Oracle® Communication Services Gatekeeper

Managing Accounts and SLAs

Release 4.0

June 2008

ORACLE

Oracle Communication Services Gatekeeper Managing Accounts and SLAs, Release 4.0

Contents:

Managing Groups

Managing SLAs

Managing a Subscriber SLA

Managing Sessions

Defining Service Provider Group and Application Group SLAs

Defining Global Node and Service Provider Group Node SLAs

CHAPTER 1

Introduction and Document Roadmap

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

Document Scope and Audience

This document describes how to perform service provider and application provisioning for Network Gatekeeper, including:

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

The document will be of use to telecom operators, especially in relation to service provider and application provisioning. Managers, support engineers, and sales and marketing people will also find information of value here.

Guide to this Document

The document contains the following chapters:

- Chapter 1, “Introduction and Document Roadmap”: This chapter
- Chapter 2, “Creating and Maintaining Service Provider and Application Accounts”: An overview of the Network Gatekeeper service provider and application administration model, including a high level workflow
- Chapter 3, “Managing Application Instances”: Instructions for managing application instances
- Chapter 4, “Managing Service Provider and Application Accounts”: Instructions for managing service provider and application accounts
- Chapter 5, “Managing Groups”: Instructions for managing service provider and application groups
- Chapter 8, “Managing Sessions”: Step-instructions for managing application account sessions.
- Chapter 9, “Defining Service Provider Group and Application Group SLAs”: Description of Service provider group and application group SLAs
- Chapter 10, “Defining Global Node and Service Provider Group Node SLAs”: Description of global node and service provider group node LAs

CHAPTER 2

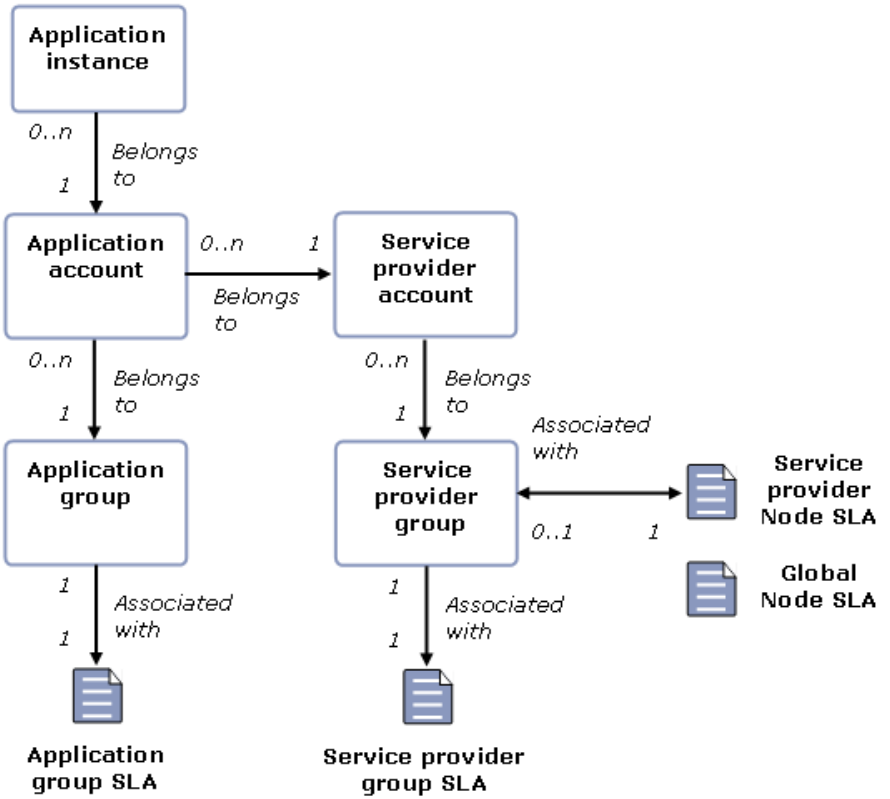
Creating and Maintaining Service Provider and Application Accounts

An essential part of managing WebLogic Network Gatekeeper is dealing with service provider partners. Network Gatekeeper provides a partner administration model to help operators handle the needs and demands of their partners in a flexible and powerful way:

- Each application that uses the application-facing interfaces in Network Gatekeeper must establish an account. The account is uniquely identified by an *application instance*.
- An *application instance* 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 a service level agreement on the application-level.
- *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*.
- A *global node SLA* is not associated with any special group or account, but is valid for all accounts and groups.

See Figure 2-1 for an illustration of the relationship among the different account types.

Figure 2-1 Service provider and application model with SLAs



The separation of accounts and groups provides a flexible way of defining service level agreements. Groups allow the Network Gatekeeper administrator to offer tiers of service at both the service provider (via service provider group SLAs) and application (via application group SLAs) level.

For example, a Network Gatekeeper administrator might use SLAs to create three different categories of service providers:

- Bronze
- Silver

- Gold

Each of these would have different privileges when it comes to maximum throughput and QoS parameters. When a new service provider is provisioned, the service provider is associated with an 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 with a different QoS parameter defined in the service provider level SLA.

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

The service provider and application account and group registrations are performed either through the WebLogic Network Gatekeeper Administration Console or through external management systems integrated with WebLogic Network Gatekeeper using the Network Gatekeeper Partner Relationship Management Interfaces or via MBeans. This guide describes many of the tasks associated with establishing and maintaining these accounts and groups using the Administration Console.

Note: Service provider accounts, application accounts, and application instances have states that can be changed using OAM, so they can be temporarily be taken out of service. See States of Accounts.

Typical workflow

1. 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.
2. 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.
3. Create service provider and application groups that correspond to these SLAs, see Managing Groups.
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.

6. Create a service provider account and associate it with the appropriate service provider group. See Managing Service Provider and Application Accounts.
7. Create an application account and associate it with the appropriate application group. See Managing Service Provider and Application Accounts.
8. Create an application instance, see Managing Application Instances.
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.

Note: It is also possible to create unique SLAs and groups for an individual application. Accounts can be associated with other groups in runtime, so the class of service can be adjusted over time.

CHAPTER 3

Managing Application Instances

This section describes how application instances are managed and provisioned:

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

See Reference: ApplicationInstances, for information 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.

Summary of Tasks related to Application Instances

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

States of Application Instances

Application instances have states. Two states are possible:

- ACTIVATED
- DEACTIVATED

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

In state DEACTIVATED, the application using this application instance ID is not allowed to send traffic through Network Gatekeeper. The account is still valid and the state can be transitional by the Network Gatekeeper Administrator using:

Operation: setApplicationInstanceState

Web Services Security

When Web Services Security is being used as authentication for the application instance, its credentials must be set up, as in the mapping rules in Table 3-1.

Table 3-1 Credential mapping

ApplicationInstance	UserNameToken	X.509	SAML
ApplicationInstanceName	userName	CN	CN
Password	password	n/a	n/a

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 http://edocs.bea.com/wls/docs100/admin_ref/utils.html

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 to use your own key. It should be 24 Bytes.

