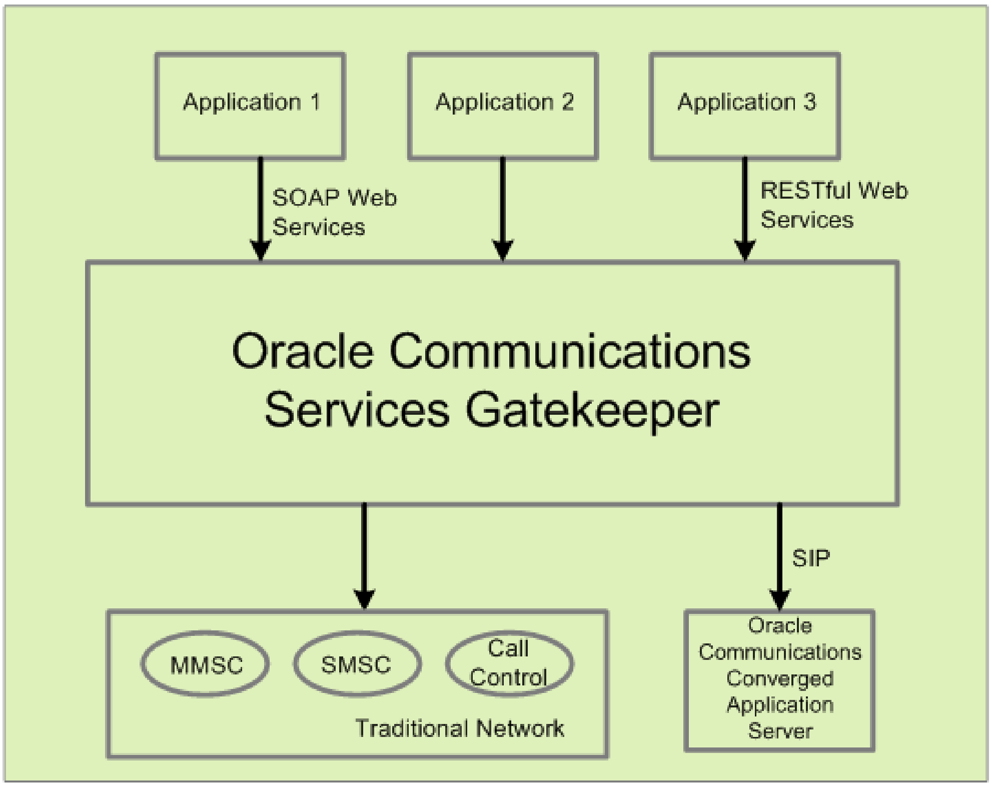
- Services Gatekeeper offers SOAP-style facades based on well-known standards such as Parlay X and some native protocol interfaces.
- Various RESTful web service interfaces designed for ease of use in pure HTTP environments.
- The Oracle Service Bus environment also contains SOAP-style interfaces that pre-integrated, offering application developers SOAP-based functionality and the flexibility of SOA.
- The Oracle extended web services supports protocols which have not yet been incorporated into standardized forms (WAP Push, Binary SMS, and Subscriber Profile). These extended web services interfaces are published as standard Web Services Definition Language (WSDL) files, so application service developers can use their choice of toolsets.
- E-mail communication service which uses MMS in the northbound interface and a plug-in that enables the sending of e-mail through SMTP and receiving e-mail through POP3 and IMAP protocols. You use the RESTful Email Communication interface to send e-mail messages and to fetch information about e-mail messages that have been received for the applications and stored on Services Gatekeeper.

- OneAPI interfaces for MMS, SMS, and various other services.

Developers can focus on creating compelling and innovative services, leaving the communication services components of Services Gatekeeper to handle the mechanics of interacting with the various underlying network elements.

Services Gatekeeper also provides an xparam mechanism to manipulate network protocol parameters which do not exist in the web service API. The xparams are listed in the individual Communication Service descriptions in the *Services Gatekeeper Communication Service Reference Guide*.

Figure 1–4 Standardized Application Interfaces



2

Administering Services Gatekeeper

This chapter explains the operation, administration, and maintenance (OAM) functionality that Oracle Communications Services Gatekeeper provides.

About Administering Services Gatekeeper

You use the Partner Relationship Management (PRM) portals to create and manage APIs, and your partners and network service suppliers (NSSs) that may supply them. See *Services Gatekeeper API Management Guide* for details on these portals and their functions.

You perform most system administration tasks using the Services Gatekeeper Administration Console. For example Services Gatekeeper prompts you for an administrative user that controls the PRM portals during the installation. For example, you use the Administration Console to:

- Create and manage additional administrative users
- Configure the MBeans attributes that control Services Gatekeeper communication services
- Deploy additional software plugins for use in the PRM portals or communication services.

The Administration Console is a specialized extension of the Oracle WebLogic Server Administration Console. It is a Web-based application that you can run in any environment that runs the supported Web browsers. For general information about the Administration Console, see the overview of the Administration Console in *Oracle Fusion Middleware Introduction to Oracle WebLogic Server* at:

http://docs.oracle.com/cd/E15523_01/web.1111/e13752/toc.htm#INTRO107

For some management tasks, you can also use scripts that you run in the WebLogic Scripting Tool. For details, see *Oracle Fusion Middleware Oracle WebLogic Scripting Tool* at:

http://docs.oracle.com/cd/E24329_01/web.1211/e24491/toc.htm

You can integrate all or selected parts of the management application with an external operations support system (OSS) by using JMX/JMS. Additionally, you can configure Services Gatekeeper to distribute alarms through SNMP traps.

You can also integrate application service provider management functionality with external systems using the Services Gatekeeper partner relationship management interfaces.

You can restrict administration access by dividing administrative users into user groups that each have access to different aspects of the administrative functionality.

Within user groups, individual users can have different levels of access. See *Services Gatekeeper System Administrator's Guide* for more information.

