Oracle® Communications Services Gatekeeper

Accounts and SLAs Guide

Release 5.1

E37537-01

June 2013

ORACLE®

Oracle Communications Services Gatekeeper Accounts and SLAs Guide, Release 5.1

E37537-01

Contents

4 Managing Groups

5 Managing SLAs

6 Managing Sessions

7 Defining Service Provider Group and Application Group SLAs

8 Defining Global Node and Service Provider Group Node SLAs

A Sample SLA

Preface

This document describes service provider and application provisioning for Oracle Communications Services Gatekeeper.

It covers:

- An overview of the administration model
- Managing service provider groups
- Managing service provider accounts
- Managing application groups
- Managing application accounts
- Managing application instances

Audience

The book is intended primarily for telecom operators who perform service provider and application provisioning tasks.

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For more information, see the following documents in the Oracle Communications Services Gatekeeper set:

- *Alarm Handling Guide*
- *Application Developer's Guide*
- *Communication Service Guide*
- *Concepts Guide*

- *Installation Guide*
- *Licensing Guide*
- *Partner Relationship Management Guide*
- *Platform Development Studio Developer's Guide*
- *Platform Test Environment Guide*
- *RESTful Application Developer's Guide*
- *SDK User's Guide*
- *Statement of Compliance*
- *System Administrator's Guide*
- *System Backup and Restore Guide*

1

About SLAs and Accounts

Oracle Communications Services Gatekeeper uses Service Level Agreements (SLAs) to define and enforce access to Services Gatekeeper and to the underlying telecommunication network.

SLAs are associated with application service providers through a tiered system of accounts that are organized into account groups.

About SLAs

An SLA is an XML document that defines and enforces access to Services Gatekeeper or, in the case of node SLAs, to the underlying network nodes. The core element in an SLA is a contract element whose child elements specify the details of the access. Examples of these details are the dates in which access is permitted, the number of requests to be processed within a certain time period, and the denial of the use of certain methods or parameter values to an application.

You can load an SLA into a Services Gatekeeper system from the Administration Console console using the operations provided in the ApplicationSLAs section of the AccountService container service. See "Managing SLAs" for information about the AccountService operations to load and manage SLAs.

As an alternative, you can manage SLAs through external management systems integrated with Services Gatekeeper using the Oracle Communications Services Gatekeeper Partner Relationship Management interfaces. For information about the Partner Relationship Management interfaces, see the *Oracle Communications Systems Gatekeeper Partner Relationship Management Guide*. You can also manage SLAs programmatically using MBeans. For information about the ServicelevelAgreement MBean, see the *Oracle Communications Services Gatekeeper OAM Java API Reference*.

System SLAs

System SLAs have static XML Schema Documents (XSDs). The enforcement logic is already in place for these types of SLAs, except for the subscriber SLA, which requires custom subscriber policy logic. The following types of SLAs are system SLAs:

- Service provider group SLA
- Application group SLA
- Service provider node SLA
- Global node SLA
- Subscriber SLA

Service provider group SLAs and application group SLAs define service provider and application access to Services Gatekeeper. See "Defining Service Provider Group and Application Group SLAs" for information about these types of SLAs and Appendix A for a complete example.

Node SLAs define access to the underlying network, either by service provider group or for Services Gatekeeper as a whole regardless of the service provider whose requests are being processed. See "Defining Global Node and Service Provider Group Node SLAs" for information about node SLAs.

Subscriber SLAs define subscriber policy and manage subscriber access. The implementor defines the subscriber policy logic using the Platform Development Studio. For more information, see the chapter on subscriber-centric policy in *Oracle Communications Services Gatekeeper Platform Development Studio Developer's Guid*e.

Custom SLAs

Custom SLAs are enforced by customized logic for which the XSDs are developed using the Platform Development Studio. For information on custom SLAs and an example of one, see the chapter on custom service level agreements in *Oracle Communications Services Gatekeeper Platform Development Studio Developer's Guid*e.

The following SLAs are custom SLAs:

- Custom service provider group SLAs
- Custom application group SLAs
- Custom global SLAs

The custom logic that operates on data is provided in the SLAs and on data that comes with the request. The data in the SLA is formatted according to the SLA XSD, and the enforcement logic uses the Document Object Model (DOM) representation of the XSD to get the data in the SLA. The request can provide information such as application ID, application instance, service provider ID, application group ID, and service provider group ID.

Custom SLAs can be associated with application groups and service provider groups or be global. A custom global SLA is valid for all requests, regardless of which application group or service provider group the application that triggered the request.

About Accounts

Each service provider and application that uses the Services Gatekeeper application-facing interfaces must establish an account.

Service provider accounts, application accounts, and application instances have states that can be changed using the Administration console. This enables the administrator to take them out of service temporarily.

Accounts

There are two types of accounts:

- Service Provider Account: Every service provider that has access to the operator's Services Gatekeeper facilities has a service provider account.
- Application Account: Every application that has access to the operator's Services Gatekeeper facilities has an application account. Each application account is associated with a service provider account.

See "Managing Service Provider and Application Accounts" for information about creating and managing accounts.

Account Groups

For the purposes of SLA enforcement, accounts are organized into account groups. There are two types of account groups:

- Service provider group
- Application group

When you load an Application Group SLA or Service Provider Group SLA, you provide the identifier of the group with which the SLA is associated along with the SLA.

Following is an example of how a Services Gatekeeper administrator might use system SLAs to create three different service provider groups:

- Bronze
- Silver
- Gold

Each of these service provider groups would be associated with different SLAs with different privileges regarding maximum throughput and Quality of Service (QoS parameters. When a new service provider is provisioned, the service provider is associated with the appropriate group based on, for example, the amount of network traffic the service provider is projected to generate. If the outcome is not what was predicted, the service provider account can be reassigned to another service provider group that has a different QoS parameter defined in its service provider SLA.

See "Managing Groups" for information about creating and managing groups.

Application Instances

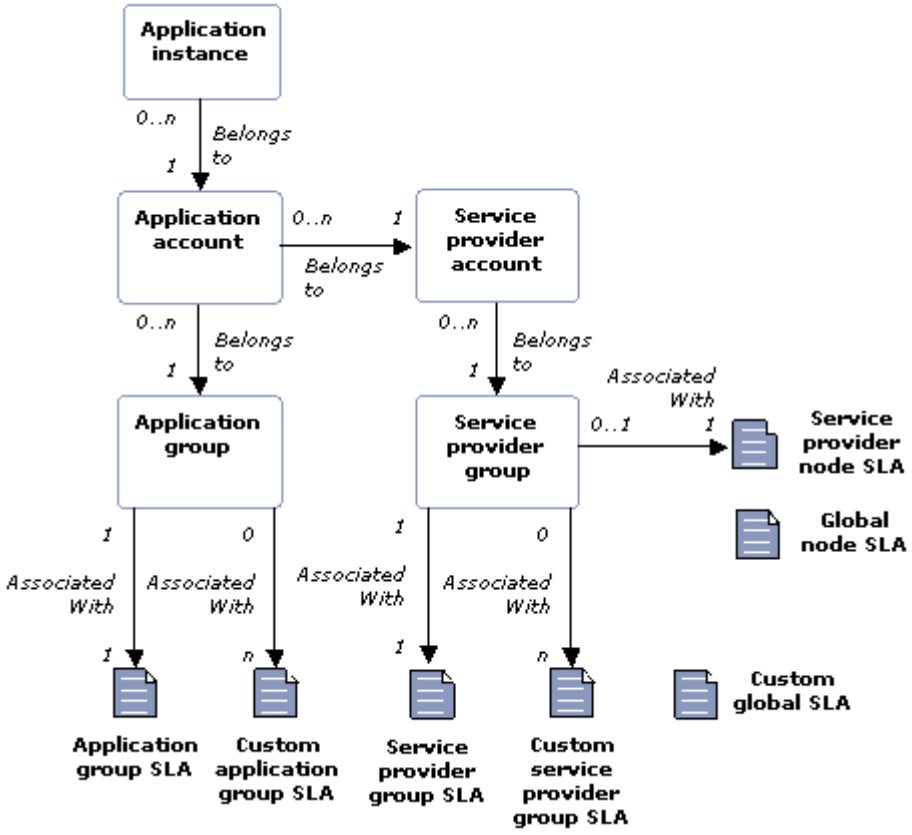
In the context of SLAs in Services Gatekeeper, an application instance is an application service provider. Each application instance has a user name and password or certificate. Each application instance is associated with a service provider account / application account combination. An application instance must be in the ACTIVATED state to send traffic through Services Gatekeeper.

See "Managing Application Instances" for more information about creating and managing application instances.

Associations Among SLA Types, Account Types, and Group Types

Figure 1–1 illustrates the associations among the different account types, group types, and SLA types.

Figure 1–1 Service provider and application model with SLAs



Note the following relationships:

- An *application instance*, which uniquely identifies an account, belongs to an *application account* and a *service provider account*.
- An *application account* belongs to a *service provider account* and an *application group*.
- A *service provider account* belongs to a *service provider group*. A service provider account can have zero or more application accounts associated with it.
- An *application group* is associated with an *application group SLA* and zero or more *custom application group SLAs*.
- *A service provider group* is associated with a service level agreement on the service provider-level and may also be associated with a *service provider node SLA* and zero or more *custom service provider group SLAs*.
- Zero or more custom global SLAs can be defined,. These are not associated with any special group or account, but are valid for all accounts and groups
- A *global node SLA* is not associated with any special group or account, but is valid for all accounts and groups.

IDs of service provider groups, application groups, application instances, and service provider accounts must be unique. The combination of service provider account and application account for an application instance must be unique.

The separation of accounts and groups provides a flexible way of defining service level agreements. Groups allow the Services Gatekeeper administrator to offer tiers of service at both the service provider and application level.

Application group SLAs and service provider group SLAs can be managed and enforced either locally in one site or across geo-redundant sites.

Typical SLA and Account Workflow

1. Create service provider and application groups that will correspond to the levels of service to be defined in the SLAs. See "Managing Groups" for more information.
2. Define the set of service classes, expressed as SLAs, at both the service provider group level and the application group level. See "Defining Service Provider Group and Application Group SLAs" for more information.
3. Define the set of node service classes, expressed as SLAs, at both the service provider node level and global node level. See "Defining Global Node and Service Provider Group Node SLAs" for more information.
4. Associate the service provider group SLAs and application group SLAs with the groups. See "Managing SLAs".
5. Load the global node SLA and service provider group node SLAs. See "Managing SLAs" for more information.
6. Create a service provider account and associate it with the appropriate service provider group. See "Managing Service Provider and Application Accounts" for more information.
7. Create an application account and associate it with the appropriate application group. See "Managing Service Provider and Application Accounts" for more information.
8. Create an application instance, see "Managing Application Instances" for more information.
9. Depending on which communication service is being used, provision communication service-specific data using the operation and management for relevant communication service.
10. Distribute the credentials to be used by the application to the service provider.

Using Partner Manager Portal to Manage SLAs and Accounts

Partner Manager Portal along with the companion application, Partner Portal act as "front-office" offerings provide a friendlier workflow, insulating you from the intricacies of Services Gatekeeper administration. Use the menu selections and/or input fields on well-defined pages of the two Portal applications to set up your SLAs and manage your service provider accounts.

Network operators and service provider managers use Partner Manager Portal to manage the following:

- Service Provider accounts
- Application accounts
- Service provider group SLAs
- Service provider group node SLAs
- Global node SLAs

Network operators authorize service providers to act as partners with access to Partner Portal and create applications using these SLAs. For more information on how to use these two applications, see Oracle Communications Services Gatekeeper Partner

Manager Portal Online Help and Oracle Communications Services Gatekeeper Partner Portal Online Help.

2

Managing Application Instances

This chapter describes how application instances are managed and provisioned in Services Gatekeeper:

- Summary of Tasks Related to Application Instances
- States of Application Instances
- Web Services Security
- Password Encryption

See "Reference: ApplicationInstances" for information about finding the operations in the Administration Console and for the name of the MBean.

Prior to registering application instances, service provider and application accounts must have been created. See "Managing Service Provider and Application Accounts"for more information.

Summary of Tasks Related to Application Instances

Table 2–1 lists the tasks related to application instances and the operations you use to perform those tasks.

Table 2–1 Tasks Related to Application Instances

Task	Operation to Use
Get information about the number of application instances	countApplicationInstances
Add, remove and get information about an application instance	addApplicationInstance removeApplicationInstance getApplicationInstance
List registered application instances	listApplicationInstances
Define additional properties for an application instance	setApplicationInstancePassword setApplicationInstanceProperties setApplicationInstanceReference setApplicationInstanceState

States of Application Instances

Application instances have states. Two states are possible:

- **ACTIVATED**

- **DEACTIVATED**

In **ACTIVATED** state, the application using this application instance ID is in normal operation.

In **DEACTIVATED** state, the application using this application instance ID is not allowed to send traffic through Oracle Communications Services Gatekeeper. The account is still valid and the state can be transitional by the Oracle Communications Services Gatekeeper Administrator using "setApplicationInstanceState".

Web Services Security

When Web Services Security is being used as authentication for the application instance, its credentials must be set up, as described in Table 2–2.

Table 2–2 Credential Mapping

ApplicationInstance	UserNameToken	X.509	SAML
ApplicationInstanceName	userName	CN	CN
Password	password	Not applicable	Not applicable

In addition, if X.509 or SAML are being used, the certificates and private keys must be provisioned in the WebLogic Server keystore. For example, `ImportPrivateKey` utility can be used to load the key and digital certificate files into the keystore. See ImportPrivateKey in *Oracle Fusion Middleware Command Reference for Oracle WebLogic Server* at:

`http://download.oracle.com/docs/cd/E15523_01/web.1111/e13749/utils.htm`

for a description of `ImportPrivateKey`.

Password Encryption

Application instance passwords are encrypted using 3DES algorithm and stored in persistent storage. If no encryption key is configured, a default encryption key is used, but best practice to use your own key. It should be 24 Bytes.

To configure the encryption key, from the Administration Console:

1. Select **Security realms** > **myrealm**.
2. Click the **Providers** tab.
3. Click the **Authentication** tab.
4. Click **WLNG Application Authenticator**.
5. In the Settings for WLNG Application Authenticator screen, click the **Configuration** tab.
6. Click the **Provider Specific** tab.
7. In the **Encryption Key** field, enter your key.
8. In **Please type again To confirm** field, enter your key again for verification.

Reference: ApplicationInstances

All operations are reachable from:

- Oracle Communications Services Gatekeeper Administration Console Managed Object: Container Services > AccountService > ApplicationInstances
- MBean: com.bea.wlcp.wlng.account.management.ApplicationInstanceMBean

Following is a list of operations for management.

- addApplicationInstance
- countApplicationInstances
- getApplicationInstance
- listApplicationInstances
- removeApplicationInstance
- setApplicationInstancePassword
- setApplicationInstanceProperties
- setApplicationInstanceReference
- setApplicationInstanceState

addApplicationInstance

The **addApplicationInstance** operation adds an application instance. Each instance is always connected to a service provider/application account combination.

Scope

Domain

Signature

```
addApplicationInstance(ApplicationInstanceName: String, Password: String,
ApplicationIdentifier: String, ServiceProviderIdentifier: String, Reference:
String)
```

Parameters

ApplicationInstanceName
ID for the new application instance. Used by an application to authenticate. Must be unique.

Password
Password for the application instance.

ApplicationIdentifier
ID of the application account the application instance shall belong to.

ServiceProviderIdentifier
ID of the service provider the application instance shall belong to.

Reference
See "Account References"

Must be unique.

countApplicationInstances

The **countApplicationInstances** operation displays the number of registered application instances for a service provider/application account/application instance combinations. Can be filtered by state.

Scope

Domain

Signature

```
countApplicationInstances(ApplicationIdentifier: String,
ServiceProviderIdentifier: String, State: String)
```

Parameters

ApplicationIdentifier
ID of the application account.

Optional. Leave empty to match all.

ServiceProviderIdentifier
ID of the service provider account.

Optional. Leave empty to match all.

State
State of application instance.

Valid values:

Optional. Leave empty to match all.

- **ACTIVATED**
- **DEACTIVATED**

getApplicationInstance

The **getApplicationInstance** operation gets information about a registered application instance. The information includes:

- ID of the application account to which the application instance belongs.
- ID of the service provider account to which the application instance belongs.
- State: See "States of Application Instances".
- Reference: See "Account References".
- Properties: See "Account Properties".