Configure the encryption key, starting in the Administration Console:

1. Select **Security realms**–>**myrealm**
2. Select **Providers** tab–>**Authentication** tab.
3. Select **WLNG Application Authenticator**
4. Select **Configuration** tab –> **Provider specific** tab.

   In **Encryption Key** field enter your key.

   In **Please type again To confirm** field enter your key again for verification.

Reference: ApplicationInstances

All operations are reachable from:

- Network Gatekeeper Administration Console Managed Object: Container Services–>AccountService–>ApplicationInstances

- MBean: com.bea.wlcp.wlng.account.management.ApplicationInstanceMBean

Below is a list of operations for management.

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

Operation: addApplicationInstance

Scope: Domain

Adds an application instance. Always connected to a service provider/application account combination.

Signature:

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

Table 3-2 addApplicationInstance

addApplicationInstance	
Parameter	**Description**
ApplicationInstanceName	ID for the new application instance. Used by an application to authenticate. Must be unique.
Password	Password for the application instance.

Table 3-2 addApplicationInstance

addApplicationInstance	
Parameter	**Description**
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.

Operation: countApplicationInstances

Scope: Domain

Displays the number of registered application instances for a service provider/application account/application instance combinations. Can be filtered by state.

Signature:

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

Table 3-3 countApplicationInstances

countApplicationInstances	
Parameter	**Description**
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: • ACTIVATED • DEACTIVATED Optional. Leave empty to match all.

Operation: getApplicationInstance

Scope: Domain

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.

Signature:

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

Table 3-4 getApplicationInstance

getApplicationInstance	
Parameter	**Description**
ApplicationInstanceName	ID of the application instance to get information about.

Operation: listApplicationInstances

Scope: Domain

Displays a list of registered application instances for a specific service provider/application account/application instance state combination.

The list contains application instance IDs.

Signature:

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

Table 3-5 listApplicationInstances

listApplicationInstances	
Parameter	**Description**
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).

Operation: removeApplicationInstance

Scope: Domain

Removes a registered application instance.

The account must be in state DEACTIVATED.

Signature:

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

Table 3-6 removeApplicationInstance

removeApplicationInstance	
Parameter	**Description**
ApplicationInstanceName	ID of the application instance to remove.

Operation: setApplicationInstancePassword

Scope: Domain

Defines a new password for an application instance.

Signature:

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

Table 3-7 setApplicationInstancePassword

setApplicationInstancePassword	
Parameter	**Description**
ApplicationInstanceName	ID of the application instance.
Password	New password.

Operation: setApplicationInstanceProperties

Scope: Domain

Specifies properties for an application instance: see Account Properties.

Note: Only applicable when accessing the MBean directly.

Signature:

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

Table 3-8 setApplicationInstanceProperties

setApplicationInstanceProperties	
Parameter	**Description**
ApplicationInstanceName	ID of the application instance to set properties for.
Properties	See Account Properties.

Operation: setApplicationInstanceReference

Scope: Domain

Specifies a reference for an application instance: see Account References.

Signature:

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

Table 3-9 setApplicationInstanceReference

setApplicationInstanceReference	
Parameter	**Description**
ApplicationInstanceName	ID of the application instance to set the reference for.
Reference	See Account References.

Operation: setApplicationInstanceState

Scope: Domain

Performs a state transition for an application instance: see States of Application Instances.

Signature:

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

Table 3-10 setApplicationInstanceState

setApplicationInstanceState	
Parameter	**Description**
ApplicationInstanceName	ID of the application instance to change state for.
State	See States of Application Instances. One of: • ACTIVATED • DEACTIVATED

CHAPTER 4

Managing Service Provider and Application Accounts

This section describes how service provider and application accounts are managed and provisioned:

- 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.

Prior to registering application accounts, service provider accounts must be created (see Service Provider Accounts) and application groups must be created (see Application Groups).

Prior to registering service provider accounts, service provider groups must be created: see Service Provider Groups.

Summary of tasks Related to Accounts

Application Accounts

To...	Use...
Get information about the number of application accounts	Operation: countApplicationAccounts
Add, remove and get information about an application account	Operation: addApplicationAccount Operation: removeApplicationAccount Operation: getApplicationAccount
List registered application accounts	Operation: listApplicationAccounts
Define additional properties for an application account	Operation: setApplicationAccountGroup Operation: setApplicationAccountProperties Operation: setApplicationAccountReference Operation: setApplicationAccountState

Service Provider Accounts

To...	Use...
Get information about the number of service provider accounts	Operation: countServiceProviderAccounts
Add, remove and get information about a service provider account	Operation: addServiceProviderAccount Operation: removeServiceProviderAccount Operation: getServiceProviderAccount

To...	Use...
List registered service provider accounts	Operation: listServiceProviderAccounts
Define additional properties for a service provider account	Operation: setServiceProviderAccountGroup Operation: setServiceProviderAccountProperties Operation: setServiceProviderAccountReference Operation: setServiceProviderAccountState

States of Accounts

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

Accounts can be in either one of the following states:

Account state	An application belonging to the account...
ACTIVATED	is allowed to send requests to the application-facing interfaces exposed by Network Gatekeeper.
DEACTIVATED	is not allowed to send requests to the application-facing interfaces exposed by Network Gatekeeper.

State transition is provided using:

- Operation: setApplicationAccountState
- Operation: setServiceProviderAccountState

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

- Operation: countApplicationAccounts
- Operation: countServiceProviderAccounts
- Operation: listApplicationAccounts
- Operation: listServiceProviderAccounts

When an account is created using Operation: addApplicationAccount or Operation: 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:

- Operation: getApplicationInstance
- Operation: getApplicationAccount
- Operation: getApplicationGroup
- Operation: getServiceProviderAccount
- Operation: 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 properties are defined as parameters in the following operations:

- Operation: addApplicationInstance
- Operation: setApplicationInstanceReference
- Operation: setApplicationInstanceProperties
- Operation: addApplicationAccount
- Operation: setApplicationAccountReference
- Operation: addServiceProviderAccount
- Operation: setApplicationAccountReference

They are retrieved as part of the result of:

- Operation: getApplicationInstance
- Operation: getApplicationAccount

- Operation: getServiceProviderAccount

Reference: Attributes and Operations for ApplicationAccounts

Managed object: Container Services–>AccountService–>ApplicationAccounts

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

Below is a list of operations for configuration and maintenance:

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

Operation: addApplicationAccount

Scope: Domain

Adds an application account.

Signature:

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

Table 4-1 addApplicationAccount

addApplicationAccount	
Parameter	**Description**
Application Identifier	ID of the application account to add.
Service Provider Identifier	ID of the service provider account the application account shall belong to.
Application Group Identifier	ID of the application group the application account shall belong to.
Reference	See Account References.

Operation: addServiceProviderAccount

Scope: Domain

Adds a service provider account.