About the Administration Console

The Services Gatekeeper Administration Console is a Web browser-based, graphical user interface that you use to manage a WebLogic Server domain.

A standard production installation for Services 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 hosts the Administration Console and provides a central point for managing the Services Gatekeeper domain. All other server instances in the domain are called managed servers. In Services Gatekeeper, managed servers are divided into Access Tiers and Network Tiers. In a domain that contains only a single WebLogic Server instance, such as in a development environment, that server functions as both the administration server and managed servers, and performs the functions of both the Access Tier and the Network Tier.

Overview of Administration Tasks

Use the Administration Console to:

- Configure, start, and stop Services Gatekeeper instances
- Configure Services Gatekeeper clusters
- Configure Services Gatekeeper services, such as database connectivity (JDBC) and messaging (JMS)
- Monitor server and application performance
- View server and domain log files
- View application deployment descriptors
- Edit selected run-time elements of the application deployment descriptors
- Upgrade communication services
- Configure security parameters and roles

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

- Configure Services Gatekeeper communication services
- Manage administrative users and groups
- Provision application service providers, applications, application instances, and related SLAs.
- Monitor alarms, CDRs, and EDRs
- Create multiple plug-in instances, set up plug-in routing, and so on

Tasks you perform outside the Administration Console:

- Extend Services Gatekeeper functionality
- Back up and restore the system
- Upgrade the system

See *Services Gatekeeper Communication Service Reference Guide* for information about configuring and managing communication services.

See *Services Gatekeeper System Administrator's Guide* for information about configuring and managing other aspects of Services Gatekeeper.

About Integrating an Operations Support System

You can integrate all or selected parts of the Service Gatekeeper management with an external OSS through secured JMX/JMS interfaces. For more information about working with JMX, see "Designing Manageable Applications" in *Oracle Fusion Middleware Developing Manageable Applications with JMX for Oracle WebLogic Server*, and "*Fusion Middleware Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*".

You can set up alarm supervision systems that use external JMS listeners to receive user definable types of event-based data, including standard alarms. Services Gatekeeper sends SNMP traps to any registered SNMP managers.

3 Redundancy, Notifications, Load Balancing, and High Availability

This chapter explains the redundancy, load balancing, and high availability software and hardware components that Oracle Communications Services Gatekeeper provides.

About High Availability

The Services Gatekeeper high-availability feature uses the clustering mechanisms made available by Oracle WebLogic Server. For general information about Oracle WebLogic Server and clustering, see *Fusion Middleware Using Clusters for Oracle WebLogic Server* at:

`http://docs.oracle.com/cd/E15523_01/web.1111/e13709/toc.htm`

About Clustering Services Gatekeeper

The default (single-tier) Services Gatekeeper provides a simple, quick method for adding a second clustered system to your implementation to take advantage of the high availability features. Because it is single tier, the installation is much simpler than the multi-tier equivalent. See *Services Gatekeeper Getting Started Guide* for details on how to set it up.

About Clustering Multi-tier Services Gatekeeper

For high availability and security reasons, a multi-tier Services Gatekeeper domain is deployed in two tiers: the Access Tier (AT) and the Network Tier (NT).

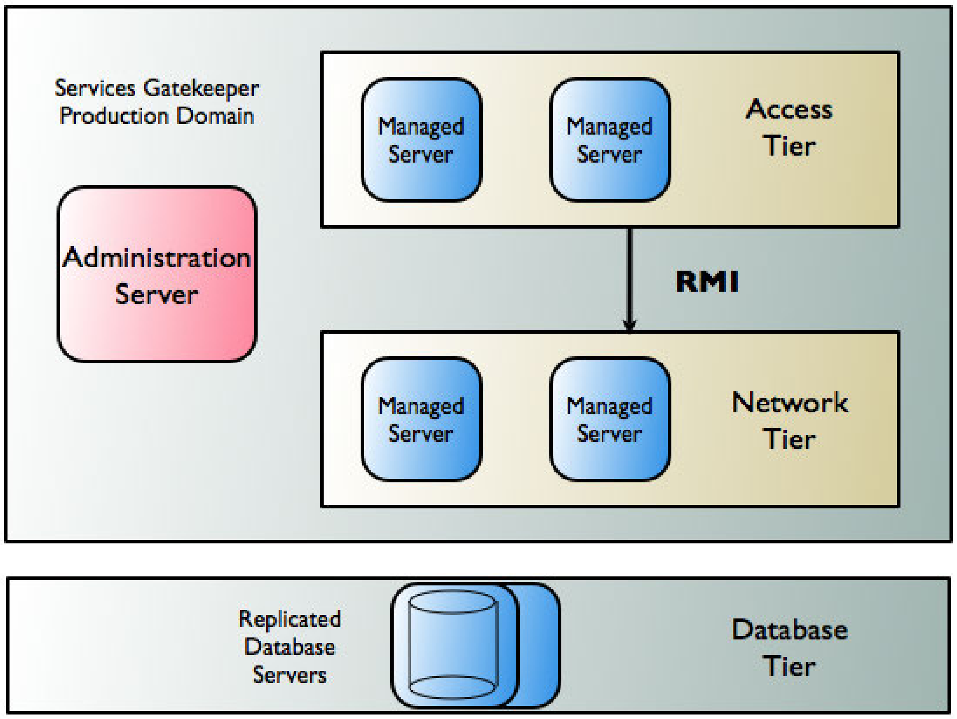
The Native SMPP and Native UCP communication services operate entirely in the Network Tier. When two tiers are used, a Server Services provides access to applications in the Network Tier.

Each tier consists of at least one cluster, with at least two server instances per cluster, and all server instances run in active mode, independently of each other. In the context of WebLogic Server, the servers in all clusters are managed servers. Together the clusters comprise a single WebLogic Server administrative domain, controlled through an Administration Server.