Scope

Domain

Signature

```
getApplicationInstance(ApplicationInstanceName: String)
```

Parameters

ApplicationInstanceName
ID of the application instance to get information about.

listApplicationInstances

The **listApplicationInstances** operation displays a list of registered application instances for a specific service provider/application account/application instance state combination.

The list contains application instance IDs.

Scope

Domain

Signature

```
listApplicationInstances(ApplicationIdentifier: String, ServiceProviderIdentifier:
String, State: String, Offset: int, Size: int)
```

Parameters

ApplicationIdentifier
ID of the application account to list application instances for.

Optional. Leave empty to match all.

ServiceProviderIdentifier
ID of the service provider account to list application instances for.

Optional. Leave empty to match all.

State
State of application instance.

Valid values:

- **ACTIVATED**
- **DEACTIVATED**

Optional. Leave empty to match all.

Offset
Offset in the list. Starts with 0 (zero)

Size
Size of the list.

For no restrictions on the size of the list, use 0 (zero).

removeApplicationInstance

The **removeApplicationInstance** operation removes a registered application instance.

The account must be in **DEACTIVATED** state.

Scope

Domain

Signature

```
removeApplicationInstance(ApplicationInstanceName: String)
```

Parameters

ApplicationInstanceName
ID of the application instance to remove.

setApplicationInstancePassword

The **setApplicationInstancePassword** operation defines a new password for an application instance.

Scope

Domain

Signature

```
setApplicationInstancePassword( ApplicationInstanceName: String, Password: String)
```

Parameters

ApplicationInstanceName
ID of the application instance.

Password
New password.

setApplicationInstanceProperties

The **setApplicationInstanceProperties** operation specifies properties for an application instance. See "Account Properties".

> **Note:** This application is applicable only when the MBean is accessed directly.

Scope

Domain

Signature

```
setApplicationInstanceProperties(ApplicationInstanceName: String, Properties:
Set<Map.Entry<String, String>>)
```

Parameters

ApplicationInstanceName
ID of the application instance to set properties for.

Properties
See "Account Properties".

setApplicationInstanceReference

The **setApplicationInstanceReference** operation specifies a reference for an application instance. See "Account References".

Scope

Domain

Signature

```
setApplicationInstanceReference(ApplicationInstanceName: String, Reference:
String)
```

Parameters

ApplicationInstanceName
ID of the application instance to set the reference for.

Reference
See "Account References".

setApplicationInstanceState

The **setApplicationInstanceState** operation performs a state transition for an application instance. See "States of Application Instances".

Scope

Domain

Signature

```
setApplicationInstanceState(ApplicationInstanceName: String, State: String)
```

Parameters

ApplicationInstanceName
ID of the application instance to change state for.

State
One of:

- **ACTIVATED**
- **DEACTIVATED**

See "States of Application Instances".

3

Managing Service Provider and Application Accounts

This chapter describes how service provider and application accounts are managed and provisioned. See the following sections:

- Summary of Tasks Related to Accounts
- States of Accounts
- Account Properties
- Account References
- Reference: Attributes and Operations for ApplicationAccounts

See "Reference: Attributes and Operations for ApplicationAccounts" for information on finding the operations in the Administration Console and for the name of the MBean.

Before you register application accounts, service provider accounts (see "Service Provider Accounts") and application groups (see "Application Groups") must be created.

Before you can register service provider accounts, service provider groups must be created. See "Service Provider Groups" for more information.

Summary of Tasks Related to Accounts

The following is a summary of tasks related to accounts.

Application Accounts

Table 3–1 lists the tasks related to application accounts and the operations you use to perform those tasks.

Table 3–1 Tasks Related to Application Accounts

Task	Operation to Use
Get information about the number of application accounts	countApplicationAccounts
Add, remove and get information about an application account	addApplicationAccount removeApplicationAccount getApplicationAccount
List registered application accounts	listApplicationAccounts

Table 3–1 (Cont.) Tasks Related to Application Accounts

Task	Operation to Use
Define additional properties for an application account	setApplicationAccountGroup setApplicationAccountProperties setApplicationAccountReference setApplicationAccountState

Service Provider Accounts

Table 3–2 lists the tasks related to service provider accounts and the operations you use to perform those tasks.

Table 3–2 Tasks Related to Service Provider Accounts

Task	Operation to Use
Get information about the number of service provider accounts	countServiceProviderAccounts
Add, remove and get information about a service provider account	addServiceProviderAccount removeServiceProviderAccount getServiceProviderAccount
List registered service provider accounts	listServiceProviderAccounts
Define additional properties for a service provider account	setServiceProviderAccountGroup setServiceProviderAccountProperties setServiceProviderAccountReference setServiceProviderAccountState

States of Accounts

To provide an easy way to take an application out-of-service temporarily, service provider and application accounts have associated states. States are also used by the Partner Relationship Management module to enforce workflow management.

Accounts can be in either one of the states described in Table 3–3:

Table 3–3 Account State Indicators

Account State	Restriction on Application Belonging to Account
ACTIVATED	An application belonging to such an account is allowed to send requests to the application-facing interfaces exposed by Services Gatekeeper.
DEACTIVATED	An application belonging to such an account is not allowed to send requests to the application-facing interfaces exposed by Oracle Communications Services Gatekeeper.

State transition is provided using:

- setApplicationAccountState
- setServiceProviderAccountState

It is possible to filter on account state in all the following methods:

- countApplicationAccounts

- countServiceProviderAccounts
- listApplicationAccounts
- listServiceProviderAccounts

When an account is created using "addApplicationAccount" or "addServiceProviderAccount", state is always ACTIVATED.

Account Properties

An account can have a set of associated properties. These are generic key-value pairs.You cannot set these using the Administration Console, but you can use MBeans or the PRM Web Services.

The properties are displayed in the following operations:

- getApplicationInstance
- getApplicationAccount
- getApplicationGroup
- getServiceProviderAccount
- getServiceProviderGroup

Account References

An account can have an associated reference. The reference is a form of alias or internal ID for the account. It is used to correlate the account with an operator-internal ID.

The references are defined as parameters in the following operations:

- addApplicationInstance
- setApplicationInstanceReference
- setApplicationInstanceProperties
- addApplicationAccount
- setApplicationAccountReference
- addServiceProviderAccount
- setApplicationAccountReference

They are retrieved as part of the result of:

- getApplicationInstance
- getApplicationAccount
- getServiceProviderAccount

Reference: Attributes and Operations for ApplicationAccounts

Managed object: Container Services > AccountService > ApplicationAccounts

MBean: com.bea.wlcp.wlng.account.management.ApplicationAccountMBean

Following is a list of operations for configuration and maintenance.

- addApplicationAccount

- addServiceProviderAccount
- countApplicationAccounts
- countServiceProviderAccounts
- getApplicationAccount
- getServiceProviderAccount
- listApplicationAccounts
- listServiceProviderAccounts
- removeApplicationAccount
- removeServiceProviderAccount
- setApplicationAccountGroup
- setApplicationAccountProperties
- setApplicationAccountReference
- setApplicationAccountState
- setServiceProviderAccountGroup
- setServiceProviderAccountProperties
- setServiceProviderAccountReference
- setServiceProviderAccountState

addApplicationAccount

The **addApplicationAccount** operation adds an application account.

Scope

Domain

Signature

```
addApplicationAccount(Application Identifier: String, Service Provider Identifier:
String, Application Group Identifier: String, Reference: String)
```

Parameters

Application Identifier
ID of the application account to add.

Service Provider Identifier
ID of the service provider account to which the application account belongs.

Application Group Identifier
ID of the application group to which the application account belongs.

Reference
See "Account References".

addServiceProviderAccount

The **addServiceProviderAccount** operation adds a service provider account.

Scope

Domain

Signature

```
addServiceProviderAccount(Service Provider Identifier: String, Service Provider
Group Identifier: String, Reference: String)
```

Parameters

Service Provider Identifier
ID of the service provider account to add.

The account will not be created if this parameter is blank spaces.

Service Provider Group Identifier
ID of the service provider group to which the service provider account belongs.

Reference
See "Account References".

countApplicationAccounts

The **countApplicationAccounts** operation displays the number of registered application accounts. The display can be filtered on any combination of:

- service provider account
- application group
- application account state.

Scope

Domain

Signature

```
countApplicationAccounts(serviceProviderIdentifier : String,
applicationGroupIdentifier: String, state : String)
```

Parameters

ServiceProviderIdentifier
ID of the service provider account.

Optional. Leave empty to match all.

ApplicationGroupIdentifier
ID of the application group.

Optional. Leave empty to match all.

State
State of application account. See "States of Accounts".

Valid values:

- **ACTIVATED**
- **DEACTIVATED**

Optional. Leave empty to match all.

countServiceProviderAccounts

The **countServiceProviderAccounts** operation displays the number of registered service provider accounts. Can be filtered on any combination of:

- service provider group
- service provider account state.

Scope

Domain

Signature

```
countServiceProviderAccounts(serviceProviderGroupIdentifier : String, state
:String)
```

Parameters

serviceProviderGroupIdentifier
ID of the service provider group.

Optional. Leave empty to match all.

State
State of service provider account. See "States of Accounts".

Valid values:

- **ACTIVATED**
- **DEACTIVATED**

Optional. Leave empty to match all.

getApplicationAccount

The **getApplicationAccount** operation gets information about an application account. Because the application account is not necessarily unique across service provider accounts, you must also specify the relevant service provider account.

The information includes:

- Application account ID.
- ID of the service provider account to which the application account belongs.
- State, see "States of Accounts".
- Reference, see "Account References".
- Properties, see Account Properties.

Scope

Domain

Signature

```
getApplicationAccount(Application Identifier: String, Service Provider Identifier:
String)
```

Parameters

Application Identifier

ID of the application account to get information about.

Service Provider Identifier

ID of the service provider account to which the application account belongs.

getServiceProviderAccount

The **getServiceProviderAccount** operation gets information about a service provider account.

The retrieved information includes:

- Service provider account ID.
- State, see "States of Accounts".
- Reference, see "Account References".
- Properties, see "Account Properties".

Scope

Domain

Signature

```
getServiceProviderAccount(Service Provider Identifier: String)
```

Parameters

Service Provider Identifier
ID of the service provider account.

listApplicationAccounts

The **listApplicationAccounts** operation displays a list of application accounts. The display can be filtered on any combination of:

- service provider account.
- application group.
- state of application account.

The resultant list contains application account IDs.

Scope

Domain

Signature

```
listApplicationAccounts(ServiceProviderIdentifier: String,
ApplicationGroupIdentifier : String, State: String, Offset: int, Size: int)
```

Parameters

ServiceProviderIdentifier
ID of the service provider account.

Optional. Leave empty to match all.

ApplicationGroupIdentifier
Application group ID.

Optional. Leave empty to match all.

State
State of application account. See "States of Accounts"

Valid values:

- **ACTIVATED**
- **DEACTIVATED**

Optional. Leave empty to match all.

Offset
Offset in the list. Starts with 0 (zero)

Size
Size of the list.

For no restrictions on the size of the list, use 0 (zero).

listServiceProviderAccounts

The **listServiceProviderAccounts** operation displays a list of service provider accounts. Can be filtered on any combination of:

- service provider group.
- state of service provider account.

The resultant list contains service provider account IDs.

Scope

Domain

Signature

```
listServiceProviderAccounts(ServiceProviderGroupIdentifier: String, State: String,
Offset: int, Size: int)
```

Parameters

ServiceProviderGroupIdentifier
Service provider group ID.

Optional. Leave empty to match all.

State
State of service provider account. See "States of Accounts"

Valid values:

- **ACTIVATED**
- **DEACTIVATED**

Optional. Leave empty to match all.

Offset
Offset in the list. Starts with 0 (zero)

Size
Size of the list.

For no restrictions on the size of the list, use 0 (zero).

removeApplicationAccount

The **removeApplicationAccount** operation removes an application account.

To be removed, the account must not have any associated application instances. The application account state must also be **DEACTIVATED**.

Because the application account is not necessarily unique across service provider accounts, you must also specify the relevant service provider account.

Scope

Domain

Signature

```
removeApplicationAccount(Application Identifier: String,Service Provider
Identifier: String)
```

Parameters

Application Identifier
ID of the application account to remove.

Service Provider Identifier
ID of the service provider account the application account belongs to.

removeServiceProviderAccount

The **removeServiceProviderAccount** operation removes a service provider account.

To be removed, the service provider account must not have any associated application accounts and must be in the **DEACTIVATED** state.

Scope

Domain

Signature

```
removeServiceProviderAccount(Service Provider Identifier: String)
```

Parameters

Service Provider Identifier
ID of the service provider account to remove.

setApplicationAccountGroup

The **setApplicationAccountGroup** operation sets the application account group for a particular application account/service provider account combination. Is also used to change to which group a combination belongs.

Scope

Domain

Signature

```
setApplicationAccountGroup(Application Identifier: String, Service Provider
Identifier: String, Application Group Identifier: String)
```

Parameters

Application Identifier
ID of the application account to be assigned.

Service Provider Identifier
ID of the service provider account associated with the application account to be assigned.

Application Group Identifier
ID of the application group to which the combination is to be assigned.

setApplicationAccountProperties

The **setApplicationAccountProperties** operation specifies properties for an application account/service provider account combination. See "Account Properties".

Note: Only applicable when accessing the MBean directly.

Scope

Domain

Signature

```
setApplicationAccountProperties(Application Identifier: String, Service Provider
Identifier: String, Properties: Set<Map.Entry<String, String>>)
```

Parameters

Application Identifier

ID of the application account whose properties are being set.

Service Provider Identifier

ID of the service provider account to which the application account belongs.

Properties

See "Account Properties".

setApplicationAccountReference

The **setApplicationAccountReference** operation specifies a reference for an application account/service provide account combinations. See "Account References".

Scope

Domain

Signature

```
setApplicationAccountReference(Application Identifier: String, Service Provider
Identifier: String, Reference: String)
```

Parameters

Application Identifier
ID of the application account whose reference is being set.

Service Provider Identifier
ID of the service provider account to which the application account belongs.

Reference
See "Account References".

setApplicationAccountState

The **setApplicationAccountState** operation performs a state transition for an application account/service provide account combination. See "States of Accounts".

Scope

Domain

Signature

```
setApplicationAccountReference(Application Identifier: String, Service Provider
Identifier: String, State: String)
```

Parameters

Application Identifier

ID of the application account whose state is to be changed.

Service Provider Identifier

ID of the service provider account to which the application account belongs.

State

One of:

- **ACTIVATED**
- **DEACTIVATED**

See "States of Accounts".

setServiceProviderAccountGroup

The **setServiceProviderAccountGroup** operation specifies to which service provider group a certain service provider account belongs. Can be used as well to change the group assignment.

Scope

Domain

Signature

```
setServiceProviderAccountGroup(Service Provider Identifier: String, Service
Provider Group Identifier: String)
```

Parameters

Service Provider Identifier
ID of the service provider account being assigned.

Service Provider Group Identifier
ID of the service provider group to which the account is being assigned.

setServiceProviderAccountProperties

The **setServiceProviderAccountProperties** operation specifies properties for a service provider account. See "Account Properties".

> **Note:** Only applicable when accessing the MBean directly.

Scope

Domain

Signature

```
setServiceProviderAccountProperties(Service Provider Identifier: String,
Properties: Set<Map.Entry<String, String>>)
```

Parameters

Service Provider Identifier

ID of the service provider account whose properties are being set.

Properties

See "Account Properties".

setServiceProviderAccountReference

The **setServiceProviderAccountReference** operation specifies a reference for a service provider account. See "Account References".

Scope

Domain

Signature

```
setServiceProviderAccountReference(Service Provider Identifier: String, Reference: String)
```

Parameters

Service Provider Identifier
ID of the service provider account whose reference is being set.

Reference
See "Account References".

setServiceProviderAccountState

The **setServiceProviderAccountState** operation performs a state transition for a service provider account. See "States of Accounts".

Scope

Domain

Signature

```
setServiceProviderAccountState(Service Provider Identifier: String, State: String)
```

Parameters

Service Provider Identifier

ID of the service provider account whose state is being changed.

State

One of:

- **ACTIVATED**
- **DEACTIVATED**

See "States of Accounts".

4

Managing Groups

This chapter describes how service provider and application groups are managed and provisioned. See the following sections:

- Summary of Tasks Related to Groups
- Reference: Attributes and Operations for ApplicationGroups

See "Reference: Attributes and Operations for ApplicationGroups" for information finding the operations in the Administration Console and for the name of the MBean.