Signature:

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

Table 4-2 addServiceProviderAccount

addServiceProviderAccount	
Parameter	**Description**
Service Provider Identifier	ID of the service provider account to add.
Service Provider Group Identifier	ID of the service provider group the service provider account shall belong to.
Reference	See Account References.

Operation: countApplicationAccounts

Scope: Domain

Displays the number of registered application accounts. Can be filtered on any combination of:

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

Signature:

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

Table 4-3 countApplicationAccounts

countApplicationAccounts	
Parameter	**Description**
ServiceProviderIdentifier	ID of the service provider account. Optional. Leave empty to match all.

Table 4-3 countApplicationAccounts

countApplicationAccounts	
Parameter	**Description**
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.

Operation: countServiceProviderAccounts

Scope: Domain

Displays the number of registered service provider accounts. Can be filtered on any combination of:

- service provider group
- service provider account state.

Signature:

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

Table 4-4 countServiceProviderAccounts

countServiceProviderAccounts	
Parameter	**Description**
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.

Operation: getApplicationAccount

Scope: Domain

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.

Signature:

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

Table 4-5 getApplicationAccount

getApplicationAccount	
Parameter	**Description**
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.

Operation: getServiceProviderAccount

Scope: Domain

Gets information about a service provider account.

The information includes:

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

Signature:

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

Table 4-6 getServiceProviderAccount

getServiceProviderAccount	
Parameter	**Description**
Service Provider Identifier	ID of the service provider account.

Operation: listApplicationAccounts

Scope: Domain

Displays a list of application accounts. Can be filtered on any combination of:

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

The resultant list contains application account IDs.

Signature:

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

Table 4-7 listApplicationAccounts

listApplicationAccounts	
Parameter	**Description**
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).

Operation: listServiceProviderAccounts

Scope: Domain

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.

Signature:

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

Table 4-8 listServiceProviderAccounts

listServiceProviderAccounts	
Parameter	**Description**
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).

Operation: removeApplicationAccount

Scope: Domain

Removes an application account.

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

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

Signature:

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

Table 4-9 removeApplicationAccount

removeApplicationAccount	
Parameter	**Description**
Application Identifier	ID of the application account to remove.
Service Provider Identifier	ID of the service provider account the application account belongs to.

Operation: removeServiceProviderAccount

Scope: Domain

Removes a service provider account.

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

Signature:

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

Table 4-10 removeServiceProviderAccount

removeServiceProviderAccount	
Parameter	**Description**
Service Provider Identifier	ID of the service provider account to remove.

Operation: setApplicationAccountGroup

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.

Signature:

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

Table 4-11 setApplicationAccountGroup

setApplicationAccountGroup	
Parameter	**Description**
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.

Operation: setApplicationAccountProperties

Specifies properties for an application account/service provider account combination: see Account Properties.

Note: Only applicable when accessing the MBean directly.

Signature:

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

Table 4-12 setApplicationAccountProperties

setApplicationAccountProperties	
Parameter	**Description**
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.

Operation: setApplicationAccountReference

Specifies a reference for an application account/service provide account combinations: see Account References.

Signature:

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

Table 4-13 setApplicationAccountProperties

setApplicationAccountProperties	
Parameter	**Description**
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.

Operation: setApplicationAccountState

Performs a state transition for an application account/service provide account combination: see States of Accounts.

Signature:

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

Table 4-14 setApplicationAccountState

setApplicationAccountState	
Parameter	**Description**
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	See States of Accounts. One of: • ACTIVATED • DEACTIVATED

Operation: setServiceProviderAccountGroup

Specifies to which service provider group a certain service provider account belongs. Can be used as well to change the group assignment.

Signature:

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

Table 4-15 setServiceProviderAccountGroup

setServiceProviderAccountGroup	
Parameter	**Description**
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.

Operation: setServiceProviderAccountProperties

Specifies properties for a service provider account: see Account Properties.

Note: Only applicable when accessing the MBean directly.

Signature:

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

Table 4-16 setServiceProviderAccountProperties

setServiceProviderAccountProperties	
Parameter	**Description**
Service Provider Identifier	ID of the service provider account whose properties are being set.
Properties	See Account Properties.

Operation: setServiceProviderAccountReference

Specifies a reference for a service provider account: see Account References.

Signature:

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

Table 4-17 setServiceProviderAccountReference

setServiceProviderAccountReference	
Parameter	**Description**
Service Provider Identifier	ID of the service provider account whose reference is being set.
Reference	See Account References.

Operation: setServiceProviderAccountState

Performs a state transition for a service provider account: see States of Accounts.

Signature:

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

Table 4-18 setServiceProviderAccountState

setServiceProviderAccountState	
Parameter	**Description**
Service Provider Identifier	ID of the service provider account whose state is being changed.
State	See States of Accounts. One of: • ACTIVATED • DEACTIVATED

CHAPTER 5

Managing Groups

This section describes how service provider and application groups are managed and provisioned:

- 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.

Prior to registering application and service provider groups, SLAs must be created, see Defining Service Provider Group and Application Group SLAs.

Summary of Tasks Related to Groups

Application Groups

To...	Use...
Get information about the number of application groups	Operation: countApplicationGroups
Add, remove and get information about an application group.	Operation: addApplicationGroup Operation: removeApplicationGroup Operation: getApplicationGroup

To...	Use...
List registered application groups.	Operation: listApplicationGroups
Define additional properties for an application group	Operation: setApplicationGroupProperties

Service Provider Groups

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

Reference: Attributes and Operations for ApplicationGroups

Managed object: Container Services->AccountService->ApplicationGroups

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

Below is a list of operations for configuration and maintenance

- Operation: addApplicationGroup
- Operation: addServiceProviderGroup
- Operation: countApplicationGroups
- Operation: countServiceProviderGroups
- Operation: getApplicationGroup

Operation: addApplicationGroup

Scope: Domain

Adds a new application group.

Signature:

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

Table 5-1 addApplicationGroup

addApplicationGroup	
Parameter	**Description**
ApplicationGroupIdentifier	Unique ID for the application group.

Operation: addServiceProviderGroup

Scope: Domain

Adds a new service provider group.

Signature:

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

Table 5-2 addServiceProviderGroup

addServiceProviderGroup	
Parameter	**Description**
ServiceProviderGroupIdentifier	Unique ID for the service provider group.

Operation: countApplicationGroups

Scope: Domain

Displays the number of application groups.

Signature:

```
countApplicationGroups()
```

Table 5-3 countApplicationGroups

countApplicationGroups	
Parameter	**Description**
-	-

Operation: countServiceProviderGroups

Scope: Domain

Displays the number of service provider groups.

Signature:

```
countServiceProviderGroups()
```

Table 5-4 countServiceProviderGroups

countServiceProviderGroups	
Parameter	**Description**
-	-

Operation: getApplicationGroup

Scope: Domain

Displays information about an application group, including:

- Application group description
- Properties: see Account Properties

Signature:

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

Table 5-5 getApplicationGroup

getApplicationGroup	
Parameter	**Description**
ApplicationGroupIdentifier	ID of the application group whose description is desired.