There is an additional tier containing the database. Within the cluster, data is made highly available using a cluster-aware storage service, which distributes all state data across all Network Tier instances.

Figure 3–1 shows an example of the deployment of servers in the three tiers.

Figure 3–1 Sample Multi-Tier Production Deployment



The Access Tier and the Network Tier communicate by using Java remote method invocation (RMI). Application requests are load balanced between the Access Tier and the Network Tier and failover mechanisms are present between the two tiers. See "Managing Traffic Within a Multi-tier Services Gatekeeper" for more information about load balancing and failover in application-initiated and network-triggered traffic flows.

Managing Traffic Within a Multi-tier Services Gatekeeper

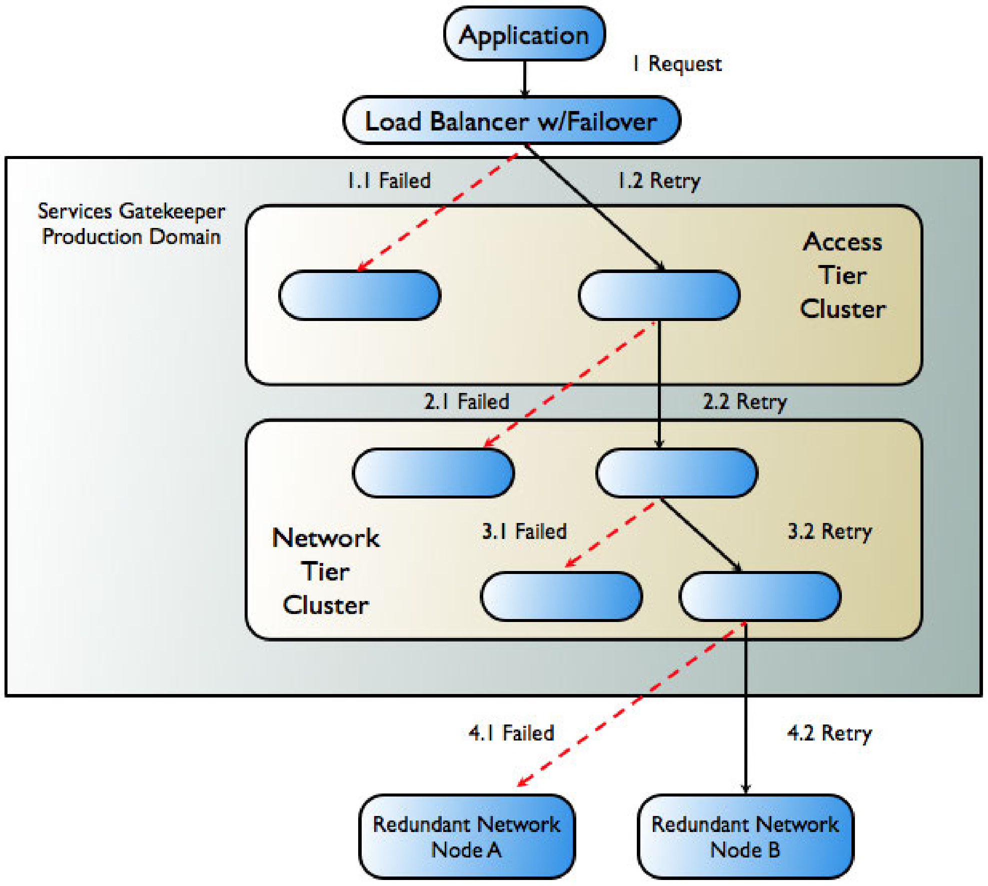
This section describes how Services Gatekeeper handles failover in Access Tiers and Network Tiers for both application-initiated and network-triggered traffic.

Failover for Application-Initiated Traffic

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

Figure 3–2 shows the worst-case scenario for an application-initiated request as it passes through Services Gatekeeper, and the failover mechanisms that attempt to keep the request active.

Figure 3–2 Failover in Application-Initiated Traffic



The following steps describe the workflow as illustrated in Figure 3–2.

1. The application sends a request to Services Gatekeeper. In a production environment, this request is routed through a hardware load balancer that is usually protocol-aware. If the request towards the initial Access Tier server fails (1.1 in Figure 3–2), either a timeout 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 in Figure 3–2) and it succeeds. That server 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 or unmarshalling the request as it travels to the Network Tier server (2.1 in Figure 3–2).
4. A failover mechanism in the Access Tier successfully sends the request to a different server in the Network Tier cluster (2.2 in Figure 3–2). That server then attempts to send the request on to the network node.
5. The request is sent to a plug-in in the Network Tier that is unavailable (3.1 in Figure 3–2). An interceptor from the interceptor stack retries the remaining eligible plug-ins in the same server and successfully sends the request to an available plug-in (3.2 in Figure 3–2).
6. The attempt to send the request to the telecom network node fails (4.1 in Figure 3–2).

7. The request is forwarded to a redundant network node (4.2 in Figure 3–2). If this request fails, the failure is reported to the application.

> **Note:** In addition to the mechanisms described above, Services Gatekeeper also enables you to create multiple instances of a single SMPP plug-in type, with multiple binds, which can set up redundant connections to one or more network nodes. This mechanism can also increase throughput and help optimize traffic to SMSCs that have small transport windows.

Failover for Network-triggered Traffic

Network-triggered traffic can consist of the following:

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

Services Gatekeeper handles failover for network-triggered traffic by using internal mechanisms in combination with the capabilities of the telecom network node or external components such as load-balancers with failover capabilities.

Some network nodes can handle the registration of multiple callback interfaces. In such cases, Services Gatekeeper registers one primary and one secondary callback interface. If the node cannot send a request to the network plug-in registered as the primary callback interface, the node is responsible for retrying the request by sending it to the plug-in that is registered as the secondary callback interface. This plug-in resides in another Network Tier instance. The plug-ins communicate with each other and ensure both callback interfaces are registered. See "Primary and Secondary Notification" for more information.