Before you can register application and service provider groups, SLAs must be created. See "Defining Service Provider Group and Application Group SLAs".

Summary of Tasks Related to Groups

The following is a summary of tasks related to groups.

Application Groups

Table 4–1 describes the tasks related to application groups and the operations you use to perform those tasks.

Table 4–1 Tasks Related to Application Groups

Task	Operation to Use
Get information about the number of application groups	countApplicationGroups
Add, remove and get information about an application group	addApplicationGroup removeApplicationGroup getApplicationGroup
List registered application groups	listApplicationGroups
Define additional properties for an application group	setApplicationGroupProperties

Service Provider Groups

Table 4–2 describes the tasks related to service provider groups and the operations you use to perform those tasks.

Table 4–2 Tasks Related to Service Provider Groups

Task	Operation to Use
Get information about the number of service provider groups	countServiceProviderGroups
Add, remove and get information about a service provider group	addServiceProviderGroup removeServiceProviderGroup getServiceProviderGroup
List registered service provider groups	listServiceProviderGroups
Define additional properties for an service provider group	setServiceProviderGroupProperties

Reference: Attributes and Operations for ApplicationGroups

Managed object: Container Services > AccountService > ApplicationGroups

MBean: com.bea.wlcp.wlng.account.management.ApplicationGroupMBean

Following is a list of operations for configuration and maintenance.

- addApplicationGroup
- addServiceProviderGroup
- countApplicationGroups
- countServiceProviderGroups
- getApplicationGroup
- getServiceProviderGroup
- listApplicationGroups
- listServiceProviderGroups
- removeApplicationGroup
- removeServiceProviderGroup
- setApplicationGroupProperties
- setServiceProviderGroupProperties

addApplicationGroup

The **addApplicationGroup** operation adds a new application group.

Scope

Domain

Signature

```
addApplicationGroup(ApplicationGroupIdentifier: String)
```

Parameters

ApplicationGroupIdentifier
Unique ID for the application group

addServiceProviderGroup

The **addServiceProviderGroup** operation adds a new service provider group.

Scope

Domain

Signature

```
addServiceProviderGroup( ServiceProviderGroupIdentifier: String)
```

Parameters

ServiceProviderGroupIdentifier
Unique ID for the service provider group

countApplicationGroups

The **countApplicationGroups** operation displays the number of application groups.

Scope

Domain

Signature

```
countApplicationGroups()
```

countServiceProviderGroups

The **countServiceProviderGroups** operation displays the number of service provider groups.

Scope

Domain

Signature

```
countServiceProviderGroups()
```

getApplicationGroup

The **getApplicationGroup** operation displays information about an application group, including:

- Application group description.
- Properties; see "Account Properties" for more information.

Scope

Domain

Signature

```
getApplicationGroup(ApplicationGroupIdentifier: String)
```

Parameters

ApplicationGroupIdentifier
ID of the application group whose description is desired.

getServiceProviderGroup

The **getServiceProviderGroup** operation displays information about a service provider group.

The information includes:

- Service provider group description.
- Properties; see "Account Properties" for more information.

Scope

Domain

Signature

```
getServiceProviderGroup(ServiceProviderGroupIdentifier: String)
```

Parameters

ServiceProviderGroupIdentifier
ID of the service provider group for which to get information.

listApplicationGroups

The **listApplicationGroups** operation displays a list of application groups.

The resultant list contains application group IDs.

Scope

Domain

Signature

```
listApplicationGroups(Offset: int, Size: int)
```

Parameters

Offset
Offset in the list. Starts with 0 (zero).

Size
Size of the list.

For no restrictions on the size of the list, use 0 (zero).

listServiceProviderGroups

The **listServiceProviderGroups** operation displays a list of service provider groups.

The resultant list contains service provider group IDs.

Scope

Domain

Signature

```
listServiceProviderGroups(Offset: int, Size: int)
```

Parameters

Offset
Offset in the list. Starts with 0 (zero).

Size
Size of the list.

For no restrictions on the size of the list, use 0 (zero).

removeApplicationGroup

The **removeApplicationGroup** operation removes an existing application group. The group must not have any applications associated with it.

Scope

Domain

Signature

```
removeApplicationGroup(ApplicationGroupIdentifier: String)
```

Parameters

ApplicationGroupIdentifier
ID for the application group.

removeServiceProviderGroup

The **removeServiceProviderGroup** operation removes an existing service provider group. The group must not have any service providers associated with it.

Scope

Domain

Signature

```
removeServiceProviderGroup(ServiceProviderGroupIdentifier: String)
```

Parameters

ServiceProviderGroupIdentifier
ID for the service provider group.

setApplicationGroupProperties

The **setApplicationGroupProperties** operation specifies properties for an application group.

> **Note:** This operation is applicable only when the MBean is accessed directly.

Scope

Domain

Signature

```
setApplicationGroupProperties(ApplicationGroupIdentifier: String, Properties:
Set<Map.Entry<String, String>>)
```

Parameters

ApplicationGroupIdentifier
ID of the application group whose properties are to be set.

Properties
See "Account Properties" for more information.

setServiceProviderGroupProperties

The **setServiceProviderGroupProperties** operation specifies properties for a service provider group.

> **Note:** This operation is applicable only when the MBean is accessed directly.

Scope

Domain

Signature

```
setServiceProviderGroupProperties(ServiceProviderGroupIdentifier: String,
Properties: Set<Map.Entry<String, String>>)
```

Parameters

ServiceProviderGroupIdentifier
ID of the service provider group whose properties are to be set.

Properties
See "Account Properties" for more information.

5

Managing SLAs

This chapter describes how the different types of SLAs are managed and provisioned. See the following sections:

Introduction to SLA types

See "Reference: ApplicationSLAs" for information on finding the operations in the Administration Console and for the name of the MBean.

There are two different kinds of SLAs:

- System level SLAs
- Custom SLAs

System level SLAs have static XSDs that are already defined in Services Gatekeeper, while custom SLAs provides the possibility to use a custom XSD. Any given service provider group or application group can have only one system SLA associated with it, while they can have many custom SLAs. Create custom SLAs when additional SLA enforcement logic is required that is not provided by default in Services Gatekeeper. The enforcement logic for a Custom SLA needs to be created. See *Platform Development Studio Developer's Guide* for information on how to develop Custom SLAs.

When SLAs are loaded into memory, they are stored in the SLA repository. SLAs can be loaded into the repository in two ways:

- The contents of the SLA is provided as a parameter in the operation that loads the SLA.
- The SLA is stored in a file and the URL to that file is provided as a parameter in the operation that loads the SLA.

SLAs are retrieved from the SLA repository using the retrieve operations.

The SLA loaded into the SLA repository is the one being enforced. Changes to the file after the SLA is loaded into the repository are not automatically reflected in the active version.

Table 5–1 outlines the different SLA types, the IDs and their scope.

Table 5–1 SLA Types

SLA Type	ID(s)	Description
Application Group	application system:geo_application	System SLA. Defines how application can use Services Gatekeeper. See "Application Group SLAs" for a summary of related management operations. For information on creating these SLAs, see "Defining Service Provider Group and Application Group SLAs". The SLA type ID: ■ **application** indicates that the SLA is loaded to and enforced in the network tier cluster it is loaded. ■ **system:geo_application** indicates that the SLA is loaded to and enforced across all network tier clusters in a geo-redundant configuration.
Service Provider Group	service_provider system:geo_service_provider	System SLA. Defines how service providers can use Services Gatekeeper. See "Service Provider Group SLAs" for a summary of related management operations. For information on creating these SLAs, see "Defining Service Provider Group and Application Group SLAs". The SLA type ID: ■ **service_provider** indicates that the SLA is loaded to and enforced in the network tier cluster it is loaded. ■ **system:geo_service_provider** indicates that the SLA is loaded to and enforced across all network tier clusters in a geo-redundant configuration.
Global Node	global_node	System SLA. Defines how Services Gatekeeper is allowed to use the underlying telecom network nodes. See "Global Node SLAs" for a summary of related management operations. For information on creating these SLAs, see "Defining Service Provider Group and Application Group SLAs". This SLA type is loaded and enforced locally in the network tier cluster it is loaded.
Service Provider Node	service_provider_node	System SLA. Defines how Service Providers are allowed to use the underlying telecom network nodes. See "Service Provider Node SLAs". For information on creating these SLAs, see "Defining Service Provider Group and Application Group SLAs". This SLA type is loaded and enforced locally in the network tier cluster it is loaded.

Table 5–1 (Cont.) SLA Types

SLA Type	ID(s)	Description
Subscriber	subscr	System SLA. Defines classes of application services that can be associated with subscribers in the context of Services Gatekeeper. Using the Platform Development Studio, an operator or integrator may create a subscriber-centric policy mechanism. The specifics of this mechanism are covered in the "Subscriber-centric Policy" chapter in the *Oracle Communications Services Gatekeeper Platform Development Studio Developer's Guide*, a separate document in this set. This SLA type is loaded and enforced locally in the network tier cluster on which it is loaded.
Custom	Defined at load time	Custom SLA. A custom SLA is defined by an XSD, that must be created and loaded. A custom SLA type ID is associated with the XSD, and this type is the one being referenced when loading the custom SLAs. In addition to the SLA, the enforcement logic that operates on the data in the SLA must be created. See the section Custom Service Level Agreements in Oracle Communications Services Gatekeeper *Platform Development Studio Developer's Guide*. Just like system SLAs, the custom SLAs are associated with service provider groups and application groups. In addition there is a custom global SLA, that does not take into consideration the originator of the request, but affects all requests. SLAs of custom type are loaded and enforced locally in the network tier cluster it is loaded.

Note: The prefix system is reserved, and should not be used by custom SLAs. For reasons of backwards compatibility, there is a set of SLA type without this prefix.

Summary of Tasks Related to SLAs

This provides a summary of tasks related to the management of SLAs.

Service Provider and Application Group System SLAs

The first group concerns Service Provider and Application Group System SLAs.

Application Group SLAs

Table 5–2 describes the tasks related to managing Application Group SLAs and the operations you use to perform those tasks.

Table 5–2 Tasks Related to Managing Application Group SLAs

Task	Operation to Use
Associate Application Group with SLA	"loadApplicationGroupSlaByType" loadApplicationGroupSlaFromUrlByType Deprecated: loadApplicationGroupSla loadApplicationGroupSlaFromUrl
View Application Group SLA	retrieveApplicationGroupSlaByType Deprecated: retrieveApplicationGroupSla

Service Provider Group SLAs

Table 5–3 describes the tasks related to managing Service Provider Group SLAs and the operations you use to perform those tasks.

Table 5–3 Tasks Related to Managing Service Provider Group SLAs

Task	Operation to Use
Associate Service Provider Group with SLA	loadServiceProviderGroupSlaByType loadServiceProviderGroupSlaFromUrlByType Deprecated: loadServiceProviderGroupSla loadServiceProviderGroupSlaFromUrl
View Service Provider Group SLA	retrieveServiceProviderGroupSlaByType Deprecated: retrieveServiceProviderGroupSla

Node SLAs

This group describes Node SLA management.

Global Node SLAs

Table 5–4 describes the tasks related to managing Global Node SLAs and the operations you use to perform those tasks.

Table 5–4 Tasks Related to Managing Global Node SLAs

Task	Operation to Use
Associate Global Node with SLA	loadGlobalSlaByType loadGlobalSlaFromUrlByType Deprecated: loadGlobalNodeSla
View Global Node SLA	retrieveGlobalSlaByType Deprecated: retrieveGlobalNodeSla

Service Provider Node SLAs

Table 5–5 describes the tasks related to Service Provider Node SLAs and the operations you use to perform those tasks.

Table 5–5 Tasks Related to Managing Service Provider Node SLAs

Task	Operation to Use
Associate Service Provider Node with SLA	loadServiceProviderGroupSlaByType loadServiceProviderGroupSlaFromUrlByType Deprecated: loadServiceProviderGroupNodeSla loadServiceProviderGroupNodeSlaFromUrl
View Service Provider Node SLA	retrieveServiceProviderGroupSlaByType Deprecated: retrieveServiceProviderGroupNodeSla

Subscriber SLAs

Subscriber SLAs are a feature that can be developed using the Platform Development Studio. Table 5–6 describes the tasks related to managing Subscriber SLAs and the operations you use to perform those tasks.

Table 5–6 Tasks Related to Managing Subscriber SLAs

Task	Operation to Use
Associate Subscriber with SLA	loadGlobalSlaByType loadGlobalSlaFromUrlByType Deprecated: loadSubscriberSla loadSubscriberSlaFromUrl
View Subscriber SLA	retrieveGlobalSlaByType Deprecated: retrieveSubscriberSla

Custom SLAs

Custom SLAs are a feature that can be developed using the Platform Development Studio. This section describes the management of Custom SLAs.

Custom XSDs

Table 5–7 describes the tasks related to managing Custom XSDs and the operations you use to perform those tasks.

Table 5–7 Tasks Related to Managing Custom XSDs

Task	Operation to Use
Set up Custom XSD	setupCustomSlaXSDDefinition setupCustomSlaXSDDefinitionFromUrl
View Custom XSD	retrieveCustomSlaXSDDefinition
Count Custom XSD	countCustomSlaXSDDefinition

Table 5–7 (Cont.) Tasks Related to Managing Custom XSDs

Task	Operation to Use
List Custom XSD	listCustomSlaXSDDefinition

Custom Application Group SLAs

Table 5–8 describes the tasks related to managing Custom Application Group SLAs and the operations you use to perform those tasks.

Table 5–8 Tasks Related to Managing Custom Application Group SLAs

Task	Operation to Use
Associate Custom Application Group with SLA	loadApplicationGroupSlaByType loadApplicationGroupSlaFromUrlByType
View Custom Application Group SLA	retrieveApplicationGroupSlaByType
Count Custom Application Group SLA	countApplicationGroupsByType
List Custom Application Group SLA	listApplicationGroupsByType

Custom Service Provider Group SLAs

Table 5–9 describes the tasks related to managing Custom Service Provider Group SLAs and the operations you use to perform those tasks.

Table 5–9 Tasks Related to Managing Custom Service Provider Group SLAs

Task	Operation to Use
Associate Custom Service Provider Group with SLA	loadServiceProviderGroupSlaByType loadServiceProviderGroupSlaFromUrlByType
View Custom Service Provider Group SLA	retrieveServiceProviderGroupSlaByType
Count Custom Service Provider Group SLA	countServiceProviderGroupsByType
List Custom Service Provider Group SLA	listServiceProviderGroupSlaTypes listServiceProviderGroupsByType

Custom Global SLAs

Table 5–10 describes the tasks related to managing Custom Global SLAs and the operations you use to perform those tasks.

Table 5–10 Tasks Related to Managing Custom Global SLAs

Task	Operation to Use
Associate Custom Global with SLA	loadGlobalSlaByType loadGlobalSlaFromUrlByType
View Custom Global SLA	retrieveGlobalSlaByType
Count Custom Global SLA	countGlobalSlaTypes

Table 5–10 (Cont.) Tasks Related to Managing Custom Global SLAs

Task	Operation to Use
List Custom Global SLA	listGlobalSlaTypes

Reference: ApplicationSLAs

All operations are reachable from:

- Services Gatekeeper Administration Console Managed Object: ApplicationSLAs
- MBean: com.bea.wlcp.wlng.account.management.ServiceLevelAgreementMBean

Following is a list of operations for management:

- countApplicationGroupsByType
- countApplicationGroupSlaTypes
- countApplicationSlaGroups
- countCustomSlaXSDDefinition
- countGlobalSlaTypes
- countServiceProviderGroupsByType
- countServiceProviderGroupSlaTypes
- countServiceProviderSlaGroups
- listApplicationGroupsByType
- listApplicationGroupSlaTypes
- listApplicationSlaGroups
- listCustomSlaXSDDefinition
- listGlobalSlaTypes
- listServiceProviderGroupsByType
- listServiceProviderGroupSlaTypes
- listServiceProviderSlaGroups
- loadApplicationGroupSla
- loadApplicationGroupSlaByType
- loadApplicationGroupSlaFromUrl
- loadApplicationGroupSlaFromUrlByType
- loadGlobalNodeSla
- loadGlobalNodeSlaFromUrl
- loadGlobalSlaByType
- loadGlobalSlaFromUrlByType
- loadServiceProviderGroupNodeSla
- loadServiceProviderGroupNodeSlaFromUrl
- loadServiceProviderGroupSla
- loadServiceProviderGroupSlaFromUrl

- loadServiceProviderGroupSlaFromUrlByType
- loadServiceProviderGroupSlaByType
- loadSubscriberSla
- loadSubscriberSlaFromUrl
- retrieveApplicationGroupSla
- retrieveApplicationGroupSlaByType
- retrieveCustomSlaXSDDefinition
- retrieveGlobalNodeSla
- retrieveGlobalSlaByType
- retrieveServiceProviderGroupNodeSla
- retrieveServiceProviderGroupSla
- retrieveServiceProviderGroupSlaByType
- retrieveSubscriberSla
- setupCustomSlaXSDDefinition
- setupCustomSlaXSDDefinitionFromUrl

countApplicationGroupsByType

The **countApplicationGroupsByType** operation displays the number of a application groups that are associated with a specific system SLA type or custom SLA type.