Operation: getServiceProviderGroup

Scope: Domain

Displays information about a service provider group.

The information includes:

- Service provider group description.
- Properties: see Account Properties

Signature:

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

Table 5-6 getServiceProviderGroup

getApplicationGroup	
Parameter	**Description**
ServiceProviderGroupIdentifier	ID of the service provider group to get information for.

Operation: listApplicationGroups

Scope: Domain

Displays a list of application groups.

The resultant list contains application group IDs.

Signature:

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

Table 5-7 listApplicationGroups

listApplicationGroups	
Parameter	**Description**
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).

Operation: listServiceProviderGroups

Scope: Domain

Displays a list of service provider groups.

The resultant list contains service provider group IDs.

Signature:

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

Table 5-8 listServiceProviderGroups

listServiceProviderGroups	
Parameter	**Description**
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).

Operation: removeApplicationGroup

Scope: Domain

Removes an existing application group. The group must not have any applications associated with it.

Signature:

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

Table 5-9 removeApplicationGroup

removeApplicationGroup	
Parameter	**Description**
ApplicationGroupIdentifier	ID for the application group.

Operation: removeServiceProviderGroup

Scope: Domain

Removes an existing service provider group. The group must not have any service providers associated with it.

Signature:

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

Table 5-10 removeServiceProviderGroup

removeServiceProviderGroup	
Parameter	**Description**
ServiceProviderGroupIdentifier	ID for the service provider group.

Operation: setApplicationGroupProperties

Specifies properties for an application group: see Account Properties.

Note: Only applicable when accessing the MBean directly.

Signature:

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

Table 5-11 setApplicationAccountProperties

setApplicationAccountProperties	
Parameter	**Description**
ApplicationGroupIdentifier	ID of the application group whose properties are to be set.
Properties	See Account Properties.

Operation: setServiceProviderGroupProperties

Specifies properties for a service provider group: see Account Properties.

Note: Only applicable when accessing the MBean directly.

Signature:

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

Table 5-12 setServiceProviderGroupProperties

setServiceProviderGroupProperties	
Parameter	**Description**
ServiceProviderGroupIdentifier	ID of the service provider group whose properties are to be set.
Properties	See Account Properties.

CHAPTER 6

Managing SLAs

This section describes how the four main types of SLAs are managed and provisioned:

- Summary of Tasks Related to SLAs
  - Service Provider and Application Group SLAs
  - Node SLAs

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

There are two main types of SLAs:

- Service provider group and application group level SLAs. These define how applications and service providers can use Network Gatekeeper: see Service Provider and Application Group SLAs.
- Service provider node and global node SLAs. These define how Network Gatekeeper is allowed to use the underlying telecom network nodes: see Node SLAs.

For information on creating SLAs, see Defining Service Provider Group and Application Group 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 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 can be retrieved from the SLA repository using the retrieve operations.

Note: The SLA that is loaded into the SLA repository is the one being enforced. Changes to the file after the SLA is loaded into the repository will not automatically be reflected in the active version.

Summary of Tasks Related to SLAs

Service Provider and Application Group SLAs

Application group SLAs

Service provider groups...	Management operation
...are associated with SLAs using	Operation: loadApplicationGroupSla Operation: loadApplicationGroupSlaFromUrl
...can be viewed using	Operation: retrieveApplicationGroupSla

Service provider groups SLAs

Application group SLAs...	Management operation
...are associated with SLAs using	Operation: loadServiceProviderGroupSla Operation: loadServiceProviderGroupSlaFromUrl
...can be viewed using	Operation: retrieveServiceProviderGroupSla

Node SLAs

Global node SLAs

Global node SLAs...	Management operation
...are loaded using	Operation: loadGlobalNodeSla Operation: loadGlobalNodeSlaFromUrl
...can be viewed using	Operation: retrieveGlobalNodeSla

Service Provider Node SLAs

Service Provider Node SLAs...	Management operation
...are associated with a service provider group using	Operation: loadServiceProviderGroupNodeSla Operation: loadServiceProviderGroupNodeSlaFromUrl
...can be viewed using	Operation: retrieveServiceProviderGroupNodeSla

Reference: ApplicationSLAs

All operations are reachable from:

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

Below is a list of operations for management:

- Operation: countApplicationSlaGroups
- Operation: countServiceProviderSlaGroups
- Operation: listApplicationSlaGroups

Operation: countApplicationSlaGroups

Scope: Domain

Displays the number of registered application groups that have SLAs associated with them.

Signature:

```
countApplicationSlaGroups()
```

Table 6-1 countApplicationSlaGroups

countApplicationSlaGroups	
Parameter	**Description**
-	-

Operation: countServiceProviderSlaGroups

Scope: Domain

Displays the number of registered service provider groups that have SLAs associated with them.

Signature:

```
countServiceProviderGroups()
```

Table 6-2 countServiceProviderSlaGroups

countServiceProviderSlaGroups	
Parameter	**Description**
-	-

Operation: listApplicationSlaGroups

Scope: Domain

Displays a list of registered application groups that has application SLAs associated.

The list contains application group IDs.

Signature:

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

Table 6-3 listApplicationSlaGroups

listApplicationSlaGroups	
Parameter	**Description**
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).

Operation: listServiceProviderSlaGroups

Scope: Domain

Displays a list of registered service provider groups that has application SLAs associated.

The list contains service provider group IDs.

Signature:

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

Table 6-4 listServiceProviderSlaGroups

listServiceProviderSlaGroups	
Parameter	**Description**
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).

Operation: loadApplicationGroupSla

Scope: Domain

Associates an application group SLA with an application group using the contents of the SLA as a parameter.

Signature:

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

Table 6-5 loadApplicationGroupSla

loadApplicationGroupSla	
Parameter	**Description**
ApplicationGroupIdentifier	Unique ID for the application group.
ServiceLevelAgreement	Contents of an application group SLA.

Operation: loadApplicationGroupSlaFromUrl

Scope: Domain

Associates an application group SLA with an application group using the URL to a file that contains the SLA.

Signature:

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

Table 6-6 loadApplicationGroupSlaFromUrl

loadApplicationGroupSlaFromUrl	
Parameter	**Description**
ApplicationGroupIdentifier	Unique ID for the application group.
ServiceLevelAgreement URL	The URL to the SLA.

Operation: loadGlobalNodeSla

Scope: Domain

Loads the global node SLA using the contents of the SLA as a parameter.

Signature:

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

Table 6-7 loadGlobalNodeSla

loadGlobalNodeSla	
Parameter	**Description**
ServiceLevelAgreement	Contents of a global node SLA.

Operation: loadGlobalNodeSlaFromUrl

Scope: Domain

Loads aglobal node SLA using the URL to a file that contains the SLA as a parameter.

Signature:

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

Table 6-8 loadGlobalNodeSlaFromUrl

loadGlobalNodeSlaFromUrl	
Parameter	**Description**
ServiceLevelAgreement URL	The URL to the global node SLA.