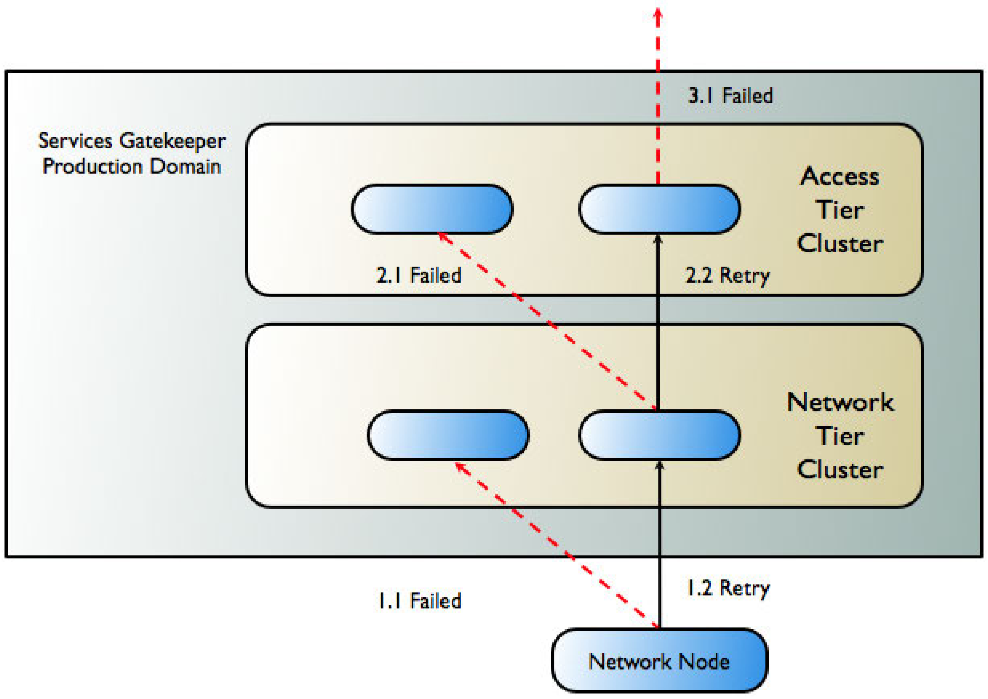
For communication services using SMPP, all Services Gatekeeper plug-ins can function equally as receivers of any transmission from the network node.

For HTTP-based protocols, such as MM7, MLP, and PAP, Services Gatekeeper relies on an HTTP load balancer with failover functionality between the telecom network node and Services Gatekeeper. See "Single Notification" 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 a single connection to one telecom network node.

Figure 3–3 shows the worst-case scenario for a network-triggered request for medium life span notifications using a network node that supports primary and secondary callback interfaces.

Figure 3–3 Failover in Network-Triggered Traffic



The following steps describe the workflow as illustrated in Figure 3–3.

1. A telecom network node sends a request to the Services Gatekeeper network plug-in that has been registered as the primary callback interface. It fails (1.1 in Figure 3–3) due to either a communication or server failure.
2. The telecom network node successfully resends the request to the plug-in that is registered as the secondary callback interface (1.2 in Figure 3–3). This plug-in is in a different server instance within the Network Tier cluster.
3. The Network Tier attempts to send the request to the callback mechanism in the Access Tier. The attempt fails (2.1 in Figure 3–3), either because the request fails to reach the Access Tier, or failure occurs during the marshalling or unmarshalling process.
4. The Network Tier resends the request, targeting another server in the Access Tier, and the attempt succeeds (2.2 in Figure 3–3).

Note: If 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 mechanism in the Access Tier attempts to send the request to the application (3.1 in Figure 3–3). If the application is unreachable or does not respond, the request is considered to have failed and an error is reported to the network node.

Registering Notifications with Network Nodes

Before applications can receive network-triggered traffic (notifications), they must register to do so with Services Gatekeeper, either by sending a request or having the operator set up the notification using management methods. In turn, these notifications must be registered with the underlying network node that will be supplying them. The form of this registration depends on the capabilities of that node.

If registration for notifications is supported by the underlying network node protocol, then the communication service's network plug-in performs the registration, whether the registration is the result of an application-initiated registration request or of an online provisioning step in Services Gatekeeper. All OSA/Parlay Gateway interfaces support such registration for notifications.

Note: Some network protocols support some, but not all registration types. For example, in MM7 an application can register to receive notifications for delivery reports on messages 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 off-line when provisioning the MMSC.

Whether the notification is set up in the network or by using OAM, Services Gatekeeper correlates all network-triggered traffic with its corresponding application.

Notification Life Span

Notifications are placed into three categories, based on the expected life span of the notification. These categories determine the failover strategies used:

- Short life span

  These notifications have an expected life span of a few seconds. Typically, these are delivery acknowledgments for delivering 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 have an expected life span of minutes up to a few days. Typically, these are acknowledgments for delivering the request to a user terminal. For this category, the delivery notification criteria that have been registered are replicated to one additional instance of the network protocol plug-in. The plug-in that receives the notification registers a secondary notification with the network node, if possible.