Scope

Domain

Signature

```
countApplicationGroupsByType(SlaType: String)
```

Parameters

SlaType
The system SLA type or custom SLA type.

countApplicationGroupSlaTypes

The **countApplicationGroupSlaTypes** operation displays the number of system and custom SLA types for an application group.

Scope

Domain

Signature

```
countApplicationGroupSlaTypes(applicationGroupIdentifier: String)
```

Parameters

applicationGroupIdentifier
The ID of the application group.

countApplicationSlaGroups

The **countApplicationSlaGroups** operation is deprecated. Use "countApplicationGroupsByType" with slaType **application**.

Information on **countApplicationSlaGroups** is provided here for backward-compatibility only.

This operation displays the number of registered application groups that have SLAs associated with them.

Scope

Domain

Signature

```
countApplicationSlaGroups()
```

countCustomSlaXSDDefinition

The **countCustomSlaXSDDefintions** operation displays the number of registered custom SLA types.

Scope

Domain

Signature

```
countCustomSlaXSDDefinitions()
```

countGlobalSlaTypes

The **countGlobalSlaTypes** operation displays the number of registered SLA types that have global scope.

Scope

Domain

Signature

```
countGlobalSlaTypes()
```

countServiceProviderGroupsByType

The **countServiceProviderGroupsByType** operation displays the number of service provider groups that are associated with a specific SLA type.

Scope

Domain

Signature

```
countServiceProviderGroupsByType(slaType: String)
```

Parameters

slaType
The system SLA type or custom SLA type.

For custom SLAs, the SLA type was registered in "setupCustomSlaXSDDefinition".

countServiceProviderGroupSlaTypes

The **countServiceProviderGroupsSlaTypes** operation displays the number of system SLA types and custom SLA types for a service provider group.

Scope

Domain

Signature

```
countServiceProviderGroupSlaTypes(serviceProviderGroupIdentifier: String)
```

Parameters

serviceProviderGroupIdentifier
The ID of the service provider group.

countServiceProviderSlaGroups

The **countServiceProviderSlaGroups** operation is deprecated. Use "countServiceProviderGroupsByType" with slaType **service_provider**.

The **countServiceProviderSlaGroups** operation displays the number of registered service provider groups that have SLAs associated with them.

Scope

Domain

Signature

```
countServiceProviderGroups()
```

listApplicationGroupsByType

The **listApplicationGroupsByType** operation displays a list of registered application groups that have a system level or custom SLA associated, filtered by the SLA type.

The list contains application group IDs.

Scope

Domain

Signature

```
listApplicationGroupsByType(slaType: String, Offset: int, Size: int)
```

Parameters

slaType
The SLA type.

For custom SLAs, the SLA type was registered in "setupCustomSlaXSDDefinition".

For system level SLAs, see " SLA Types".

Offset
Offset in the list. Starts with 0 (zero)

Size
Size of the list.

For no restrictions on the size of the list, use 0 (zero).

listApplicationGroupSlaTypes

The **listApplicationGroupSlaTypes** operation displays a list of system and custom SLAs for an application group.

Scope

Domain

Signature

```
listApplicationGroupSlaTypes(applicationGroupIdentifier: String, Offset: int,
Size: int)
```

Parameters

applicationGroupIdentifier
The ID of the application group.

Offset
Offset in the list. Starts with 0 (zero)

Size
Size of the list.

For no restrictions on the size of the list, use 0 (zero).

listApplicationSlaGroups

The **listApplicationSlaGroups** operation displays a list of registered application groups that has application SLAs associated.

The list contains application group IDs.

Scope

Domain

Signature

```
listApplicationSlaGroups(Offset: int, Size: int)
```

Parameters

Offset
Offset in the list. Starts with 0 (zero)

Size
Size of the list.

For no restrictions on the size of the list, use 0 (zero).

listCustomSlaXSDDefinition

The **listApplicationSlaGroups** operation displays a list of registered custom SLA XSD.

The list contains custom SLA types.

Scope

Domain

Signature

```
listCustomSlaXSDDefinition(Offset: int, Size: int)
```

Parameters

Offset
Offset in the list. Starts with 0 (zero)

Size
Size of the list.

For no restrictions on the size of the list, use 0 (zero).

listGlobalSlaTypes

The **listGlobalSlaTypes** operation displays a list of system and custom global SLA types.

Scope

Domain

Signature

```
listGlobalSlaTypes(Offset: int, Size: int)
```

Parameters

Offset
Offset in the list. Starts with 0 (zero)

Size
Size of the list.

For no restrictions on the size of the list, use 0 (zero).

listServiceProviderGroupsByType

The **listServiceProviderGroupsByType** operation displays a list of registered service provider groups that have a system level or custom SLA associated, filtered by the SLA type.

The list contains service provider group IDs.

Scope

Domain

Signature

```
listServiceProviderGroupsByType(slaType: String, Offset: int, Size: int)
```

Parameters

slaType
The SLA type.

For custom SLAs, the SLA type was registered in "setupCustomSlaXSDDefinition".

For system level SLAs, see " SLA Types".

Offset
Offset in the list. Starts with 0 (zero)

Size
Size of the list.

For no restrictions on the size of the list, use 0 (zero).

listServiceProviderGroupSlaTypes

The **listServiceProviderGroupSlaTypes** operation displays a list of system and custom SLA types defined for a service provider group.

Scope

Domain

Signature

```
listServiceProviderGroupSlaTypes(applicationGroupIdentifier: String, Offset: int,
Size: int)
```

Parameters

serviceProviderGroupIdentifier
The ID of the service provider group.

Offset
Offset in the list. Starts with 0 (zero)

Size
Size of the list.

For no restrictions on the size of the list, use 0 (zero).

listServiceProviderSlaGroups

The **listServiceProviderSlaGroups** operation is deprecated. Use "listServiceProviderGroupsByType" with slaType **service_provider**.

The **listServiceProviderSlaGroups** operation displays a list of registered service provider groups that has application SLAs associated.

The list contains service provider group IDs.

Scope

Domain

Signature

```
listServiceProviderSlaGroups(Offset: int, Size: int)
```

Parameters

Offset
Offset in the list. Starts with 0 (zero)

Size
Size of the list.

For no restrictions on the size of the list, use 0 (zero).

loadApplicationGroupSla

The **loadApplicationGroupSla** operation is deprecated. Use "loadApplicationGroupSlaByType" with slaType **application**.

The **loadApplicationGroupSla** operation associates an application group SLA with an application group using the contents of the SLA as a parameter.

Scope

Domain

Signature

```
loadApplicationGroupSla(ApplicationGroupIdentifier String, ServiceLevelAgreement:
String)
```

Parameters

ApplicationGroupIdentifier
Unique ID for the application group.

ServiceLevelAgreement
Contents of an application group SLA.

loadApplicationGroupSlaByType

The **loadApplicationGroupSlaByType** operation associates a system or custom SLA with an application group using the contents of the SLA as a parameter.

Scope

Domain or all geo-redundant sites

Signature

```
loadApplicationGroupSlaByType(slaType: String, ApplicationGroupIdentifier :
String, ServiceLevelAgreement: String)
```

Parameters

slaType
The SLA type.

For custom SLAs, the SLA type was registered in "setupCustomSlaXSDDefinition".

For system level SLAs, see " SLA Types".

ApplicationGroupIdentifier
Unique ID for the application group.

ServiceLevelAgreement
Contents of the system level SLA or custom SLA. The content must be formatted according to the XSD associated with the SLA type.

loadApplicationGroupSlaFromUrl

The **loadApplicationGroupSlaFromUrl** operation is deprecated. Use "loadApplicationGroupSlaFromUrlByType" with slaType **application**.

The **loadApplicationGroupSlaFromUrl** operation associates an application group SLA with an application group using the URL to a file that contains the SLA.

Scope

Domain

Signature

```
loadApplicationGroupSlaFromUrl(ApplicationGroupIdentifier : String,
ServiceLevelAgreementURL: String)
```

Parameters

ApplicationGroupIdentifier
Unique ID for the application group.

ServiceLevelAgreement URL
The URL to the SLA.

loadApplicationGroupSlaFromUrlByType

The **loadApplicationGroupSlaFromUrlByType** operation associates a system or custom SLA with an application group using the URL to a file that contains the SLA.

Scope

Domain or all geo-redundant sites

Signature

```
loadApplicationGroupSlaFromUrlByType(slaType: String, ApplicationGroupIdentifier :
String, serviceLevelAgreementURL: String)
```

Parameters

slaType
The SLA type.

For custom SLAs, the SLA type was registered in "setupCustomSlaXSDDefinition".

For system level SLAs, see " SLA Types".

ApplicationGroupIdentifier
Unique ID for the application group.

serviceLevelAgreementURL
The URL to the system level SLA or custom SLA. The content must be formatted according to the XSD associated with the SLA type.

loadGlobalNodeSla

The **loadGlobalNodeSla** operation is deprecated. Use "loadGlobalSlaByType" with slaType **global_node**.

The **loadGlobalNodeSla** operation loads the global node SLA using the contents of the SLA as a parameter.

Scope

Domain or all geo-redundant sites

Signature

```
loadGlobalNodeSla(ServiceLevelAgreement: String)
```

Parameters

ServiceLevelAgreement
Contents of a global node SLA. The content must be formatted according to the XSD associated with the SLA type.

loadGlobalNodeSlaFromUrl

The **loadGlobalNodeSlaFromUrl** operation is deprecated. Use "loadGlobalSlaFromUrlByType" with slaType **global_node**.

The **loadGlobalNodeSlaFromUrl** operation loads a global node SLA using the URL to a file that contains the SLA as a parameter.

Scope

Domain

Signature

```
loadGlobalNodeSlaFromUrl(ServiceLevelAgreement: String)
```

Parameters

ServiceLevelAgreement URL
The URL to the global node SLA. The content must be formatted according to the XSD associated with the SLA type.

loadGlobalSlaByType

The **loadGlobalSlaByType** operation loads a system level or custom global SLA.

Scope

Domain

Signature

```
loadGlobalSlaByType(ServiceLevelAgreement: String)
```

Parameters

slaType
The SLA type.

For custom SLAs, the SLA type was registered in "setupCustomSlaXSDDefinition".

For system level SLAs, see " SLA Types".

ServiceLevelAgreement
Contents of the global SLA. The content must be formatted according to the XSD associated with the SLA type.

loadGlobalSlaFromUrlByType

The **loadGlobalSlaFromUrlByType** operation loads a global SLA using the URL to a file that contains the SLA as a parameter.

Scope

Domain

Signature

```
loadGlobalSlaFromUrlByType(ServiceLevelAgreement: String)
```

Parameters

slaType
The SLA type.

For custom SLAs, the SLA type was registered in "setupCustomSlaXSDDefinition".

For system level SLAs, see " SLA Types".

ServiceLevelAgreement
The URL to the global SLA. The content must be formatted according to the XSD associated with the SLA type.

loadServiceProviderGroupNodeSla

The **loadServiceProviderGroupNodeSla** operation is deprecated. Use "loadServiceProviderGroupSlaByType" with slaType **service_provider_node**.

The **loadServiceProviderGroupNodeSla** operation associates a service provider group node SLA with a service provider group using the contents of the SLA a parameter.

Scope

Domain

Signature

```
loadServiceProviderGroupNodeSla(ServiceProviderGroupIdentifier : String,
ServiceLevelAgreement: String)
```

Parameters

ServiceProviderGroupIdentifier
Unique ID for the service provider group.

ServiceLevelAgreement
The contents of the service provider group node SLA.

loadServiceProviderGroupNodeSlaFromUrl

The **loadServiceProviderGroupSlaFromUrl** operation is deprecated. Use "loadServiceProviderGroupSlaFromUrlByType" with slaType **service_provider_node**.

The **loadServiceProviderGroupSlaFromUrl** operation associates a service provider group node SLA with a service provider group using the URL to a file that contains the SLA. as a parameter.

Scope

Domain

Signature

```
loadServiceProviderGroupNodeSlaFromUrl(ServiceProviderGroupIdentifier : String,
ServiceLevelAgreementUrl: String)
```

Parameters

ServiceProviderGroupIdentifier
Unique ID for the service provider group.

ServiceLevelAgreement URL
The URL to the SLA.

loadServiceProviderGroupSla

The **loadServiceProviderGroupSla** operation is deprecated. Use "loadServiceProviderGroupSlaByType" with slaType **service_provider**.

The **loadServiceProviderGroupSla** operation associates a service provider group SLA with a service provider group using the contents of the SLA as a parameter.

Scope

Domain

Signature

```
loadServiceProviderGroupSla(ServiceProviderGroupIdentifier: String,
ServiceLevelAgreement: String)
```

Parameters

ServiceProviderGroupIdentifier
Unique ID for the service provider group.

ServiceLevelAgreement
Contents of a service provider group SLA. The content must be formatted according to the XSD associated with the SLA type.

loadServiceProviderGroupSlaFromUrl

The **loadServiceProviderGroupSlaFromUrl** operation is deprecated. Use "loadServiceProviderGroupSlaFromUrlByType" with slaType **service_provider**.

The **loadServiceProviderGroupSlaFromUrl** operation associates a service provider group SLA with a service provider group using the URL to a file that contains the SLA as a parameter.

Scope

Domain

Signature

```
loadServiceProviderGroupSlaFromUrl(ServiceProviderGroupIdentifier : String,
ServiceLevelAgreementURL: String)
```

Parameters

ServiceProviderGroupIdentifier
The ID of the service provider group.

ServiceLevelAgreement URL
The URL to the SLA. The content must be formatted according to the XSD associated with the SLA type.

loadServiceProviderGroupSlaFromUrlByType

The **loadServiceProviderGroupSlaFromUrlByType** operation associates a system level or custom SLA with a service provider group using the URL to a file that contains the SLA as a parameter.

Scope

Domain or all geo-redundant sites

Signature

```
loadServiceProviderGroupSlaFromUrlByType(slaType: String,
serviceProviderGroupIdentifier : String, serviceLevelAgreementURL: String)
```

Parameters

slaType
The SLA type.

For custom SLAs, the SLA type was registered in "setupCustomSlaXSDDefinition".

For system level SLAs, see " SLA Types".

serviceProviderGroupIdentifier
The ID for the service provider group.

serviceLevelAgreementURL
The URL to the SLA. The content must be formatted according to the XSD associated with the SLA type.

loadServiceProviderGroupSlaByType

The **loadServiceProviderGroupSlaByType** operation associates a system level or custom SLA with a service provider group using the contents of the SLA as a parameter.

Scope

Domain or all geo-redundant sites

Signature

```
loadServiceProviderGroupSlaByType(slaType: String, serviceProviderGroupIdentifier
: String, ServiceLevelAgreement: String)
```

Parameters

slaType
The SLA type.

For custom SLAs, the SLA type was registered in "setupCustomSlaXSDDefinition".

For system level SLAs, see " SLA Types".

serviceProviderGroupIdentifier
The ID for the service provider group.

ServiceLevelAgreement
Contents of the system level SLA or custom SLA. The content must be formatted according to the XSD associated with the custom SLA type.

loadSubscriberSla

The **loadSubscriberSla** operation is deprecated. Use "loadGlobalSlaByType" with slaType **subscr**.

Loads the Subscriber SLA into the repository.

Scope

Domain

Signature

```
loadSubscriberSla(ServiceLevelAgreement: String)
```

Parameters

ServiceLevelAgreement
The contents of a Subscriber SLA.

loadSubscriberSlaFromUrl

The **loadSubscriberSlaFromUrl** operation is deprecated. Use "loadGlobalSlaFromUrlByType" with slaType **subscr**.