Operation: loadServiceProviderGroupSlaFromUrl

Scope: Domain

Associates a service provider group SLA with a service provider group using the URL to a file that contains the SLA as a parameter.

Signature:

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

Table 6-9 loadServiceProviderGroupSlaFromUrl

loadServiceProviderGroupNodeSlaFromUrl	
Parameter	**Description**
ServiceProviderGroupIdentifier	Unique ID for the service provider group.
ServiceLevelAgreementURL	The URL to the SLA.

Operation: loadServiceProviderGroupSla

Scope: Domain

Associates a service provider group SLA with a service provider group using the contents of the SLA as a parameter.

Signature:

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

Table 6-10 loadServiceProviderGroupSla

loadServiceProviderGroupSla	
Parameter	**Description**
ServiceProviderGroupIdentifier	Unique ID for the service provider group.
ServiceLevelAgreement	Contents of a service provider group SLA.

Operation: loadServiceProviderGroupNodeSla

Scope: Domain

Associates a service provider group node SLA with a service provider group using the contents of the SLA a parameter.

Signature:

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

Table 6-11 loadServiceProviderGroupNodeSlaFromUrl

setServiceProviderGroupNodeSlaFromUrl	
Parameter	**Description**
ServiceProviderGroupIdentifier	Unique ID for the service provider group.
ServiceLevelAgreement	The contents of the service provider group node SLA.

Operation: loadServiceProviderGroupNodeSlaFromUrl

Scope: Domain

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.

Signature:

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

Table 6-12 loadServiceProviderGroupNodeSlaFromUrl

setServiceProviderGroupNodeSlaFromUrl	
Parameter	**Description**
ServiceProviderGroupIdentifier	Unique ID for the service provider group.
ServiceLevelAgreement URL	The URL to the SLA.

Operation: retrieveApplicationGroupSla

Scope: Domain

Retrieves an application group SLA.

Signature:

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

Table 6-13 retrieveApplicationGroupSla

retrieveApplicationGroupSla	
Parameter	**Description**
ApplicationGroupIdentifier	Unique ID for the service provider group.

Operation: retrieveGlobalNodeSla

Scope: Domain

Retrieves a global node SLA.

Signature:

```
retrieveGlobalNodeSla()
```

Table 6-14 retrieveGlobalNodeSla

retrieveGlobalNodeSla	
Parameter	**Description**
-	-

Operation: retrieveServiceProviderGroupNodeSla

Scope: Domain

Retrieves an service provider group node SLA.

Signature:

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

Table 6-15 retrieveServiceProviderGroupNodeSla

retrieveServiceProviderGroupNodeSla	
Parameter	**Description**
ServiceProviderGroupIdentifier	ID for the service provider group.

Operation: retrieveServiceProviderGroupSla

Scope: Domain

Retrieves an service provider group SLA.

Signature:

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

Table 6-16 retrieveServiceProviderGroupSla

retrieveServiceProviderGroupNodeSla	
Parameter	**Description**
ServiceProviderGroupIdentifier	ID for the service provider group.

CHAPTER 7

Managing a Subscriber SLA

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 *Platform Development Studio - Developer's Guide*, a separate document in this set. This chapter simply covers the three management operations available to the system administrator for loading the Subscriber SLA that the mechanism uses into the SLA repository and for retrieving the Subscriber SLA from the repository.

Note: The Network Gatekeeper Management Console displays these operations whether or not a subscriber Profile Provider has been created and deployed.

Reference: Subscriber SLA

All operations are reachable from:

- Network Gatekeeper Management Console Managed Object: Container Services–> AccountService–>ApplicationSLAs
- MBean: com.bea.wlcp.wlng.account.management.ServiceLevelAgreementMBean

Below is a list of attributes and operations for management:

- Operation: loadSubscriberSla
- Operation: loadSubscriberSlaFromUrl
- Operation: retrieveSubscriberSLA

Operation: loadSubscriberSla

Scope: Domain

Loads the Subscriber SLA into the repository.

Signature:

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

Table 7-1 loadSubscriberSla

loadApplicationGroupSla	
Parameter	**Description**
ServiceLevelAgreement	The contents of a Subscriber SLA.

Operation: loadSubscriberSlaFromUrl

Scope: Domain

Loads the Subscriber SLA into the repository using a URL.

Signature:

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

Table 7-2 loadSubscriberSla

loadApplicationGroupSla	
Parameter	**Description**
ServiceLevelAgreementURL	The URL for the Subscriber SLA

Operation: retrieveSubscriberSLA

Scope: Domain

Returns the Subscriber SLA.

Signature:

```
retrieveSubscriberSla()
```

Table 7-3 retrieveSubscriberSla

loadApplicationGroupSla	
Parameter	**Description**
-	-

CHAPTER 8

Managing Sessions

This section describes how to configure the behavior of and manage ongoing sessions:

- 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 Network Gatekeeper. Sessions can be used for establishing site affinity when using geographical redundant sites.

The setting of Attribute: sessionRequired specifies if sessions are used or not.

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

When a session is established it is always in state ACTIVE.

When a session is in state...	It transitions into state...	When...
ACTIVE	INVALID	The age of the state ACTIVE of a session is older than a configurable time-interval, see Attribute: validityTime.
ACTIVE	n/a (This is not a state but rather that the session is not current, or not existing.)	Either one of these scenarios are applied: • An application request to destroy the session using, depending on which session establishment mechanism being used, either: – Operation: destroySession on the Session Manager Web Service. – Operation: applicationLogout on the Access Web Service. • A management user performs one of the following management operations: – Operation: destroySession – Operation: destroyApplicationInstanceSession – Operation: destroyApplicationSessions – Operation: destroyServiceProviderSessions
INVALID	ACTIVE	An application request to refresh the session using, depending on which session establishment mechanism being used, either: • Operation: refreshSession on the Session Manager Web Service. • Operation: refreshLoginTicket on the Access Web Service.

When a session is in state...	It transitions into state...	When...
INVALID	n/a (This is not a state but rather that the session is not current, or not existing.)	The age of the state INVALID of a session is older than a configurable time-interval, see Attribute: expiryTime.
		A management user performs one of the following management operations: • Operation: destroyApplicationInstanceSession • Operation: destroyApplicationSessions • Operation: destroySession • Operation: destroyServiceProviderSessions

Reference: ApplicationSessions

All operations are reachable from:

- Network Gatekeeper Management Console Managed Object: Container Services–> AccountService–>ApplicationSessions
- MBean: com.bea.wlcp.wlng.account.management.ApplicationSessionMBean

Below is a list of attributes and operations for ApplicationSessions:

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

- Operation: listApplicationSessions

Attribute: expiryTime

Scope: Domain

Unit: minutes

Format: int

Specifies the expiry time of sessions.

See About Sessions.

Attribute: sessionRequired

Scope: Domain

Unit: n/a

Format: Boolean

Specifies if sessions must be established prior to use the application-facing interfaces in Network Gatekeeper. See About Sessions.

Attribute: validityTime

Scope: Domain

Unit: minutes

Format: int

Specifies the validity time of sessions.