- Long life span

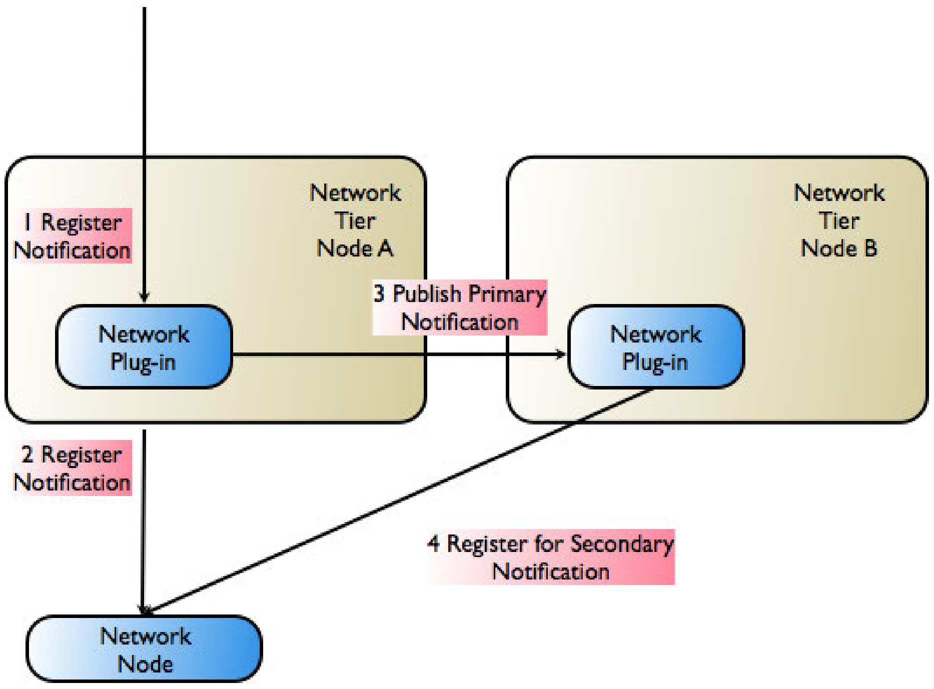
  These notifications have an expected life span of more than a few days. Typically, these are registrations for notifications of network-triggered SMS and MMS messages or calls that need to be handled by an application. For this category, the delivery notification criteria are replicated to all instances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

Primary and Secondary Notification

Figure 3–4 illustrates how Services Gatekeeper registers both primary and secondary notifications with network nodes that support them. This capability must be supported by the network protocol in the implementation of the protocol in both the network node and in the communication service's network plug-in.

Note: The scenario in Figure 3–4 assumes that the network node supports registration for notifications with overlapping criteria (primary/secondary).

Figure 3–4 Primary and Secondary Notification Registration



The following steps describe the workflow as illustrated in Figure 3–4.

1. The application sends a request to register for notifications to the network protocol plug-in.
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: Primary and secondary notifications are not necessarily ordered. The most recently registered notification may, for example, be designated as the primary notification.

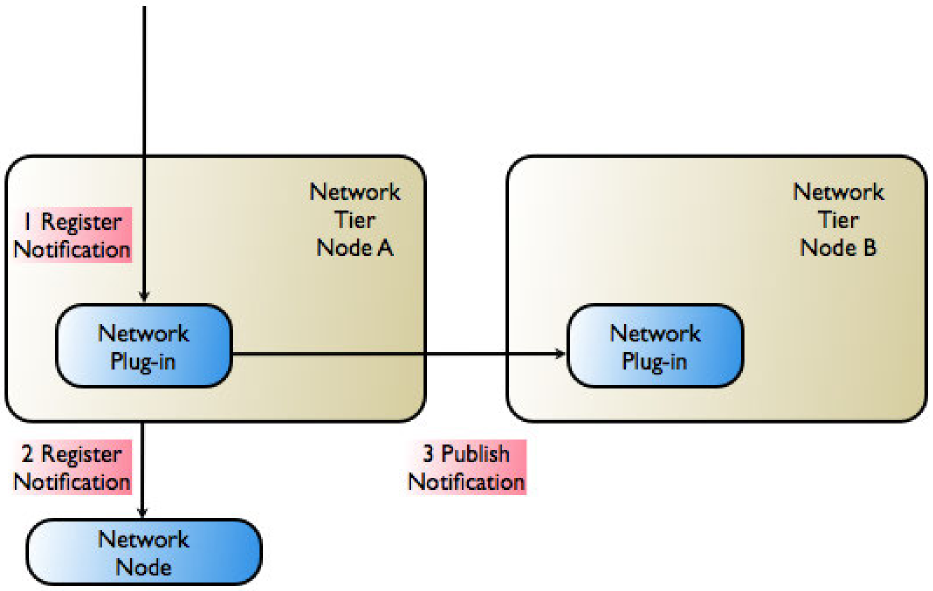
When a network-triggered request that matches the criteria in a previously registered notification reaches the telecom network node, the node first tries the network plug-in that registered as the primary callback interface. If the network-triggered request fails, the network node has the responsibility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

Single Notification

Figure 3–5 illustrates the registration step in Services Gatekeeper if the underlying network node does not support primary/secondary notification registration.

Note: The scenario in Figure 3–5 assumes that the network node does not support registration for notifications with overlapping criteria. Only one notification for a given criteria is allowed.

Figure 3–5 Single Notification Registration



The following steps describe the workflow as illustrated in Figure 3–5.

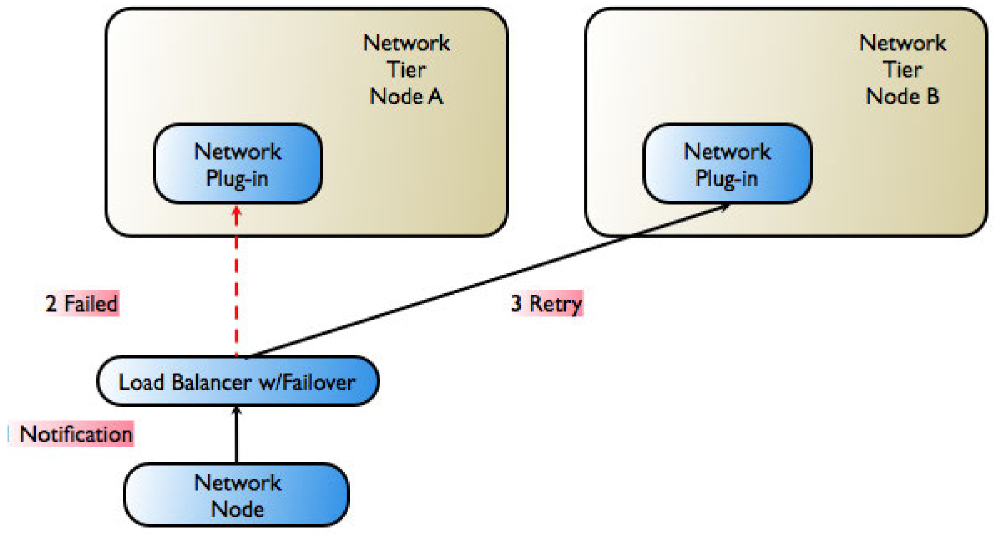
1. The application sends a request to register for notifications to the network protocol plug-in.
2. The notification request is registered with the telecom network node.
3. The notification information (matching criteria, target URL, and so on) is propagated to another instance of the network protocol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