The **loadSubscriberSlaFromUrl** operation loads the Subscriber SLA into the repository using a URL.

Scope

Domain

Signature

```
loadSubscriberSlaFromURL(ServiceLevelAgreementURL: String)
```

Parameters

ServiceLevelAgreementURL
The URL for the Subscriber SLA

retrieveApplicationGroupSla

The **retrieveApplicationGroupSla** operation is deprecated. Use "retrieveApplicationGroupSlaByType" with slaType **application**.

The **retrieveApplicationGroupSla** operation retrieves an application group SLA.

Scope

Domain

Signature

```
retrieveApplicationGroupSla(ApplicationGroupIdentifier : String)
```

Parameters

ApplicationGroupIdentifier
Unique ID for the service provider group.

retrieveApplicationGroupSlaByType

The **retrieveApplicationGroupSlaByType** operation retrieves a system level or custom SLA for an application group.

Scope

Domain

Signature

```
retrieveApplicationGroupSlaByType(slaType : String, ApplicationGroupIdentifier :
String)
```

Parameters

slaType
The SLA type.

For custom SLAs, the SLA type was registered in "setupCustomSlaXSDDefinition".

For system level SLAs, see " SLA Types".

ApplicationGroupIdentifier
Unique ID for the application group.

retrieveCustomSlaXSDDefinition

The **retrieveCustomSlaXSDDefinition** operation retrieves the XSD that defines the custom SLA type.

Scope

Domain

Signature

```
retrieveCustomSlaXSDDefinition(slaType : String)
```

Parameters

slaType
The name of the custom SLA type.

retrieveGlobalNodeSla

The **retrieveGlobalNodeSla** operation is deprecated. Use "retrieveGlobalSlaByType" with slaType **global_node**.

The **retrieveGlobalNodeSla** operation retrieves a global node SLA.

Scope

Domain

Signature

```
retrieveGlobalNodeSla()
```

retrieveGlobalSlaByType

The **retrieveGlobalSlaByType** operation retrieves a system level or custom global SLA.

Scope

Domain

Signature

```
retrieveGlobalSlaByType(slaType : String)
```

Parameters

slaType
For custom SLAs, the SLA type was registered in "setupCustomSlaXSDDefinition".

For system level SLAs, see " SLA Types".

retrieveServiceProviderGroupNodeSla

The **retrieveServiceProviderGroupNodeSla** operation is deprecated. Use "retrieveServiceProviderGroupSlaByType" with slaType **service_provider_node**.

The **retrieveServiceProviderGroupNodeSla** operation retrieves an service provider group node SLA.

Scope

Domain

Signature

```
retrieveServiceProviderGroupNodeSla(ServiceProviderGroupIdentifier : String)
```

Parameters

ServiceProviderGroupIdentifier
ID for the service provider group.

retrieveServiceProviderGroupSla

The **retrieveServiceProviderGroupSla** operation is deprecated. Use "retrieveServiceProviderGroupSlaByType" with slaType **service_provider**.

The **retrieveServiceProviderGroupSla** operation retrieves an service provider group SLA.

Scope

Domain

Signature

```
retrieveServiceProviderGroupSla(ServiceProviderGroupIdentifier : String)
```

Parameters

ServiceProviderGroupIdentifier
ID for the service provider group.

retrieveServiceProviderGroupSlaByType

The **retrieveServiceProviderGroupSlaByType** operation retrieves a system level or custom SLA for a service provider group.

Scope

Domain

Signature

```
retrieveServiceProviderGroupSlaByType(slaType : String,
serviceProviderGroupIdentifier : String)
```

Parameters

slaType
The SLA type.

For custom SLAs, the SLA type was registered in "setupCustomSlaXSDDefinition".

For system level SLAs, see " SLA Types".

serviceProviderGroupIdentifier
The ID for the service provider group.

retrieveSubscriberSla

The **retrieveSubscriberSla** operation is deprecated. Use "retrieveGlobalSlaByType" with slaType **subscr**.

The **retrieveSubscriberSla** operation returns the Subscriber SLA.

Scope

Domain

Signature

```
retrieveSubscriberSla()
```

setupCustomSlaXSDDefinition

The **setupCustomSlaXSDDefinition**operations sets up an XSD document defined as a custom Service Level Agreement type.

Scope

Domain

Signature

```
setupCustomSlaXSDDefinition(slaType : String, xsdDocument : String)
```

Parameters

slaType
The named type of the custom SLA. Used when loading custom SLAs and when fetching the custom SLA in the enforcement logic for the custom SLA.

This parameter is case-sensitive.

xsdDocument
The XSD document that describes the custom type.

setupCustomSlaXSDDefinitionFromUrl

The **setupCustomSlaXSDDefinitionFromUrl** operation sets up an XSD document defined as a custom Service Level Agreement type.

Scope

Domain

Signature

```
setupCustomSlaXSDDefinitionFromUrl(slaType : String, xsdDocumentURL : String)
```

Parameters

slaType
The named type of the custom SLA. Used when loading custom SLAs and when fetching the custom SLA in the enforcement logic for the custom SLA.

This parameter is case-sensitive.

xsdDocumentURL
The URL of the XSD document that describes the custom type.

6

Managing Sessions

This chapter describes how to configure the behavior of and manage ongoing sessions. See the following sections:

- About Sessions
- Reference: ApplicationSessions

See "Reference: ApplicationSessions" for information on where to find the operations in the Administration Console and the name of the MBean.

About Sessions

You can configure whether or not to use sessions between applications and Oracle Communications Services Gatekeeper. Sessions can be used for establishing site affinity when using geographical redundant sites.

The setting of "Attribute: sessionRequired" specifies whether sessions are used.

When using sessions, the application must get a session from the Session Web Service prior to using the other application-facing interfaces. The application must also provide the session ID in all requests to the application-facing interfaces, except for requests to the Session Web Service.

A session has a unique ID. A session, once established, can have two states:

- **ACTIVE**
- **INVALID**

After a session has been established, it is always in the **ACTIVE** state.

Table 6–1 describes the possible states for a session, the state into which it can transition from the current state, and the condition under which such a transition occurs.

Table 6–1 Session States and Conditions for Successful Transition

Current Session State	Valid Transition State	Scenario in Which Transition Occurs
ACTIVE	INVALID	The age of the ACTIVE state of a session is older than a configurable time-interval. See "Attribute: validityTime".

Table 6–1 (Cont.) Session States and Conditions for Successful Transition

Current Session State	Valid Transition State	Scenario in Which Transition Occurs
ACTIVE	Not applicable This is not a state but rather that the session is not current or does not exist.	A management user performs one of the following management operations: ■ destroySession ■ destroyApplicationInstanceSession ■ destroyApplicationSessions ■ destroyServiceProviderSessions
INVALID	Not applicable This is not a state but rather that the session is not current or does not exist.	The age of the INVALID state of a session is older than a configurable time-interval. See "Attribute: expiryTime". A management user performs one of the following management operations: ■ destroyApplicationInstanceSession ■ destroyApplicationSessions ■ destroySession ■ destroyServiceProviderSessions

Reference: ApplicationSessions

All operations are reachable from:

- Oracle Communications Services Gatekeeper Administration Console Managed Object: Container Services > AccountService > ApplicationSessions
- MBean: com.bea.wlcp.wlng.account.management.ApplicationSessionMBean

Following is a list of attributes and operations for ApplicationSessions:

- Attribute: expiryTime
- Attribute: sessionRequired
- Attribute: validityTime
- countApplicationSessions
- destroyServiceProviderSessions
- getApplicationInstanceSession
- destroySession
- destroyApplicationInstanceSession
- destroyApplicationSessions
- getCurrentSession
- listApplicationSessions

Attribute: expiryTime

Scope: Domain

Unit: Minutes

Format: Integer

Specifies the expiry time of sessions.

See "About Sessions" for more information.

Attribute: sessionRequired

Scope: Domain

Unit: Not applicable

Format: Boolean

Specifies if sessions must be established prior to use the application-facing interfaces in Services Gatekeeper. See "About Sessions" for more information.

Attribute: validityTime

Scope: Domain

Unit: Minutes

Format: Integer

Specifies the validity time of sessions.

See "About Sessions" for more information.

countApplicationSessions

The **countApplicationSessions** operation displays the number of established sessions filtered on:

- service provider account
- application account

Scope

Domain

Signature

```
countApplicationSessions(ApplicationIdentifier: String, ServiceProviderIdentifier:
String)
```

Parameters

ApplicationIdentifier
ID of the application account.

Optional. Leave empty to match all.

Service ProviderIdentifier
ID of the service provider account.

Optional. Leave empty to match all.

destroySession

The **destroySession** operation destroys an ongoing session using the ID of the session.

Scope

Domain

Signature

```
destroySession(SessionIdentifier: String)
```

Parameters

SessionIdentifier
ID of the session to destroy.

destroyApplicationInstanceSession

The **destroyApplicationInstanceSession** operation destroys the ongoing session using the ID of the application instance that established the session.

Scope

Domain

Signature

```
destroyApplicationInstanceSession(ApplicationInstanceIdentifier: String)
```

Parameters

ApplicationInstanceIdentifier
ID of the application instance that established the session.

destroyApplicationSessions

The **destroyApplicationSessions** operation destroys all on-going session established by applications associated with a specific service provider /application account combination.

Scope

Domain

Signature

```
destroyApplicationSessions(ApplicationIdentifier: String,
ServiceProviderIdentifier: String)
```

Parameters

ApplicationIdentifier
ID of the application account.

ServiceProviderIdentifier
ID of the service provider account.

destroyServiceProviderSessions

The **destroyServiceProviderSessions** operation destroys all on-going session established by applications associated with a specific service provider account.

Scope

Domain

Signature

```
destroyServiceProviderSessions(ServiceProviderIdentifier: String)
```

Parameters

ServiceProviderIdentifier
ID of the service provider account.

getApplicationInstanceSession

The **getApplicationInstanceSession** operation gets information about a session, active or invalid, using the ID of the application instance that established the session.

The information includes:

- Session ID.
- ID of the application instance that established the session.
- Age, the time since the session was established. Given in minutes.
- Validity indicator for the session:
  - `true` if ACTIVE
  - `false` if INVALID

Scope

Domain

Signature

```
getApplicationInstanceSession(ApplicationInstanceIdentifier: String)
```

Parameters

ApplicationInstanceIdentifier
ID of the application instance.

getCurrentSession

The **getCurrentSession** operation gets information about a session, on-going or invalid, using the ID of the session.

The information includes:

- Session ID.
- ID of the application instance that established the session.
- Age, the time since the session was established. Given in minutes.
- Validity indicator for the session:
  - `true` if valid
  - `false` if invalid

Scope

Domain

Signature

```
getCurrentSession(SessionIdentifier: String)
```

Parameters

SessionIdentifier
ID of the session to get information about.

listApplicationSessions

The **listApplicationSessions** operation displays a list of IDs for all sessions, ongoing or invalid, established by applications. Filters on:

- service provider account
- application account

Scope

Domain

Signature

```
listApplicationSessions(ApplicationIdentifier: String, ServiceProviderIdentifier:
String, Offset: int, Size: int)
```

Parameters

applicationIdentifier
ID of the application account.

Optional. Leave empty to match all.

serviceProviderIdentifier
ID of the service provider account.

Optional. Leave empty to match all.

Offset
Offset in the list. Starts with 0 (zero)

Size
Size of the list.

For no restrictions on the size of the list, use 0 (zero).

7

Defining Service Provider Group and Application Group SLAs

An application's usage policies of Oracle Communications Services Gatekeeper are specified in Service Level Agreement (SLA) XML files. There is an SLA associated with the service provider group and one associated with the application group.

For more information on this administration model, see "About SLAs and Accounts"

For a sample of a complete SLA, see "Sample SLA".

This chapter describes the following topics:

- Structure of a Service Level Agreement
- Structure of a Contract
- Structure of a Composed Service Contract

Structure of a Service Level Agreement

The xsds for the SLAs are in **/sla_schema/** in *Middleware_Home*/**ocsg_5.1/modules/com.bea.wlcp.wlng.account_5.1.0.0.jar**:

- **app_sla_file.xsd** is the application group SLA xsd.
- **sp_sla_file.xsd** is and the service provider group SLA xsd.

An SLA contain can contain three different types of contracts:

- <serviceContract>
- <serviceTypeContract>
- <composedServiceContract>

The SLA must contain at least one `<serviceContract>`. The `<serviceTypeContract>` and `<composedServiceContract>` elements are optional.

The `<serviceTypeContract>` element includes usage restrictions per service type. The `<serviceContract>` element defines restrictions on a more granular level: per interface and method. The `<composedServiceContract>` element defines usage restrictions that apply to multiple communication services.

It is possible to override parts of a `<serviceContract>` based on time-of-day and day-of-week. This is not possible in a `<serviceTypeContract>` or `<composedServiceContract>`.

Example 7–1 shows the service provider level SLA XML file's main structure.

Note: If the SLA XML file has white space before the <?xml . . .> tag, the SLA will not load.

Example 7–1 Service Level Agreement Structure

```
<Sla [serviceProviderGroupID | applicationGroupID] ="<Group>"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="[sp_sla_file.xsd | app_sla_file.xsd]">
  <serviceTypeContract>
    <serviceTypeName></serviceTypeName>
    <startDate></startDate>
    <endDate></endDate>
    <rate></rate>
    <quota></quota>
  </serviceTypeContract>
  <serviceContract>
    <startDate></startDate>
    <endDate></endDate>
    <scs></scs>
    <contract>
    ....
    </contract>
    <overrides>
      <override>
        <startDate></startDate>
        <endDate></endDate>
        <startDow></startDow>
        <endDow></endDow>
        <startTime></startTime>
        <endTime></endTime>
        <contract>
        ...
        </contract>
      </override>
    </overrides>
  </serviceContract>
  <composedServiceContract>
    <composedServiceName></composedServiceName>
      <service>
        <serviceTypeName></serviceTypeName>
        <method>
          <scs></scs>
          <methodName></methodName>
        </method>
      </service>
      <service>
        <serviceTypeName></serviceTypeName>
      </service>
      <startDate></startDate>
      <endDate></endDate>
      <rate>
        <reqLimit></reqLimit>
        <timePeriod></timePeriod>
      </rate>
      <quota>
        <qtaLimit></qtaLimit>
        <days></days>
        <limitExceedOK></limitExceedOK>
      </quota>
```

```
  </composedServiceContract>
</Sla>
```

<Sla>

This element contains service contracts specifying the conditions under which a service provider or an application is allowed to access and use service capabilities.

The **serviceProviderGroupID** attribute specifies the service provider group to which the service provider belongs and for which group the SLA is valid. For SLAs on application level, the **applicationGroupID** attribute is used instead to specify the application group to which the application belongs and for which the SLA is valid.

The **xmlns:xsi** attribute contains processing information and should not be changed.

The **xsi:noNamespaceSchemaLocation** attribute points to the xsd for the service provider group SLA or the application group SLA, depending on which type of SLA it is:

- **app_sla_file.xsd** is the application group SLA xsd
- **sp_sla_file.xsd** is and the service provider group SLA xsd

The SLA for a a service provider group of application group can contain one or more of the following child elements:

- <serviceTypeContract>
- <serviceContract>
- <<composedServiceContract>

<serviceTypeContract>

Parent: <Sla>

This element contains contractual data specifying the conditions under which a service provider or an application is allowed to access and use specific service types in Services Gatekeeper. One `<serviceTypeContract>` element is needed for each service type that a service provider or an application can access.

The data to be defined in the `<serviceTypeContract>` element for each interface is described below.

The `<serviceTypeContract>` element contains the following child elements:

- <serviceTypeName>
- <startDate>
- <endDate>
- <rate>
- <quota>

<serviceTypeName>

Parent: <serviceTypeContract>, <service>

This element specifies the service type to which a `<serviceTypeContract>` or `<service>` applies. It specifies an identifier that defines the service type. Use the Plug-in Manager's **listServiceType** operation to get a list of the service type names. See section "Managing and Configuring the Plug-in Manager" in *Oracle*

Communications Services Gatekeeper System Administrator's Guide for information on how to get a list of service types using the **listServiceType** operation.

For example, the service type for the interfaces for Parlay X 2.1 Short Messaging and RESTful Short Messaging interfaces is **Sms**.