See About Sessions.

Operation: countApplicationSessions

Scope: Domain

Displays the number of established sessions filtered on:

- service provider account
- application account

Signature:

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

Table 8-1 countApplicationSessions

countApplicationSessions	
Parameter	**Description**
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.

Operation: destroySession

Scope: Domain

Destroys an ongoing session using the ID of the session.

Signature:

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

Table 8-2 destroySession

destroySession	
Parameter	**Description**
SessionIdentifier	ID of the session to destroy.

Operation: destroyApplicationInstanceSession

Scope: Domain

Destroys the ongoing session using the ID of the application instance that established the session.

Signature:

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

Table 8-3 destroyApplicationInstanceSession

destroyApplicationInstanceSession	
Parameter	**Description**
ApplicationInstanceIdentifier	ID of the application instance that established the session.

Operation: destroyApplicationSessions

Scope: Domain

Destroys all ongoing session established by applications associated with a specific service provider /application account combination.

Signature:

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

Table 8-4 destroyApplicationSessions

destroyApplicationSessions	
Parameter	**Description**
ApplicationIdentifier	ID of the application account.
ServiceProviderIdentifier	ID of the service provider account.

Operation: destroyServiceProviderSessions

Scope: Domain

Destroys all ongoing session established by applications associated with a specific service provider account.

Signature:

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

Table 8-5 destroyServiceProviderSessions

destroyServiceProviderSessions	
Parameter	**Description**
ServiceProviderIdentifier	ID of the service provider account.

Operation: getApplicationInstanceSession

Scope: Domain

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

Signature:

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

Table 8-6 getApplicationInstanceSession

getApplicationInstanceSession	
Parameter	**Description**
ApplicationInstanceIdentifier	ID of the application instance.

Operation: getCurrentSession

Scope: Domain

Gets information about a session, ongoing 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

Signature:

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

Table 8-7 getCurrentSession

getCurrentSession	
Parameter	**Description**
SessionIdentifier	ID of the session to get information about.

Operation: listApplicationSessions

Scope: Domain

Displays a list of IDs for all sessions, ongoing or invalid, established by applications. Filters on:

- service provider account
- application account

Signature:

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

Table 8-8 listApplicationSessions

listApplicationSessions	
Parameter	**Description**
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).

CHAPTER 9

Defining Service Provider Group and Application Group SLAs

An application's usage policies of Network 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 Creating and Maintaining Service Provider and Application Accounts.

This section describes:

- Structure of a Service Level Agreement
- Contract structure

Structure of a Service Level Agreement

The xsds for the SLAs are found in `/sla_schema/` in `$BEA_HOME/wlng400/modules/com.bea.wlcp.wlng.policy_4.0.0.0.jar`:

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

The SLA contains two main types of information specified by the attributes in the <Sla> tag and the contents of the <serviceContract> tags. Service level agreement XML file overview shows the service provider level SLA XML file's main structure and the relation between the <Sla> and the <serviceContract> tags. Differences between the SLA files on service provider and application level are described in the tag descriptions below the listing.

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

Listing 9-1 Service level agreement XML file overview

```
<Sla [serviceProviderGroupID | applicationGroupID] ="<Group>"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="[sp_sla_file.xsd | app_sla_file.xsd]">
        <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>
                <enforceAcrossGeoSites></enforceAcrossGeoSites>
        </serviceContract>
</Sla>
```

<Sla>

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

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

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

The attribute **xsi:noNamespaceSchemaLocation** shall point 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

<serviceContract>

This tag contains contractual data specifying under which conditions a service provider or an application is allowed to access and use specific services in Network Gatekeeper. One <**serviceContract**> tag is needed for each application-facing interface a service provider or an application shall have access to.

The data to be defined in the <**serviceContract**> tag for each interface is described below.

<startdate>

This tag specifies the date the application can start using the service capability. Use format YYYY-MM-DD.

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

<enddate>

This tag specifies the last date the application can use the service capability. Use format YYYY-MM-DD.

Note: A earlier end date on the service provider level service contract overrides this date.

<scs>

This tag specifies the application-facing interface that the service contract applies to. The content of this tag is a Java representation of the WSDL that defines the interface. See section Managing and Configuring the Plug-in Manager in *Network Gatekeeper Administration Guide* for information on how to get a list of valid content.

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

<contract>

This tag contains all contractual data.

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

This tag, including its' sub tags, can also be defined within the <override> tag in order to override the default restrictions.

<overrides>

This tag is a container of one or more <**override**> tags.

<override>

The contract data specified within this tag is used for overriding the default contractual data given in the <**contract**> tag directly following the <**scs**> tag.

When an override occurs, only the restrictions given in the <**override**> section are used. That is, all restrictions that are present in the normal contract will be disregarded if not explicitly restated in the override section. No quota definitions are valid within the override section.

In each occurrence of this tag, the following sub tags can be used:

- <**startDate**>, specifies the start date the contract data given within the <**override**> tag is valid. Use format YYYY-MM-DD. Optional tag. If omitted, the start date for the service contract is used.
- <**endDate**>, specifies the end date the contract data given within the <**override**> tag is valid. Use format YYYY-MM-DD. If omitted, the end date for the service contract is used.
- <**startDow**>, specifies the starting weekday for which the contract data given within the <**override**> tag is valid. Use 1 for Sunday, 2 for Monday and so on. Optional tag.
- <**endDow**>, specifies the end weekday for which the contract data given within the <**override**> tag is valid. Use 1 for Sunday, 2 for Monday and so on. Optional tag if <**startDow**> is not given.
- <**startTime**>, specifies the start time for which the contract data given within the <**override**> tag is valid. Use format hh:mm:ss, where hh can be 00-24. Optional tag.
- <**endTime**>, specifies the end time for which the contract data given within the <override> tag is valid. Use format hh:mm:ss, where hh can be 00-24. Optional tag if <startTime> is not given.
- <**contract**>, specifies the contract data that overrides the contract data given directly after the <**scs**> tag. The tags that can be defined within the <**contract**> tag are described in the sections describing the contract data for each service capability.

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

- Today's date must be the same as or later than startDate
- Todays' date must be earlier than endDate (must not be the same date)
- Current time must be between startTime and endTime. endTime being earlier than **startTime** means the limit spans midnight.
- Current day of week must be between startDow and endDow or equal to startDow or endDow. endDow being less than startDow means the limit spans the week end.

Since several <**override**> tags can be defined, with different settings for the time periods for which the restrictions applies, make sure the time periods do not overlap. In the case of overlapping time periods, there is no guarantee which contract that applies.

Below are two examples of <**override**> sections.

Listing 9-2 Listing D-2 <override> example

```
<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>
```

<enforceAcrossGeoSites>

This tag specifies if the SLA shall be enforced across geo-redundant sites.

Specify **<enforceAcrossGeoSites>**true**<enforceAcrossGeoSites>** if the SLA shall be enforced across geo-redundant site, otherwise **false**.

Optional tag. If not present, default is false.