There are two possibilities for high-availability and failover support in this case:

- All plug-ins can receive notifications from the network node. This is the case with SMPP, in which all plug-ins can function as receivers for any transmission from the network node.
- A load balancer with failover support is introduced between the network protocol plug-in and the network node. This is the case with HTTP- based protocols. Figure 3–6 illustrates a protocol-aware load balancer failing over traffic to an alternate plug-in.

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

Figure 3–6 Failover Performed by Load-Balancer



Multi-tier Services Gatekeeper Network Configuration

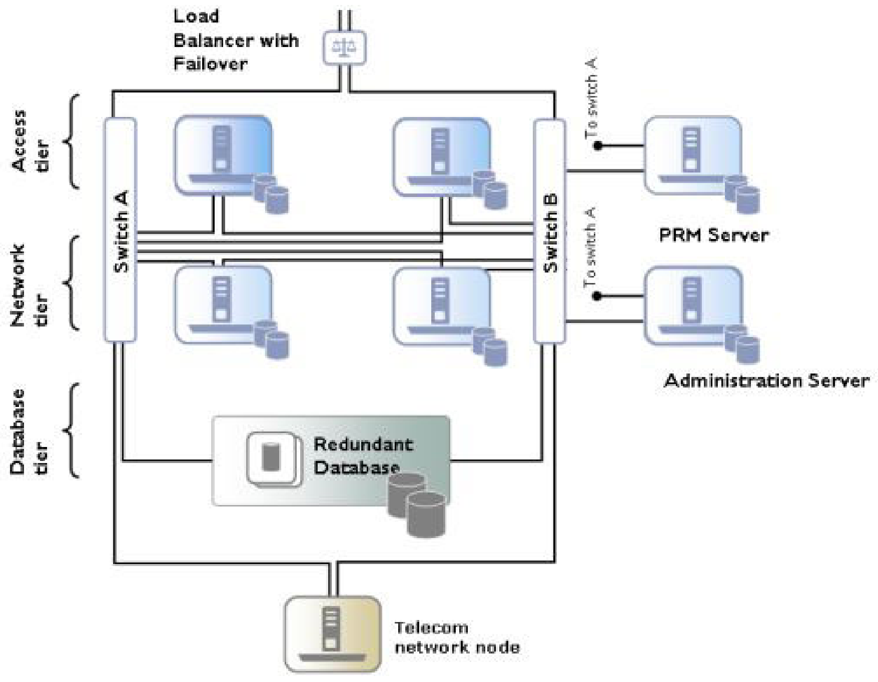
A multi-tier Services Gatekeeper production installation supports redundancy and high availability. A typical installation consists of a number of UNIX or Linux servers connected through duplicated switches. Each server has redundant network cards connected to separate switches. The servers are organized into clusters, with the number of servers in the cluster determined by the needed capacity.

Services Gatekeeper is deployed on 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, which consists of dedicated, redundant servers. For physical storage, you can optionally use a network-attached storage that uses fibre channel controller cards.

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

A sample configuration is shown in Figure 3–7. Smaller systems in which the Access Tier and the Network Tier are located on the same physical servers are possible but only for non-production systems. Particular hardware configurations depend on your specific deployment requirements.

Figure 3–7 Sample Hardware Configuration



In a high-availability configuration, all hardware components are duplicated, eliminating any 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, such as RAID.

The Administration Server may have duplicate network cards connected to each switch.