There can be only one `<serviceTypeContract>` for a specified `<serviceTypeName>` element defined in the SLA for a group in the combined set of SLAs (the local SLA and the geo-redundant SLA). For example, if a geo-redundant service provider group SLA defines a `<serviceContract>` for the **CallNotification** service type, this service type not be defined in the local service provider group SLA.

Example 7–2 <serviceTypeContract> Element

```
...
<serviceTypeContract>
  <serviceTypeName>CallNotification</serviceTypeName>
  <startDate>2008-11-01</startDate>
  <endDate>2008-11-30</endDate>
  <rate>
    <reqLimit>25</reqLimit>
    <timePeriod>1000</timePeriod>
  </rate>
  <quota>
    <qtaLimit>250</qtaLimit>
    <days>1</days>
    <limitExceedOK>true</limitExceedOK>
  </quota>
</serviceTypeContract>
<serviceTypeContract>
  <serviceTypeName>Sms</serviceTypeName>
  <startDate>2008-11-01</startDate>
  <endDate>2008-11-30</endDate>
  <rate>
    <reqLimit>35</reqLimit>
    <timePeriod>1000</timePeriod>
  </rate>
  <quota>
    <qtaLimit>350</qtaLimit>
    <days>1</days>
    <limitExceedOK>true</limitExceedOK>
  </quota>
</serviceTypeContract>
...
```

<serviceContract>

Parent: <Sla>

This element contains contractual data specifying under which conditions a service provider or an application is allowed to access and use specific interfaces and methods in Services Gatekeeper. One `<serviceContract>` element is needed for each application-facing interface that a service provider or an application can access.

The `<serviceContract>` element contains the following child elements:

- <startDate>
- <endDate>
- <scs>

- <contract>
- <overrides>

The <contract> element directly following the <scs> element contains the default restrictions.

To override these default restrictions, use a <contract> inside an <override> element.

See "Structure of a Contract" for details about the child elements contained within a <contract> element.

<composedServiceContract>

Parent: <Sla>

This element is defined all the services that comprise a composed service. The <composedServiceContract> element must contain at least one <service> element.

The <composedServiceContract> element contains the following child elements:

- <composedServiceName>
- <service>
- <startDate>
- <endDate>
- <rate>
- <quota>

See "Structure of a Composed Service Contract" for information about composed services and the <service> element.

<startDate>

Parent: <serviceTypeContract>, <serviceContract>, <composedServiceContract>, <override>, <nodeContract>, <globalContract>,

For a <serviceTypeContract>, <serviceContract> or <composedServiceContact>, this element specifies the date the application can start using the service capability.

For an <override>, it specifies the first date that the contract data in the <override> element is valid.

For a <nodeContact>, it specifies the date that the service provider can begin to access the network.

For a <globalContact>, it specifies the date that Services Gatekeeper can begin to access the network on behalf of any service provider.

Use format *YYYY-MM-DD*.

A later start date on the service provider level service contract overrides this date.

<endDate>

Parent: <serviceTypeContract>, <serviceContract>, <composedServiceContract>, <override>, <nodeContract>, <globalContract>

For a <serviceTypeContract>, <serviceContract> or <composedServiceContact>, this element specifies the last date the application can use the service capability.

For an <override>, it specifies the last date that the contract data in the <override> element is valid.

For a <nodeContact>, it specifies the last date that the service provider can access the network node.

For a <globalContact>, it specifies the last date that Services Gatekeeper can access the network on behalf of any service provider.

Use format *YYYY-MM-DD*.

An earlier end date on the service provider level service contract overrides this date.

<scs>

Parent: <serviceContract>, <method>

This element specifies the application-facing interface to which the service contract or composed service contract applies. The content is a Java representation of the Web Service Definition Language (WSDL) that defines the interface.

Use the Plug-in Manager's **getServiceInfo** operation to get a list of the application-facing interfaces. You need to pass the appropriate service type as a parameter to the **getServiceInfo** operation. You can obtain a list of service types with the Plug-in Manager's **listServiceTypes** operation. See section "Managing and Configuring the Plug-in Manager" in *Oracle Communications Services Gatekeeper System Administrator's Guide* for information on using the **getServiceInfo** and **listServiceTypes** operations.

Typically, the Java representation is expressed according to the pattern below:

package_name.*standard indicator*.**plugin.***interface_name_from_WSDL*

For example, the Java representation of the interfaces for Parlay X 2.1 Short Messaging interfaces are:

- com.bea.wlcp.wlng.px21.plugin.SendSmsPlugin
- com.bea.wlcp.wlng.px21.plugin.ReceiveSmsPlugin
- com.bea.wlcp.wlng.px21.plugin.SmsNotificationManagerPlugin

There can be only one <serviceContract> for a specified <scs> defined for a group in its combined set of SLAs (the local SLA and the geo-redundant SLA). For example, if the geo-redundant service provider group SLA defines a <serviceContract> for the com.bea.wlcp.wlng.px21.plugin.SendSmsPlugin interface, this interface can not be defined in the local service provider group SLA.

<overrides>

Parent: <serviceContract>

This element is a container of one or more <override> elements.

SLAs enforced across geo-redundant sites can not contain an <overrides> element.

<override>

Parent: <overrides>

The contract data specified within this element is used to override the default contractual data specified in the <contract> element that directly follows the <scs> element.

When an override occurs, only the restrictions specified in the <override> element are used. All restrictions specified in the default contract are disregarded if not they are not explicitly restated in the <override> element.

Each <override> element can contain the following child elements:

- <startDate>: Specifies the start date that the contract data in the <override> element is valid. If omitted, the start date for the service contract is used.
- <endDate>: Specifies the end date that the contract data in the <override> element is valid. If omitted, the end date for the service contract is used.
- <startDow>: Specifies the starting weekday for which the contract data in the <override> element is valid.
- <endDow>: Specifies the end weekday for which the contract data in the <override> element is valid. Use 1 for Sunday, 2 for Monday and so on. Optional if <startDow> is not specified.
- <startTime>: Specifies the start time for which the contract data in the <override> element is valid.
- <endTime>: Specifies the end time for which the contract data given within the <override> element is valid.
- <contract>: Specifies the contract data that overrides the default contract data specified directly after the <scs> element. See Structure of a Contract for details about the elements contained within a <contract> element.

For an override to be active all of the following conditions must be true:

- Today's date must be the same as or later than the start date.
- Today's date must be earlier than end date (must not be the same date).
- Current time must be between the start time and end time. If the end time is earlier than the start time, the time period spans midnight.
- Current day of week must be between start day of week and end day of week or equal to start day of week or day of week. If end day of week is less than start day of week, the time period spans the weekend.

Since several <override> elements can be defined with different settings for the time periods for which the restrictions applies, make sure the time periods do not overlap. if time periods overlap, there is no guarantee which contract applies.

Example 7–3 <overrides> Element with Two <override> Children

```
<overrides>
  <override>
    <startDate>2008-11-01</startDate>
    <endDate>2008-11-30</endDate>
    <startDow>2</startDow>
    <endDow>2</endDow>
    <startTime>09:00:00</startTime>
    <endTime>10:00:00</endTime>
    <contract>
    ...
    </contract>
  </override>
  <override>
    <startDate>2008-12-01</startDate>
    <endDate>2008-12-30</endDate>
    <startDow>2</startDow>
```

```
    <endDow>2</endDow>
    <contract>
    ...
    </contract>
  </override>
</overrides>
```

<rate>

Parent: <serviceTypeContract>, <methodRestriction>, <composedServiceContract>

This element defines the maximum number of requests to be guaranteed over a short time period, measured in milliseconds.

The <rate> element contains the following child elements:

- <reqLimit>
- <timePeriod>

Overall performance can be adversely affected if the <reqLimit> is frequently exceeded, because requests rejected for exceeding the limit cannot be processed until the next <timePeriod> begins.

To minimize the chances of this type of delay occurring, define a rate with small time frames, in terms of minutes rather than hours. For example, 1000 requests per minute, as defined in Example 7–4

Example 7–4 Rate in Small Time Frames: Minute

```
<rate> <reqLimit>1000</reqLimit> <timePeriod>60000</timePeriod> </rate>
```

is similar in terms of throughput to 60,000 requests per hour as defined in Example 7–5.

Example 7–5 Rate in Large Time Frames: Hour

```
<rate> <reqLimit>60000</reqLimit> <timePeriod>3600000</timePeriod> </rate>
```

But for Example 7–4, the maximum number of possible rejected requests is much smaller for the one-minute <timePeriod> than for the one-hour <timePeriod> in Example 7–5. If you define a rate with a one-hour <timePeriod> and the <reqLimit> is reached in the first 10 minutes, all subsequent requests will be rejected for the rest of the hour.

The enforcement of rates set in SLAs is based on budgets that are set in the Budget service. For information about budgets, see "Managing and Configuring Budgets" in *Oracle Communications Services Gatekeeper System Administrator's Guide*.

<reqLimit>

Parent: <rate>, <nodeRestriction>, <globalRestriction>

This element specifies the maximum number of requests over the time period specified in the corresponding <timePeriod> element.

<timePeriod>

Parent: <rate>, <nodeRestriction>, <globalRestriction>

This element specifies the time period in which the corresponding <reqLimit> applies, in milliseconds.

<quota>

Parent: <serviceTypeContract>, <methodRestriction>, <composedServiceContract>

This element defines the maximum number of requests over a long period of time, measured in days.

The counters associated with the quota are reset at the beginning of each time period.

The <quota> element contains the following child elements:

- <qtaLimit>
- <days>
- <limitExceedOK>

<qtaLimit>

Parent: <quota>

This element defines the maximum number of requests over the time period defined in the <days> element. Only one <qtaLimit> element can be specified per <quota> element.

<days>

Parent: <quota>

This element defines the time period to which the quota limit applies, specified in days. Only integers are valid.

The starting day (day 0) is the same day as the <startdate> element for the service contract. See <startDate>. Only one <days> element can be specified per <quota> element.

<limitExceedOK>

Parent: <quota>

This element specifies the action to take if the quota limit is exceeded. If true, the request will be allowed even if the quota is exceeded. If false, the request will be rejected. An alarm is always emitted if the limit is exceeded

<startDow>

Parent: <override>

Specifies the starting weekday for which the contract data given within the <override> element is valid. Use 1 for Sunday, 2 for Monday and so on. Optional.

<endDow>

Parent: <override>

Specifies the end weekday for which the contract data in the <override> element is valid. Use 1 for Sunday, 2 for Monday and so on. Optional if <startDow> is not specified

<startTime>

Parent: <override>

Specifies the start time for which the contract data given within the <override> element is valid. Use format *hh*:*mm*:*ss*, where *hh* can be 00–24. Optional.

<endTime>

Parent: <override>

Specifies the end weekday for which the contract data in the <override> element is valid. Use 1 for Sunday, 2 for Monday and so on. Optional if <startDow> is not specified.

<enforceAcrossGeoSites>

Parent: <serviceContract>

Deprecated.

This element specifies whether the SLA is enforced across geo-redundant sites.

> **Caution:** This element should not be used. Instead, use the **loadGlobalNodeSlaByType** operation to enforce an SLA across geo-redundant sites. If an SLA with the <enforceAcrossGeoSites> element set to true is loaded under the non-geographically redundant SLA type, exceptions are thrown and the SLA provisioning fails.

The previous behavior is maintained for backwards compatibility as described below.

Specify <enforceAcrossGeoSites>true<enforceAcrossGeoSites> if the SLA should be enforced across geo-redundant site, otherwise false. The default is false.

If set to true, the SLA cannot contain any <override> elements.

The decision of whether to enforce the SLA across geo-redundant sites is taken at load time using the SLA **type** parameter.

Optional element. If not used, the default is false.

Structure of a Contract

Example 7–6 illustrates the structure of the <contract> element in an SLA.

Example 7–6 Contract Structure

```
<contract>
  <guarantee>
    <methodGuarantee>
      <methodNameGuarantee></methodNameGuarantee>
      <reqLimitGuarantee></reqLimitGuarantee>
      <timePeriodGuarantee></timePeriodGuarantee>
    </methodGuarantee>
  </guarantee>
  <methodRestrictions>
    <methodRestriction>
      <methodName></methodName>
      <rate>
        <reqLimit></reqLimit>
```

```
        <timePeriod></timePeriod>
      </rate>
      <quota>
        <qtaLimit></qtaLimit>
        <days></days>
        <limitExceedOK></limitExceedOK>
      </quota>
    </methodRestriction>
  </methodRestrictions>
  <methodAccess>
    <blacklistedMethod>
      <methodName></methodName>
    </blacklistedMethod>
  </methodAccess>
  <requestContext>
    <contextAttribute>
      <attributeName></attributeName>
      <attributeValeu></attributeValue>
    </contextAttribute>
  </requestContext>
  <resultRestrictions>
    <resultRestriction>
      <methodName></methodName>
      <parameterRemovalName></parameterRemovalName>
      <parameterMatch>
        <parameterName></parameterName>
        <parameterValues></parameterValues>
      </parameterMatch>
      <filterMethod></filterMethod>
    </resultRestriction>
  </resultRestrictions>
</contract>
```

<contract>

Parent: <serviceContract>, <override>

This element includes all contract-specific data. There is one `<contract>` element for every `<serviceContract>` element and one for every `<override>` element.

The `<contract>` element can contain the following child elements:

- <guarantee>
- <methodRestrictions>
- <methodAccess>
- <params>
- <requestContext>
- <resultRestrictions>

<guarantee>

Parent: <contract>

This optional element specifies the number of method requests that the service provider or application is guaranteed during a specified time period. The time period is expressed in milliseconds.

Method requests from service providers and applications for which the method is tagged as guaranteed have precedence over requests from service providers and applications not having the method tagged as guaranteed.

Requests are given high priority if the request rate is below either the request rate specified in the service-provider-level SLA or the application-level SLA, whichever has the higher request rate. For example, if the service-provider-level SLA guarantees 30 requests per second and the application-level SLA guarantees 40 requests per second, the application level-SLA applies.

The `<guarantee>` element contains one or more <methodGuarantee> elements.

Each method specified by a `<methodGuarantee>` must have a corresponding `<methodRestriction>` element. The time period defined for a method must be identical in both the `<guarantee>` and `<methodRestriction>` elements.

<methodGuarantee>

Parent: <guarantee>

This element specifies a method for which high priority service is guaranteed. See <guarantee> for details.

The `<methodGuarantee>` element contains the following child elements:

- <methodNameGuarantee>
- <timePeriodGuarantee>
- <reqLimitGuarantee>

The `<methodGuarantee>` element is ignored for mobile-originated traffic.

<methodNameGuarantee>

Parent: <methodGuarantee>

This element specifies the name of the method to have guaranteed precedence.

Use the Plug-in Manager's **getServiceInfo** operation to get a list of valid method names for a service type. You need to pass the appropriate service type as a parameter to the **getServiceInfo** operation.You can obtain a list of service types with the Plug-in Manager's **listServiceTypes** operation. See section "Managing and Configuring the Plug-in Manager" in *Oracle Communications Services Gatekeeper System Administrator's Guide* for information on using the **getServiceInfo** and **listServiceTypes** operations.

<timePeriodGuarantee>

Parent: <methodGuarantee>

This element specifies the length of time for which the guarantee is applied, in milliseconds.

<reqLimitGuarantee>

Parent: <methodGuarantee>

This element specifies the number of requests to be guaranteed over the time period specified in the corresponding `<timeperiodGuarantee>`.

<methodRestrictions>

Parent: <guarantee>

This optional element restricts the number of method requests that an application is allowed over a specified time period. A restriction over a short time period is called a rate. A rate typically spans a few seconds. A restriction over a longer time period is called a quota. A quota typically spans several days.

The <methodRestrictions> element contains one or more <methodRestriction> elements. One <methodRestriction> element is needed for each method for which usage should be restricted. Only one <methodRestrictions> element is allowed per contract.

The <methodRestrictions> element is ignored for mobile-originated traffic.

Example 7–7 shows a sample <methodRestrictions> element with restrictions for quotas and rates. It specifies the usage restrictions for the **SendSms** and **SendSmsLogo** methods. The usage restriction for the rate is 5 requests per second (5 requests divided by 1000 milliseconds). The usage restriction for the quota is 600 requests over a 3 day period.