Note: If set to **true**, the SLA must not contain any **<override>** tag.

Contract structure

The listing below illustrates the structure of the <contract> segment of an SLA.

Listing 9-3 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>
</contract>
```

<contract>

This tag serves to hold together all service-specific data. A <**contract**> tag occurs once for every <**scs**> and once for every <**override**>.

<guarantee>

Optional.

This tag is used to specify a number of method requests the service provider or application is guaranteed during a specified time period (in milliseconds). Method requests from service providers and applications having the method tagged as guaranteed will have precedence above 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 stated 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**> tag must encapsulate the <**methodNameGuarantee**>, <**reqLimitGuarantee**>, and <**timePeriodGuarantee**> tags within a <**methodGuarantee**> tag.

Note: The <**methodGuarantee**> tag is ignored for mobile-originated traffic.

Each method in <**methodNameGuarantee**> must also be defined in <**methodRestrictions**>. The time period defined for a certain method must be identical in both the <**guarantee**> tag and the <**methodRestriction**>. See <methodRestriction>.

<methodNameGuarantee>

The name of the method to have guaranteed precedence.

<reqLimitGuarantee>

The number of requests to be guaranteed over the time period given in <**timeperiodGuarantee**>.

<timePeriodGuarantee>

The timeperiod to apply the guarantee, given in milliseconds.

<methodRestrictions>

Optional.

This tag is used for restricting the number of method requests an application is allowed to do over a short time period, the short time period is referred to as a *rate*, typically spanning a few seconds. A longer time period, referred to as a quota, typically spans several days.

Note: The <**methodRestrictions**> tag is ignored for mobile-originated traffic.

One <**methodRestrictions**> tag, including one or more <**methodRestriction**> tags, is needed for each method that should have restricted usage. Only one <**methodRestrictions**> tag is allowed.

Below is an example with usage restrictions both for quotas and rates.

Listing 9-4 Example of a usage restriction

```
<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 example above specifies the usage restrictions for the methods SendSms and SendSmsLogo. The usage restriction for the short time period is 5 requests per second (5 requests divided by 1000 milliseconds). The usage restriction for the longer time period, the quota, is 600 requests over a 3 day period.

If the application does not have any usage restrictions within the allowed methods, the whole <**restriction**> tag should be deleted or commented out.

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 the one being enforced.

<methodRestriction>

Contains restrictions for a method in the application-facing interface. Encapsulates the tags:

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

<methodName>

The name of the method to put restrictions on. The content of this tag is the Java representation of the method defined in the WSDL. See section Managing and Configuring the Plug-in Manager in *Network Gatekeeper Administration Guide* for information on how to get a list of valid method names.

<rate>

This tag defines the maximum number of requests to be guaranteed over a short time period, *rate*.

The <**rate**> tag contains two sub-tags:

- <**reqLimit**> defines the maximum number of requests over the timeperiod given in <**timePeriod**>
- <**timePeriod**> defines the time period, given in milliseconds.

<quota>

This tag defines the maximum number of requests over a longer time period.

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

The <**quota**> tag contains the tags:

- <**qtaLimit**> defines the maximum number of requests over the timeperiod defined in <**qtaLimit**>.

- <**days**> defines the timeperiod period, given in days (only integers are valid). The starting day (day 0) is the same day as the <**startdate**> for the service contract. Only one quota limit and time period can be specified per <**quota**> tag.
- <**limitExceedOK**> is used to specify what action to take if the quota limit is exceeded. If it is defined as true, the request will be allowed even if the quota is exceeded. If it is defined as false, the request will be rejected. An alarm is always emitted if the limit is exceeded.

<methodAccess>

Optional.

This tag 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, the <**methodAccess**> tag should be omitted.

This tag encapsulates one or more <**blacklistedMethod**> tags.

<params>

This tag is used to either allow or deny certain parameters values to be provided by applications.

The <**params**> tag can encapsulate zero (0) or more <**methodParameters**> tags, each containing a <**methodName**>, <**parameterName**>, <**parameterValues**> and <**acceptValues**> tag.

In the example below, the parameter subject is allowed to contain only the parameter values A, B, and C for the method sendMessage. No other values for this parameter are allowed.

Listing 9-5 Example of <params>

```
<params>
        <methodParameters>
                <methodName>sendMessage</methodName>
                <parameterName>arg0.subject</parameterName>
                <parameterValues>A B C</parameterValues>
                <acceptValues>true</acceptValues>
        </methodParameters>
```

```
</params>
```

<methodParameters>

Encapsulates the <**methodName**>, <**parameterName**>, <**ParameterValue**> and <**acceptValues**> tag.

- <**methodName**> contains name of the method whose parameters are to be allowed or denied.
- <**parameterName**> contains the name of the parameter whose values are to be defined as allowed or denied. The content of this tag is the Java representation of the parameter defined in the WSDL. See section Managing and Configuring the Plug-in Manager in *Network Gatekeeper Administration Guide* for information on how to get a list of valid parameter names. The representation is arg0.<name of parameter as defined in WSDL>
- <**parameterValue**> tag contains a list of allowed values. Several values can be defined, separated by white space.
- <**acceptValues**> tag defines if the parameter values shall be allowed or denied. If the tag contains true, the parameter values given are allowed and all other values are denied. If the tag contains false, the parameter values given are denied and all other values are allowed.

<requestContext>

Encapsulates one or more <**contextAttribute**> tags.

A <**contextAttribute**> tag contains one <**attributeName**> and one <**attributeValue**> tag.

A plug-in can retrieve the value specified in <**attributeValue**> using the name specified in <**attributeName**>.

Note: The plug-in must implement the functionality for fetching the value by name, see section Policy in *Platform Development Studio- Developer's Guide* for more information. Contact support for information on which parameters are supported by the standard Communication Services provided in this release.

Listing 9-6 Example of <requestContext>

```
<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>
```

<resultRestrictions>

Optional.

This tag is used to filter results from application-initiated requests.

This tag encapsulates one or more **<resultRestriction>** tags.

<resultRestriction>

Specifies a result restriction.

Encloses the following sub-tags:

- **<methodName>** the method for which to restrict the return parameter.
- **<parameterRemovalName>** the parameter, or set of parameters to remove.
- **<parameterMatch>** filter to match to enable the removal (optional).
- **<filterMethod>** defines how the filter is applied.

The result restriction makes is possible to remove return parameters from application-initiated requests.

The name of the method is identical to the name in the WSDL that defines the interface.