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

Geographic Redundancy in Multi-tier Services Gatekeeper

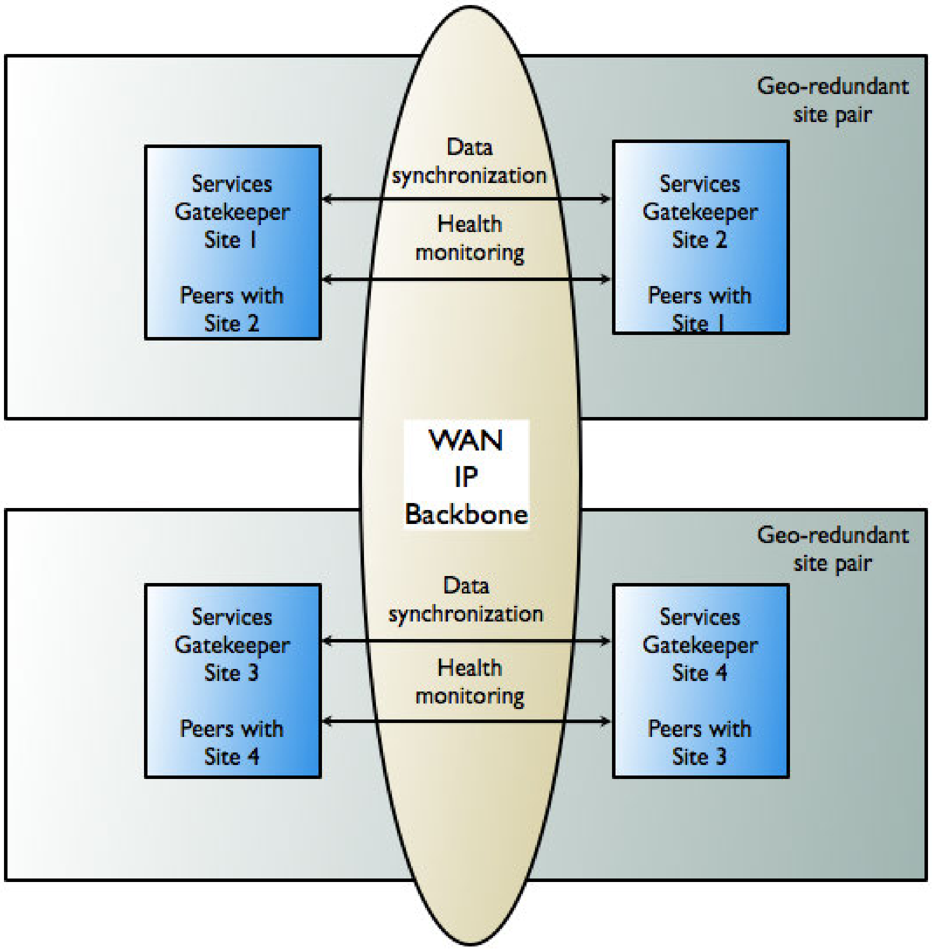
All mulit-tier Services Gatekeeper modules in production systems are deployed in clusters to ensure high availability. This prevents single points of failure in general usage. Within a cluster, the Budget container service regulates the enforcement of SLAs by using a cluster master that enforces bandwidth limits across the cluster, and slaves on each server that synchronize with the master. The Budget service is highly available and is migrated to another server if the cluster master node fails. See *Managing and Configuring Budgets* in *Services Gatekeeper System Administrator's Guide* for more information on this mechanism.

To prevent service failure during catastrophic events, such as natural disasters or massive system outages due to power failures, Services Gatekeeper can also be deployed at two geographically distant sites that are designated as site pairs. Each site, which is a Services Gatekeeper domain, has another site as its peer. Application and service provider configuration information, including related SLAs and budget information, is replicated and enforced across sites.

Note: Custom, Subscriber, Service Provider Node, and Global Node SLAs cannot be replicated across sites.

Figure 3–8 shows an overview of a geographically redundant setup.

Figure 3–8 Overview of Geographically Redundant Site Pairs



Geographically Redundant Sites

In a geographically redundant setup, all sites have a geographic site name and each site is configured to have a reference to its peer site using that name. The designated set of information is synchronized between these site peers.

One site is defined as the geomaster; the other as the slave. Checks are run periodically between the site pairs to verify data consistency. If mismatches are found, an alarm is triggered, at which point the administrator can force the slave to synchronize to the geomaster by using the **syncFromGeoMaster** operation from the Administration Console (**OCSG** -> *NT_servername* -> **Container Services** -> **GeoStorageService** -> **Operations** -> **syncFromGeoMaster**), or by using the **GeoStorageServiceMBean**. Any relevant configuration changes made to either site are written synchronously across the site pairs, so that a failure to write to either the geomaster or the slave causes the write to fail and triggers an alarm.

While the slave is synchronizing with the geomaster, both the geomaster and the slave sites are in read-only mode. No configuration changes can be made. If a slave site becomes unavailable for any reason, the geomaster site becomes read-only either until the slave site is available and has completed all data replication, or until the slave site has been removed from the geomaster site's configuration, terminating geographic redundancy.

If a new site is added to replace the terminated slave site, it must be added as a slave site. The site that is designated the geomaster site must remain the geomaster site for the lifetime of the site configuration.

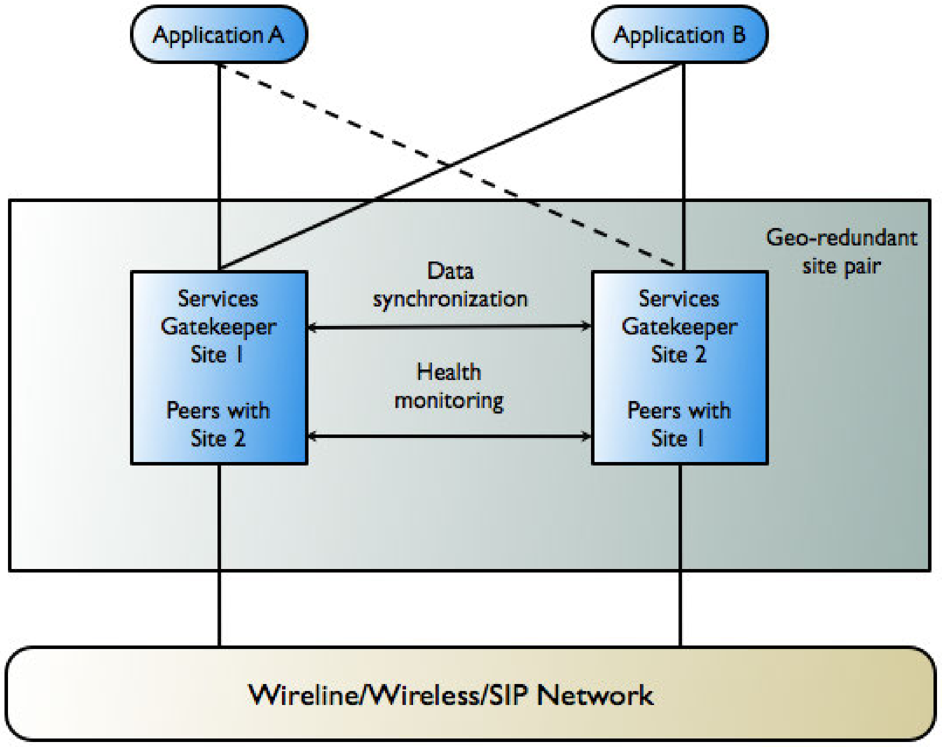
If a geomaster site fails permanently, the failed site should be removed from the configuration by using the GeoRedundantService container service. If a replacement site is added to the configuration, the remaining operating site must be reconfigured to be the geomaster and the replacement site must be added as the slave.