Example 7–7 Method Restrictions Element

```
<methodRestrictions>
  <methodRestriction>
    <methodName>sendSms</methodName>
    <rate>
      <reqLimit>5</reqLimit>
      <timePeriod>1000</timePeriod>
    </rate>
    <quota>
      <qtaLimit>600</qtaLimit>
      <days>3</days>
      <limitExceedOK>true</limitExceedOK>
    </quota>
  </methodRestriction>
  <methodRestriction>
    <methodName>sendSmsLogo</methodName>
    <rate>
      <reqLimit>5</reqLimit>
      <timePeriod>1000</timePeriod>
    </rate>
    <quota>
      <qtaLimit>600</qtaLimit>
      <days>3</days>
      <limitExceedOK>true</limitExceedOK>
    </quota>
  </methodRestriction>
</methodRestrictions>
```

The most restrictive limit is always enforced, so if a restriction in a service-provider-level SLA is more restrictive than a restriction defined in an application-level SLA, the service-provider-level SLA is enforced.

If the application does not have any usage restrictions within the allowed methods, omit the <methodRestrictions> element.

<methodRestriction>

Parent: <methodRestrictions>

This element specifies restrictions on the usage of a method in the application-facing interface.

The <methodRestriction> element contains the following child elements:

- <methodName>
- <rate>
- <quota>

<methodName>

Parent: <methodRestriction>, <blackListedMethod>, <methodParameters>, <resultRestriction>, <method>

This element specifies the name of the method to restrict.

Use the Plug-in Manager's **getServiceInfo** operation to get a list of valid method names for a service type. You need to pass the appropriate service type as a parameter to the **getServiceInfo** operation.You can obtain a list of service types with the Plug-in Manager's **listServiceTypes** operation. See section "Managing and Configuring the Plug-in Manager" in *Oracle Communications Services Gatekeeper System Administrator's Guide* for information on using the **getServiceInfo** and **listServiceTypes** operations.

<methodAccess>

Parent: <contract>

This optional element is used to block the application from accessing one or more methods in the service capability.

If the application is allowed to access all methods, omit the <methodAccess> element.

The <methodAccess> element contains the <blackListedMethod> child element.

<blackListedMethod>

Parent: <methodAccess>

This element contains one <methodName> child element.

It prohibits use of the method specified in its <methodName> child element.

Use a separate <blacklistedMethod> element for each method to block.

<params>

Parent: <contract>

This optional element is used both to allow and to prohibit certain parameters values provided by applications.

The <params> element contains zero (0) or more <methodParameters> elements.

Example 7–8 shows a <params> element that allows allowed parameter values A, B, and C for the **sendMessage** method. No other values for this parameter are allowed.

Example 7–8 <params> Element

```
<params>
  <methodParameters>
    <methodName>sendMessage</methodName>
    <parameterName>arg0.subject</parameterName>
```

```
    <parameterValues>A B C</parameterValues>
    <acceptValues>true</acceptValues>
  </methodParameters>
</params>
```

<methodParameters>

Parent: <params>

This optional element specifies method parameter values to allow or prohibit.

It contains the following child elements:

- <methodName>
- <parameterName>
- <parameterValues> simple
- <acceptValues>

<parameterName>

Parent: <methodParameters>, <parameterMatch>

This element specifies the name of the parameter whose values are to be allowed or prohibited. The content of this element is the Java representation of the parameter defined in the WSDL. The parameter representation is **arg0.***name_of_parameter_as_defined_in_WSDL*.

You can use the Plug-in Manager's **getServiceInfo** operation to get a list of valid parameter names for the method. You need to pass the appropriate service type as a parameter to the **getServiceInfo** operation.You can obtain a list of service types with the Plug-in Manager's **listServiceTypes** operation. See section "Managing and Configuring the Plug-in Manager" in *Oracle Communications Services Gatekeeper System Administrator's Guide* for information on using the **getServiceInfo** and **listServiceTypes** operations.

Example 7–9 shows an example of the parameters that **getServiceInfo** returns for the **sendMessage** method for the MultimediaMessaging service type.

Example 7–9 Sample Output from getServiceInfo (type: MultimediaMessaging) for SendMessage

```
Interface: com.bea.wlcp.wlng.px21.plugin.SendMessagePlugin
  Method: sendMessage
    Arguments:
     java.net.URI arg0.addresses[]
     java.math.BigDecimal arg0.charging.amount
     java.lang.String arg0.charging.code
     java.lang.String arg0.charging.currency
     java.lang.String arg0.charging.description
     java.lang.String arg0.priority.value
     java.lang.String arg0.receiptRequest.correlator
     java.net.URI arg0.receiptRequest.endpoint
     java.lang.String arg0.receiptRequest.interfaceName
     java.lang.String arg0.senderAddress
     java.lang.String arg0.subject
    Return arguments:
     java.lang.String arg0.result
```

If the parameter is an array (for example, arg0.addresses[]), do not use the brackets in the SLA. For the example, the parameter name in the SLA should be:

`<parameterName>arg0.addresses</parameterName>.`

<parameterValues> simple

Parent: <methodParameters>

This element contains a list of parameter values to allow or prohibit, when contained in a `<methodParameters>` element. Multiple values can be defined, separated by white space.

If the SLA is regulating the Binary SMS Communication Service, UDH values must be specified in decimal, not hexadecimal, format. In the following example, any UDH with the values of 04, 05, 44, or 7F will be blocked:

```
<methodParameters>
        <methodName>sendBinarySms</methodName>
        <parameterName>arg0.binaryMessag[0].udh</parameterName>
        <parameterValues>4 5 68 127</parameterValues>
        <acceptValues>false</acceptValues>
</methodParameters>
```

<parameterValues> complex

Parent: <parameterMatch>

This element contains one or more <parameterValue> elements.

<parameterValue>

Parent:<parameterValues> complex

Each `<parameterValue>` element contains the value of the `<parameterName>` to match if this element is the grandchild of a <parameterMatch> element. The value can be expressed as a regular expression.

<acceptValues>

Parent: <methodParameters>

This element specifies whether the parameter values controlled by the containing `<params>` element are allowed or prohibited.

If `<acceptValues>` is `true`, the parameter values specified in the `<parameterValues>` element are allowed and all other values are prohibited. If `false`, the specified parameter values are prohibited and all other values are allowed

<requestContext>

Parent: <contract>

This optional element is used to send additional parameters, which may not be defined in the interface, to a plug-in. The concept is similar to parameter tunneling, except that the attribute-value pairs are defined in the SLA instead of being passed by the application. For more information see “Parameter Tunneling” in *Oracle Communications Services Gatekeeper Platform Development Studio Developer’s Guide*.

The `<requestContext>` element contains one or more <contextAttribute> elements.

Example 7–10 shows a <requestContext> element that assigns values to two context attributes.

Example 7–10 <requestContext> Example

```
<requestContext>
  <contextAttribute>
    <attributeName>com.bea.wlcp.wlng.plugin.sms.testName1</attributeName>
    <attributeValue>testValue1</attributeValue>
  </contextAttribute>
  <contextAttribute>
    <attributeName>com.bea.wlcp.wlng.plugin.sms.testName2</attributeName>
    <attributeValue>testValue2</attributeValue>
  </contextAttribute>
</requestContext>
```

<contextAttribute>

Parent: <requestContext>

A <contextAttribute> element defines the attribute-value pairs to be sent to the plug-in.

This element contains the following one of each of the following child elements:

- <attributeName>
- <attributeValue>

A plug-in can retrieve the value specified in <attributeValue> using the name specified in <attributeName>. The plug-in must implement the functionality for fetching the value by name. For more information, see "Using Request Context Parameters in SLAs" in *Oracle Communications Services Gatekeeper Platform Development Studio Developer's Guide*.

See the "Tunneled Parameters" sections in the *Communication Service Guide* for descriptions of the context attributes that are applicable to your communication service. Note that not all communication services support tunneled parameters and context attributes.

<attributeName>

Parent: <contextAttribute>

This element contains the name that the plug-in uses to fetch the attribute.

<attributeValue>

Parent: <contextAttribute>

This element contains the value associated with the corresponding <attributeName> in the <contextAttribute> element.

<resultRestrictions>

Parent: <contract>

This optional element is used to filter results returned from application-initiated requests.

This element contains one or more <resultRestriction> elements.

Result Restrictions Example

Example 7–11 illustrates a results filter for the getData() method.

The getData() method returns an array of data, where data is a complex type consisting of the name-value pair:

- dataName
- dataValue

Example 7–11 <resultRestrictions> Element for Blacklisting Array Parameter Values

```
<resultRestrictions>
  <resultRestriction>
    <methodName>getData</methodName>
    <parameterRemovalName>result.data[].dataName</parameterRemovalName>
    <parameterMatch>
      <parameterName>result.data[].dataName</parameterName>
      <parameterValues>
        <parameterValue>ssn<parameterValue>
        <parameterValue>homephone<parameterValue>
      <parameterValues>
    </parameterMatch>
    <filterMethod>BLACK_LIST</filterMethod>
  </resultRestriction>
<resultRestrictions>
```

Assuming the result before the filter is applied is:

- result.data[0].dataName = “cellphone”
- result.data[0].dataValue =”415-555-1234”
- result.data[1].dataName = “ssn”
- result.data[1].dataValue =” 123 45 6789”
- result.data[2].dataName = “homephone”
- result.data[2].dataValue =”415-333-4444”

After applying the result filter, the result is:

- result.data[0].dataName = “cellphone”
- result.data[0].dataValue =”415-555-1234”

because `<filterMethod>` is **BLACK_LIST** and `<parameterValues>` filters out **ssn** and **homephone**. Those data items are removed from the result returned to the application.

If `<filterMethod>` had been **WHITE_LIST**, the result after applying the filter would be:

- result.data[0].dataName = “ssn”
- result.data[0].dataValue =”123 45 6789”
- result.data[0].dataName = “homephone”
- result.data[0].dataValue =”415-333-4444”

Only these parameters would match the filter that filters in the values in `<parameterValues>`.

If `<parameterRemovalName>` had been **result.data** instead of **result.data[]** the filtered result would be null in both cases, since both filters match. The absence of the **[]** means

that all results starting from the **result.data** hierarchy should be removed, given that the filter matches the result.

<resultRestriction>

Parent: <resultRestrictions>

This element defines a result restriction.

The result restriction makes it possible to remove parameters returned from application-initiated requests.

The return parameter to which the restriction applies is expressed as a String representation of a Java class. Arrays are expressed using the [] notation directly following the parameter name. The notation format is:

- **arg0.**result.*name_of_parameter_as_defined_in_WSDL* when the parameter is a single entity.
- **arg0.**result[].*name_of_parameter_as_defined_in_WSDL* when the parameter is an array.

If the return parameter is a complex type, the hierarchy is expressed using dot notation. For example, if an array for the **dateTime** parameter is returned, the return value expressed in XML is:

```
<xsd:complexType name="SmsMessage">
  <xsd:sequence>
   <xsd:element name="message" type="xsd:string"/>
   <xsd:element name="senderAddress" type="xsd:anyURI"/>
   <xsd:element name="smsServiceActivationNumber" type="xsd:anyURI"/>
   <xsd:element name="dateTime" type="xsd:dateTime" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
```

The String representation is:

```
arg0.result[].dateTime.firstDayOfWeek
arg0.result[].dateTime.lenient
arg0.result[].dateTime.minimalDaysInFirstWeek
arg0.result[].dateTime.time.date
arg0.result[].dateTime.time.hours
arg0.result[].dateTime.time.minutes
arg0.result[].dateTime.time.month
arg0.result[].dateTime.time.seconds
arg0.result[].dateTime.time.time
arg0.result[].dateTime.time.year
arg0.result[].dateTime.timeInMillis
arg0.result[].dateTime.timeZone.ID
arg0.result[].dateTime.timeZone.rawOffset
arg0.result[].message
arg0.result[].senderAddress
arg0.result[].smsServiceActivationNumber
```

The `<resultRestriction>` element contains the following child elements:

- <methodName>: method for which to restrict the return parameter.
- <parameterRemovalName>
- <parameterMatch>: optional filter value to match
- <filterMethod>: defines how the filter is applied

<parameterRemovalName>

Parent: <resultRestriction>

The `<parameterRemovalName>` element defines which part of a complex return parameter is affected by the restriction.

It is possible to specify a leaf or a node in the hierarchy of the parameter. When a node is specified, the node and all its siblings are removed.

If the <parameterRemovalName> element is:

```
<parameterRemovalName>arg0.result[].dateTime.timeZone.ID</parameterRemovalName>
```

only the **timeZone.ID** part of the result parameter is removed, because the element specifies a leaf.

If the <parameterRemovalName> element is:

```
<parameterRemovalName>arg0.result[].dateTime.time</parameterRemovalName>
```

all of the **dateTime** siblings are removed from the result parameters, because the element specifies a node. In this case, the following values would be removed:

- arg0.result[].dateTime.time.date
- arg0.result[].dateTime.time.hours
- arg0.result[].dateTime.time.minutes
- arg0.result[].dateTime.time.month
- arg0.result[].dateTime.time.seconds
- arg0.result[].dateTime.time.time
- arg0.result[].dateTime.time.year

If the specified parameter is a Boolean, the parameter is not removed but set to `false`.

For information on how to get a list of valid parameter names for a method with a specific service type, follow the instructions for <methodName>.

<parameterMatch>

Parent:<resultRestriction>

This element is optional.

If `<parameterMatch>` is used, result filtering occurs only if the parameter defined in the `<parameterName>` child element has a value defined in one of its `<parameterValue>` child elements.

The `<parameterMatch>` element contains the following child elements:

- <parameterName>: exactly one
- <parameterValues> complex: exactly one

<filterMethod>

Parent: <resultRestriction>

This element defines how the filter defined in <parameterMatch> is applied.

The value is an enumeration. Valid values are:

- BLACK_LIST

- WHITE_LIST

If `<filterMethod>` is defined as **BLACK_LIST**, all parameters matching `<parameterRemovalName>` are removed if the request matches the filter.

If `<filterMethod>` is defined as **WHITE_LIST**, only the parameters matching `<parameterRemovalName>` are kept if the request matches the filter.

Structure of a Composed Service Contract

A composed service contract is enforced for a composed service.

A composed service is created by combining multiple communication services, all of which must be available and registered in Services Gatekeeper. You can then define and apply enforcements on the composed service, instead of defining identical enforcements separately for each service. A request of any one of the communication services that participate in the composed service is treated as a request of the composed service for the purposes of SLA enforcement.

A single SLA can contain composed service contracts as well as simple service contracts. A composed service contract is located at the same level in the SLA hierarchy as a service contract.

You can configure and test a composed service contract using the Platform Test Environment. See the discussion of composed service-level agreements in the *Oracle Communications Services Gatekeeper Platform Test Environment Guide* for more information.

Composed Service Contracts

You can define a composed service contract in an application group SLA or in a service provider group SLA.

The composed service contract is created when an SLA containing a <composedServiceContract> element is loaded using the loadApplicationGroupSlaByType or loadServiceProviderGroupSlaByType operations.

Scope

A composed service contract includes all the service types of its constituent communication services and all the request rate and quota restrictions that may be applied to those services under an individual service type contract.

A composed service contract can optionally be defined at the method level of granularity so that enforcement is implemented on only specified methods. If individual methods are not specified for a specific service in the composed service contract, enforcement is applied for all methods of the specified service.

Multiple Composed Services

A single SLA can contain multiple composed service contracts.

A single communication service can participate as a component in multiple composed service contracts. For example, one composed service named “Messaging” could be composed of the Short Messaging and Multimedia Messaging communication services, while another composed service named “LocationNotification”, defined in the same SLA, could be composed of the Short Messaging and Terminal Location communication services. In this case, the shared Short Messaging service would be subject to the enforcements of both composed service contracts.

Conflicting Enforcements

It is possible to specify different enforcements that affect a single service within a single SLA. If the enforcements in the multiple contracts are different, the more restrictive enforcement is applied to requests of the shared service. But because separate counters are maintained for each service within a composed service, requests of the non-shared service are not counted towards the limit of the shared service.

For example, imagine an SLA that contains the following contracts, all of which include the Sms service:

1. a service contract for the Sms service that allows 40 requests per second
2. a Messaging composed service contract, composed of the Sms service and the Multimedia Messaging service, that allows 50 requests per second
3. a LocationNotification composed service contract, composed of the Sms service and the Terminal Location service, that allows 60 requests per second

The maximum number of requests to the Sms service will be 40 requests per second. But a request of one of the non-shared services, for example of the Multimedia messaging or Terminal Location service, does not count against this 40-request limit on the Sms service.