The return parameter 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 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
```

Note: In the example above, an array of SmsMessage is returned, hence `arg0.result[]`

The tag **<parameterRemovalName>** specifies which part of the return parameter to remove. It is possible to explicitly specify a leaf or to specify a node in the hierarchy. When specifying a node, the node and all siblings are removed. If the parameter to remove is a Boolean, it will not be removed but set to False.

For example, if specifying

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

only that part of the result parameter is removed, since it is a leaf.

If specifying:

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

All of the siblings are removed from the result parameters:

- `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`

See section Managing and Configuring the Plug-in Manager in *Network Gatekeeper System Administrator's Guide* for information on how to get a list of all method names and parameter names.

<parameterMatch>

Encapsulates the following sub-tags:

- **<parameterName>** (exactly one)
- **<parameterValues>** (exactly one).

Optional tag.

If **<parameterMatch>** is present, result filtering takes place only if the parameter defined in **<parameterName>** has a value defined in one of the **<parameterValue>** tags.

The format of **<parameterName>** is the same as for **<parameterRemovalName>**.

<ParameterValues> encapsulates one or more **<ParameterValue>** tags.

The **<ParameterValue>** tag is the value of **<parameterName>**. The value can be expressed as a regular expression.

<filterMethod>

Enumeration. Possible values:

- BLACK_LIST
- WHITE_LIST

Defines how the filter defined in <**parameterMatch**> is applied.

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.

Example

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

- dataName
- dataValue

Listing 9-7 illustrates the SLA.

Listing 9-7 Result filter 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>
  </resultFilter>
<resultFilfters>
```

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”

since **<filterMethod>** is **BLACK_LIST** and **<parameterValue>** contains only **ssn** and **homephone**, these 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”

since only these parameters matches the filter.

If <**parameterRemovalName**> is `result.data` instead of `result.data[]` the filtered result would be null in both cases, since both filters match and the absence of the [] means that all results starting from the hierarchy `result.data` should be removed given that the filter matches the result.

CHAPTER 10

Defining Global Node and Service Provider Group Node SLAs

In addition to SLAs that describe an applications use of Network Gatekeeper, there is a separate SLA that defines that application's access to underlying network elements. In addition, there is an SLA that defines Network Gatekeeper as a whole's access to underlying network elements. These are known as Node SLAs and are expressed in XML.

This section describes:

- Structure of a Node Service Level Agreement
- Node SLA Overview
- Service Provider Group Node SLA
- Global Node SLA

Structure of a Node Service Level Agreement

Table 10-1 Structure and contents of a global node and service provider group node SLA

Tag	Description
<Sla>	Main tag. Contains: <nodeContract>, zero (0) or more. <globalContract>, zero (0) or more. **Note:** <nodeContract> and <globalContract> are mutually exclusive in one SLA instance. Has the following attributes: serviceProviderGroupID, that contains the service provider group ID for the SLA is associated with, in the case of a service provider group node SLA. Remove if it is a global node SLA
<nodeContract>	Parent tag: <Sla>. Defines under which conditions one, or a set of network node(s) can be accessed by Network Gatekeeper. *This contract takes into account which service provider a requests originates from.* This tag is use for service provider node SLAs. These SLAs are provisioned using the management operations described in Service Provider Node SLAs. Contains the following tags: • <startDate>, one (1) • <endDate>, one (1) • <nodeID>, one (1) • <nodeRestrictions>, zero (0) or one (1).

Table 10-1 Structure and contents of a global node and service provider group node SLA

Tag	Description
<globalContract>	Parent tag: <Sla>. Defines under which conditions one or a set of network node(s) can be accessed by Network Gatekeeper. *This contract does not take into account which service provider a requests originates from. It defines how Network Gatekeeper as a whole can use the node(s).* These SLAs are provisioned using the management operations described in Global node SLAs Contains the following tags: • <startDate>, one (1) • <endDate>, one (1) • <nodeID>, one (1) • <globalRestrictions>, zero (0) or one (1)
<nodeRestrictions>	Parent: <nodeContract> Contains the following tags: • <nodeRestriction>, zero (0) or one (1)
<nodeRestriction>	Parent: <nodeRestrictions> Contains the following tags: • <reqLimit>, zero (0) or one (1) • <timePeriod>, zero (0) or one (1)
<globalRestrictions>	Parent: <nodeContract> Contains the following tags: <globalRestriction>, zero (0) or one (1)
<globalRestriction>	Parent: <globalRestrictions> Contains the following tags: • <reqLimit>, zero (0) or one (1) • <timePeriod>, zero (0) or one (1) • <guaranteePercentage>, zero (0) or one (1)

Table 10-1 Structure and contents of a global node and service provider group node SLA

Tag	Description
<startDate>	Specifies the date the service provider or WebLogic Network Gatekeeper, depending of it is a service provider group node SLA or a global node SLA can start accessing the network node. Format YYYY-MM-DD.
<endDate>	Specifies the last date (expiry date) the service provider or WebLogic Network Gatekeeper can access the network node. Format YYYY-MM-DD.
<nodeID>	Specifies the network node's node ID as registered in the Plug-in manager.A node ID can be assigned to one or more network nodes and the contract is subsequently valid for one or a group of nodes. The plug-in manager is used for provisioning of node IDs, see the section about the Plug-in manager in the *System Administrator's Guide*.
<reqLimit>	Content of this tag specifies a number of requests. Specifies the number of requests allowed during the time-period specified in <timePeriod>.
<timePeriod>	Specifies the time period during which the request limit, specified in <reqLimit>, is valid. Unit: milliseconds.
<guaranteePercentage>	Specifies at which request rate only requests that are guaranteed are passed on the network node. Requests are marked as guaranteed in the service provider group and application group SLAs. Expressed as a fraction of <reqLimit> divided by <timeperiod>. Unit: percentage.

Node SLA Overview

The SLAs are written in XML. The SLA looks slightly different depending on the type of node SLA. The types are:

- global node SLA
- service provider group node SLA

The schema for the global node SLA is found in:

`$BEA_Home/wlng400/modules/com.bea.wlcp.wlng.policy_<version>.jar:sla_schema/global_node_sla_file.xsd`

The schema for the service provider group node SLA is found in:

`$BEA_Home/wlng400/modules/com.bea.wlcp.wlng.policy_<version>.jar:sla_schema/sp_node_sla_file.xsd`

Service Provider Group Node SLA

The service provider group node SLA consists of an **<sla>** tag containing zero (0) or more **<nodeContract>** tags as shown in Listing 10-1. The `serviceProviderGroupID` attribute specifies the service provider group for which the SLA file is valid. The structure and contents of the **<nodeContract>** tag is further described in Global contract data.

Listing 10-1 Service provider group node SLA

```
<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>
```

Node Contract

The **<nodeContract>** tags contains a set of tags that define under which conditions a network node can be accessed by a service provider. See Listing 10-2. The contents of each tag are further described below the listing.

Listing 10-2 <nodeContract> tag contents

```
<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>
```

Global Node SLA

The global node SLA consists of an **<sla>** tag containing zero (0) or more **<globalContract>** tags as shown in Listing 10-3. In this case, the `serviceProviderGroupID` attribute is left empty. The structure and contents of the **<globalContract>** tag are further described in Node Contract.

Listing 10-3 Global Node SLA

```
<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>
```

Global Contract

The **<globalContract>** tags contains a set of tags that define under which conditions a network node can be accessed. The contents of each tag are further described below in Listing 10-4.

Listing 10-4 <globalContract> tag contents

```
<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>
```