Applications and Geographic Redundancy

For applications, geographic redundancy means that their traffic can continue to flow if there is a catastrophic failure at an operator site. Even applications that normally use only a single site for their traffic can fail over to a peer site while maintaining ongoing SLA enforcement for their accounts. This scenario is particularly relevant for SLA aspects that have longer term impact, such as quotas.

Figure 3–9 shows an example of geographically redundant site pairs and applications.

Figure 3–9 Geographically Redundant Site Pairs and Applications



In many respects, the geographic redundancy is *not* transparent to applications. 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.

Application and service provider budget and configuration information are maintained across sites, but state for *ongoing conversations* is not maintained across sites. Conversations in this context are defined in terms of the correlation identifiers that are returned to the applications by Services Gatekeeper or are passed to Services Gatekeeper from the applications. Any state associated with a correlation identifier exists on only a single geographic site and is lost if a site-wide failure occurs. Conversational state includes, but is not limited to, call state and registration for network-triggered notifications. Conversational state is considered volatile, or transient, and is not replicated at the site level.

This means that conversations must be conducted and completed at their site of origin. If an application must maintain conversational state across sites, for example, to maintain a registration for network-triggered traffic, it must register with each site individually. This type of affinity does allow load balancing between sites for different or new conversations. For example, because each request to send an SMS message constitutes a new conversation, sending SMS messages can be balanced between the sites.

The following is a high-level outline of the redundancy functionality:

- The contractual usage relationships represented by SLAs can be enforced across geographic site domains. The enforcement covers service provider group and application group SLAs.
- Service provider and application account configuration data, including any changes to this data, can be replicated across sites. This reduces the administrative overhead of setting up geographically redundant site pairs.
- Alarms are generated when:
  - Peer sites fail to establish a connection with each other for a configurable number of times.
  - There is a configuration mismatch between the two sites; for example, if site A treats site B as a peer, but site B does not recognize site A as a peer
  - The paired sites do not have identical application and service provider configuration information, including related SLAs and budget information
  - The master site fails to complete a configuration update to the slave site

4 Extending Multi-tier Services Gatekeeper

This chapter explains how to extend and update the Oracle Communications Services Gatekeeper functionality.

About Extending Multi-tier Services Gatekeeper

Both Services Gatekeeper and multi-tier Services Gatekeeper use the Partner Relationship Management (PRM) portals for API management, including creating APIs and application subscriptions that you offer to your customers. You use the PRM portals to expose new services, service features, and network capabilities to your customers. These APIs can include the default Services Gatekeeper communication services.

Multi-tier Services Gatekeeper also offers the alternatives for extending Services Gatekeeper called the Platform Development Studio. You can use the Platform Development Studio to:

- Create new Oauth extension handlers
- Create custom Platform Test Environment (PTE) modules
- Create a Web services API exposure
- Add new network plug-ins that can work with existing application-facing interfaces
- Add new types of application-facing-interfaces (facades) for existing network plug-ins
- Add new interceptors or reorder interceptors
- Use EDR listeners and SNMP MIBs to integrate with the existing network mechanisms

Even if you primarily use the PRM portals for these tasks, you will also find the Platform Development Studio examples and information helpful to understand Services Gatekeeper components.

About the Platform Development Studio

The Platform Development Studio contains the tools that you use to extend Services Gatekeeper. The Platform Development Studio is installed automatically when you install a multi-tier Services Gatekeeper. See *Services Gatekeeper Multi-tier Installation Guide* and Services Gatekeeper Extension Developer's Guide for details. Development Studio includes the following features:

- A detailed guide covering how to:

  - Use the Platform Development Studio wizard to generate an extension project
  - Understand the example communication service
  - Use Service Gatekeeper container services
  - Create Subscriber SLAs for subscriber-based policy
  - Update EDR filters and add JMS-based listeners
  - Reorder the interceptor stack or create new interceptors
- *Services Gatekeeper Platform Test Environment User's Guide*

  A detailed guide covering how to:

  - Load the Platform Test Environment (PTE)
    - In standalone mode, with a Java Swing-based GUI
    - In console mode, particularly for use with the Unit Test Framework
  - Run the application service clients
  - Use the network simulators
  - Utilize the utilities such as the MBean browser and the JMS-based EDR listeners
  - Understand the example module and extend the PTE
- An example communication service, including:
  - Source code
  - WSDL files
  - Build files
  - Example unit test
  - Example Subscriber SLA and profile provider
- The Platform Development Studio wizard

  Developers supply information to an Eclipse plug-in wizard, which automatically sets up an extension project. This project can include a substantial amount of generated code, such as:

  - The entire Access Tier, with the Web service implementation and any callback modules (EJBs) that are necessary

    Note: The Platform Development Studio wizard supports building communication services based only on Web services application-facing interfaces. Both SOAP and RESTful facade types are supported.

  - Most of the code for the common services in the Network Tier
  - A framework of the code required for the network plug-in layer of the Network Tier
- Instructions for creating Virtual Communication Services to use with the PTE and ATE
- The Platform Development Studio Communications Service wizard

- A complete Javadoc reference
- Specialized templates and Ant tasks
- The Unit Test Framework, providing:
  - A base test class, derived from JUnit
  - Mechanisms for connecting to the Platform Test Environment
  - An example unit test bundled with the communication service example
- The Profile Provider service provider interface, which enables you to create subscriber-centric policy, associating subscribers with service provider and application groups based on individualized subscriber preferences and permissions.