Budget Implications

All communication services in a composed service share the same budget restrictions in the composed service contract. Any request of the composed service triggers a decrease in the budget. If a request exceeds the budget restriction, subsequent requests of the composed service are denied. The result is that if one of the services in the composed SLA causes the budget of the composed SLA to be exceeded, requests from the other constituent services are denied.

Example Composed Service SLA

Example 7–12 shows an SLA that contains a simple service contract for the Short Messaging service and a composed service contract for a composed service named "Messaging."

The Messaging composed service includes the **sendSMS** and **sendSmsLogo** operations in the ParlayX2.1 Short Messaging communication service and all of the operations that Services Gatekeeper supports in the ParlayX2.1 Multimedia Messaging communication service.

Example 7–12 Service Level Agreement with Composed Service Contract

```
<Sla xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
serviceProviderGroupID="default_sp_group" >
    <serviceContract>
        <startDate>2010-04-17</startDate>
        <endDate>2011-04-17</endDate>
        <scs>com.bea.wlcp.wlng.px21.plugin.SendSmsPlugin</scs>
    </serviceContract>
    <composedServiceContract>
        <composedServiceName>Messaging</composedServiceName>
        <service>
            <serviceTypeName>Sms</serviceTypeName>
            <method>
                 <scs>com.bea.wlcp.wlng.px21.plugin.SendSmsPlugin</scs>
                 <methodName>sendSMS</methodName>
            </method>
            <method>
```

```
                    <scs>com.bea.wlcp.wlng.px21.plugin.SendSmsPlugin</scs>
                    <methodName>sendSmsLogo</methodName>
                </method>
            </service>
            <service>
                <serviceTypeName>MultiMediaMessage</serviceTypeName>
            </service>
            <startDate>2010-04-17</startDate>
            <endDate>2011-04-17</endDate>
            <rate>
                <reqLimit>50</reqLimit>
                <timePeriod>50</timePeriod>
            </rate>
            <quota>
                <qtaLimit>100</qtaLimit>
                <days>1</days>
                <limitExceedOK>false</limitExceedOK>
            </quota>
        </composedServiceContract>
</Sla>
```

<composedServiceName>

Parent: <composedServiceContract>

This element describes the unique, user-defined name for the composed service.

<service>

Parent: <composedServiceContract>

This element defines a constituent communication service in the composed service. The service must be available and registered in Services Gatekeeper.

There must be at least one `<service>` element in a `<composedServiceContract>` element. Normally there are multiple `<service>` elements.

The `<service>` element contains the following child elements:

- <serviceTypeName>
- <method>

The service is identified by the `<serviceTypeName>` child element of the `<service>` element. The `<serviceTypeName>` element is required.

The `<method>` child element of the `<service>` element is optional.

<method>

Parent: <service>

This optional element, if used, specifies a method of the service to which the composed service contract's enforcements are applied. There can be multiple `<method>` child elements of a single `<service>` element.

If at least one `<method>` element is specified, only the specified method or methods participate in the composed service.

If there are no `<method>` elements contained in a `<service>` element, enforcement is applied to all of the service's methods.

The `<method>` element contains the following child elements:

- <scs>
- <methodName>

8

Defining Global Node and Service Provider Group Node SLAs

This chapter describes node SLAs, which define access to underlying network elements. These are different from the SLAs that define applications' use of Oracle Communications Services Gatekeeper described in "Defining Service Provider Group and Application Group SLAs".

There are two kinds of node SLAs:

- Service provider group node SLAs define a specific service provider's access to the network.
- Global node SLAs define Services Gatekeepers's access to the network regardless of service provider.

Structure of a Node Service Level Agreement

The node SLAs are written in XML. Global node SLAs and service provider group node SLAs are slightly different.

The schema for the global node SLA is in *Middleware_Home***/ocsg_5.1/modules/com.bea.wlcp.wlng.account_5.1.0.0.jar: sla_schema/global_node_sla_file.xsd**.

The schema for the service provider group node SLA is in *Middleware_Home***/ocsg_5.1/modules/com.bea.wlcp.wlng.account_5.1.0.0.jar: sla_schema/sp_node_sla_file.xsd**.

<Sla>

When used to define a node SLA, the `<Sla>` element contains one of the following:

- zero or more <nodeContract> elements for a service provider group node SLA
- zero or more <globalContract> elements for a global node SLA

The `<nodeContract>` and `<globalContract>` elements are mutually exclusive in one SLA instance. You cannot combine `<nodeContract>` and `<globalContract>` elements in the same `<Sla>`.

If the SLA is a service provider group node SLA, the **serviceProviderGroupID** attribute specifies the service provider group for which the SLA is valid.

If the SLA is a global node SLA, omit the **serviceProviderGroupID** attribute.

Service Provider Group Node SLA

The service provider group node SLA consists of an <Sla> element containing zero (0) or more <nodeContract> elements.

The **serviceProviderGroupID** attribute in the <Sla> element specifies the service provider group for which the SLA file is valid.

Example 8–1 shows the structure of a service provider group node SLA.

Example 8–1 Service Provider Group Node SLA Structure

```
<Sla serviceProviderGroupID="spGroup1"

  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="sp_node_sla_file.xsd">

<nodeContract>

    <!--Contract data for network node 1-->

  </nodeContract>

<nodeContract>

    <!--Contract data for network node 2-->

  </nodeContract>

<nodeContract>

    <!--Contract data for network node n-->

  </nodeContract>

</Sla>
```

<nodeContract>

Parent: <Sla>

The <nodeContract> element defines under which conditions Services Gatekeeper can access one or more network nodes on behalf of the service provider group specified in the <Sla> element.

The <nodeContract> element contains the following child elements:

- <startDate>: one (1)

  Specifies the date the service provider can start accessing the network node.

- <endDate> one (1)

  Specifies the last date that the service provider can accessing the network node.

- <nodeID>: one (1)
- <nodeRestrictions>: zero (0) or one (1)

Example 8–2 shows a <nodeContract> that limits access to node A to 10 requests per second (10 requests divided by 1000 milliseconds).

Example 8–2 <nodeContract> Element

```
<nodeContract>
  <startDate>2005-01-01</startDate>
  <endDate>2010-06-01</endDate>
  <nodeID>A</nodeID>
  <nodeRestrictions>
    <nodeRestriction>
      <reqLimit>10</reqLimit>
      <timePeriod>1000</timePeriod>
    </nodeRestriction>
  </nodeRestrictions>
</nodeContract>
```

<nodeID>

Parent: <nodeContract>

The <nodeID> element specifies the network node ID of the node for which a <nodeContract> controls access. The node ID is registered in the Plug-in Manager. A node ID can be assigned to one or more network nodes and the contract can therefore be valid for one or more nodes.

Use the Plug-in Manager's **getPluginNodeId** operation to get the node ID. See "Managing and Configuring the Plug-in Manager" in *Oracle Communications Services Gatekeeper System Administrator's Guide* for information the **getPluginNodeId** operation.

<nodeRestrictions>

Parent: <nodeContract>

The <nodeRestrictions> element contains zero or one <nodeRestriction> elements.

<nodeRestriction>

Parent: <nodeRestrictions>

The <nodeRestriction> element defines the restrictions imposed by the <nodeContract> in a service provider group node SLA.

This element contains the following child elements:

- <reqLimit>: zero (0) or one (1)
- <timePeriod>: zero (0) or one (1)

Global Node SLA

The global node SLA consists of an <Sla> element containing zero (0) or more <globalContract> elements.

For a global node SLA, there should be no **serviceProviderGroupID** attribute in the <Sla> element.

Example 8–3 shows the structure of a global node SLA.

Example 8–3 Global Node SLA Structure

```
<Sla xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:noNamespaceSchemaLocation="global_node_sla_file.xsd">

<globalContract>

    <!--Contract data for network node 1-->

  </globalContract>

<globalContract>

    <!--Contract data for network node 2-->

  </globalContract>

<globalContract>

    <!--Contract data for network node n-->

  </globalContract>

</Sla>
```

<globalContract>

Parent: <Sla>

The <globalContract> element defines the conditions under which Services Gatekeeper can access one or more network nodes, regardless of which service provider sent the request.

The <globalContract> element contains the following child elements

- <startDate>: one (1)

  Specifies the date that Services Gatekeeper can start accessing the network node.

- <endDate>: one (1)

  Specifies the last date that Services Gatekeeper can access the network node.

- <nodeID>: one (1)

- <<globalRestrictions>: zero (0) or one(1)

Example 8–4 shows a <globalContract> that limits Services Gatekeeper's access to node A to 10 requests per second (10 requests divided by 1000 milliseconds). Normal priority requests are passed to the node until the maximum request rate is reached.

Example 8–4 <globalContract> Element

```
<globalContract>
  <startDate>2005-01-01</startDate>
  <endDate>2010-06-01</endDate>
  <nodeID>A</nodeID>
  <globalRestrictions>
    <globalRestriction>
      <reqLimit>1000</reqLimit>
      <timePeriod>10000</timePeriod>
      <guaranteePercentage>50</guaranteePercentage>
    </globalRestriction>
  </globalRestrictions>
</globalContract>
```

<globalRestrictions>

Parent: <globalContract>

The `<globalRestrictions>` element contains zero or one <globalRestriction> elements.

<globalRestriction>

Parent: <globalRestrictions>

The `<globalRestriction>` element defines the restrictions imposed by the `<globalContract>` in a global node SLA.

This element contains the following child elements:

- <reqLimit>: zero (0) or one (1)
- <timePeriod>: zero (0) or one (1)
- <guaranteePercentage>: zero (0) or one (1)

<guaranteePercentage>

Parent: <globalRestriction>

The `<guaranteePercentage>` element is used in a `<globalRestriction>` element in a global node contract.

It specifies the relative priority between requests marked as guaranteed (high priority) and normal priority requests. The integer value represents a percentage.

If set to 0, guaranteed requests and normal priority requests are treated equally, no matter whether they are guaranteed or normal priority. The value of 0 makes the priority of the requests irrelevant.

If set to 100, only guaranteed requests are passed on to the network node. No normal priority requests are allowed.

If set to 50, normal priority requests are passed on to the network node up to the point where the maximum request rate is reached. After that point, requests with normal priority are rejected.

Other values can be used to fine-tune how likely it is that requests of different priority are allowed. A higher value makes is more likely that normal priority requests will be rejected as traffic rate increases. A lower value makes it less likely that normal priority requests will be rejected.

The default value is 50.

A

Sample SLA

The following is a complete sample SLA for the Parlay X 2.1 Short Messaging Communication service. It contains a <serviceContract>, a <serviceTypeContract>, and a <composedServiceContract>.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Sla applicationGroupID="default_app_group"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="app_sla_file.xsd">
    <serviceContract>
        <startDate>2005-07-22</startDate>
        <endDate>9999-12-31</endDate>
        <scs>com.bea.wlcp.wlng.px21.plugin.SendSmsPlugin</scs>
        <contract>
            <guarantee>
                <methodGuarantee>
                    <methodNameGuarantee>sendSmsLogo</methodNameGuarantee>
                    <reqLimitGuarantee>10000</reqLimitGuarantee>
                    <timePeriodGuarantee>80000</timePeriodGuarantee>
                </methodGuarantee>
            </guarantee>
            <methodRestrictions>
                <methodRestriction>
                    <methodName>sendSms</methodName>
                    <rate>
                        <reqLimit>9000</reqLimit>
                        <timePeriod>1000</timePeriod>
                    </rate>
                    <quota>
                        <qtaLimit>900000</qtaLimit>
                        <days>1</days>
                        <limitExceedOK>false</limitExceedOK>
                    </quota>
                </methodRestriction>
            </methodRestrictions>
            <params>
                <methodParameters>
                    <methodName>sendSms</methodName>
                    <parameterName>arg0.message</parameterName>
                    <parameterValues>foo</parameterValues>
                    <acceptValues>false</acceptValues>
                </methodParameters>
            </params>
            <methodAccess>
                <blacklistedMethod>
                    <methodName>sendSmsLogo</methodName>
                </blacklistedMethod>
```

```
                </methodAccess>
                <requestContext>
                    <contextAttribute>
                        <attributeName>key1</attributeName>
                        <attributeValue>value1</attributeValue>
                    </contextAttribute>
                </requestContext>
                <resultRestrictions>
                    <resultRestriction>
                       <methodName>getSmsDeliveryStatus</methodName>
<parameterRemovalName>arg0.requestIdentifier</parameterRemovalName>
                        <parameterMatch>
                            <parameterName>arg0.requestIdentifier</parameterName>
                            <parameterValues>
                                <parameterValue>demo</parameterValue>
                            </parameterValues>
                        </parameterMatch>
                        <filterMethod>BLACK_LIST</filterMethod>
                    </resultRestriction>
                </resultRestrictions>
            </contract>
            <overrides>
                <override>
                    <startDate>2010-11-30</startDate>
                    <endDate>2012-11-30</endDate>
                    <startDow>2</startDow>
                    <endDow>6</endDow>
                    <contract>
                        <guarantee>
                            <methodGuarantee>
                                <methodNameGuarantee>sendSms</methodNameGuarantee>
                                <reqLimitGuarantee>1000</reqLimitGuarantee>
                                <timePeriodGuarantee>40000</timePeriodGuarantee>
                            </methodGuarantee>
                        </guarantee>
                        <methodRestrictions>
                            <methodRestriction>
                                <methodName>sendSms</methodName>
                                <rate>
                                    <reqLimit>500</reqLimit>
                                    <timePeriod>1000</timePeriod>
                                </rate>
                                <quota>
                                    <qtaLimit>10000</qtaLimit>
                                    <days>1</days>
                                    <limitExceedOK>false</limitExceedOK>
                                </quota>
                            </methodRestriction>
                        </methodRestrictions>
                        <params>
                            <methodParameters>
                                <methodName>sendSms</methodName>
                                <parameterName>arg0.message</parameterName>
                                <parameterValues>foo2</parameterValues>
                                <acceptValues>false</acceptValues>
                            </methodParameters>
                        </params>
                        <methodAccess>
                            <blacklistedMethod>
                                <methodName>sendSmsLogo</methodName>
```

```
                        </blacklistedMethod>
                        <blacklistedMethod>
                            <methodName>sendSmsRingtone</methodName>
                        </blacklistedMethod>
                    </methodAccess>
                    <requestContext>
                        <contextAttribute>
                            <attributeName>key2</attributeName>
                            <attributeValue>value2</attributeValue>
                        </contextAttribute>
                    </requestContext>
                    <resultRestrictions>
                        <resultRestriction>
                   <methodName>getSmsDeliveryStatus</methodName>
<parameterRemovalName>arg0.requestIdentifier</parameterRemovalName>
                            <parameterMatch>
                                <parameterName/>
                                <parameterValues>
                                    <parameterValue>22</parameterValue>
                                    <parameterValue>33</parameterValue>
                                </parameterValues>
                            </parameterMatch>
                            <filterMethod>WHITE_LIST</filterMethod>
                        </resultRestriction>
                    </resultRestrictions>
                </contract>
            </override>
        </overrides>
    </serviceContract>
    <serviceTypeContract>
        <serviceTypeName>Sms</serviceTypeName>
        <startDate>2010-11-30</startDate>
        <endDate>2010-11-30</endDate>
        <rate>
            <reqLimit>1000</reqLimit>
            <timePeriod>1000</timePeriod>
        </rate>
        <quota>
            <qtaLimit>90000</qtaLimit>
            <days>1</days>
            <limitExceedOK>false</limitExceedOK>
        </quota>
    </serviceTypeContract>
    <composedServiceContract>
        <composedServiceName>Messaging</composedServiceName>
        <service>
            <serviceTypeName>Sms</serviceTypeName>
            <method>
                <scs>com.bea.wlcp.wlng.px21.plugin.SendSmsPlugin</scs>
                <methodName>sendSMS</methodName>
            </method>
         </service>
        <service>
            <serviceTypeName>MultiMediaMessage</serviceTypeName>
        </service>
        <startDate>2010-04-17</startDate>
        <endDate>2011-04-17</endDate>
        <rate>
            <reqLimit>50</reqLimit>
            <timePeriod>50</timePeriod>
```

```
        </rate>
        <quota>
            <qtaLimit>100</qtaLimit>
            <days>1</days>
            <limitExceedOK>false</limitExceedOK>
        </quota>
    </composedServiceContract>
</Sla>
```