Oracle® Communications Service Broker

Integration Guide

Release 5.0

E15187-01

December 2010

ORACLE®

Oracle Communications Service Broker Integration Guide, Release 5.0

E15187-01

Copyright © 2010 Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface

This document describes the tools you can use and the integration points for integrating Oracle Communications Service Broker with external management systems.

Audience

This document is intended for system administrators, developers, and system integrators who want to integrate Service Broker configuration and management with an external management system.

Related Documents

For more information, see the following documents in the Oracle Communications Service Broker Release 5.0 documentation set:

- *Oracle Communications Service Broker 5.0 Release Notes*
- *Oracle Communications Service Broker 5.0 Concepts Guide*
- *Oracle Communications Service Broker 5.0 Installation Guide*
- *Oracle Communications Service Broker 5.0 Configuration Guide*
- *Oracle Communications Service Broker 5.0 System Administrator's Guide*

1

About Integrating Service Broker With External Management Systems

This chapter gives an overview of ways to integrate Oracle Communications Service Broker with an external management system.

Management System Integration Mechanisms

Service Broker offers the following integration mechanisms:

- Java Management Extension (JMX)

  You can directly access Service Broker MBeans by using JMX. Domain-level configuration MBeans are exposed by the Administration Console and are stored in an Administration Console subdirectory. Each Processing Server and Signaling Server has an MBean server that exposes MBeans for reading configuration values, monitoring runtime aspects of modules, and managing the server life cycle.

  See Chapter 2, "About MBeans for Monitoring and Configuration," for information about the different types of Service Broker MBeans.

  See Chapter 3, "Integrating an External Management System by Using JMX," for information about accessing Service Broker MBeans.

- SNMP agent

  SNMP enables remote monitoring of the Processing Server or Signaling Server. You can configure Service Broker to respond to SNMP queries and send SNMP traps to external management systems. See Chapter 4, "Monitoring Service Broker with SNMP," for more information.

2

About MBeans for Monitoring and Configuration

This chapter describes how you use MBeans when integrating Oracle Communications Service Broker with an external management system.

About Service Broker MBeans

Service Broker MBeans are used for creating, reading, updating, and deleting configuration data, for monitoring runtime aspects of modules, and for managing the life cycle of servers. These MBeans are categorized as either configuration MBeans or runtime MBeans:

- You use configuration MBeans to configure Service Broker components. The MBeans provide get and set operations and are exposed by the Administration Console.
- You use runtime MBeans to monitor the runtime state of components and to control the running state of Processing Servers and Signaling Servers. Runtime MBeans are exposed by the Processing Servers and Signaling Servers. The servers also expose a set of read-only configuration MBeans, used to read configuration values.

All MBeans can be accessed programmatically through JMX. Using JMX enables you to integrate Service Broker configuration, management, and monitoring with an external management system.

Information about Service Broker MBeans is described in different books, based on the type of MBean and the module to which it applies. The location of MBean documentation is provided later in this document.

The root MBeans are listed in this document. Each module has a root MBean and most root MBeans include below it a hierarchy of additional MBeans that relate to the same module. All Mbeans in the same hierarchy are documented in the same chapter.

About MBean Object Naming Conventions

There are two methods that Service Broker uses to derive MBean object names. Each method produces object names that have one of two common prefixes, which are also used in the package names:

- com.convergin
- **oracle.axia**

Both sets include runtime MBeans and configuration MBeans for various modules.

Information about the object names are located in different books:

- For information about the naming convention of **com.convergin** MBean objects, see the following documents:
  - "Service Broker MBean Object Names" under "Understanding Service Broker Configuration" in *Oracle Communications Service Broker Configuration Guide* for information about configuration MBean object names.
  - "Service Broker Runtime MBean Object Names" under "Understanding Service Broker Runtime MBeans" in *Oracle Communications Service Broker System Administrator's Guide* for information about runtime MBean object names.
- The object naming convention for **oracle.axia** MBeans is not described. Instead, the object name for each MBean is given in the MBean documentation in various chapters.

Runtime MBeans

Runtime MBeans provide operations for monitoring runtime aspects of modules and managing the life cycle of individual servers.

Examples include monitoring internal queues, listing recently sent SNMP traps, changing the running state of server, and viewing traffic statistics for a server.

Location of Runtime MBean Documentation

Oracle Communications Service Broker System Administrator's Guide includes information on these MBeans:

- (**oracle.axia**) ManagementAgentMBean, in "Life cycle of Processing Servers and Signaling Servers"
- (**oracle.axia**) DeploymentServiceMBean, in "Upgrading and Patching"
- (**oracle.axia**) DomainServiceMBean, in "Managing Domains"
- In "Monitoring Service Broker with JMX Runtime MBeans":
  - (**com.convergin**) All Interworking Module (IM) MBeans used to monitor IM traffic statistics
  - (**com.convergin**) All Signaling Server Unit (SSU) MBeans used to monitor SSU traffic statistics
  - (**com.convergin**) OeRuntimeMBean

Configuration MBeans

Configuration MBeans provide create, read, update, and delete (CRUD) operations that apply to the configuration of a domain and of individual modules.

Examples of module configurations include reading and updating configuration settings for SMNP traps, Interworking Modules, Signaling Server Units.

Examples of domain configurations include adding servers to and removing servers from a domain configuration and deploying and undeploying OSGi bundles.

The configuration MBeans define **set** and **get** operations. The settings from these operations are persisted in the domain configuration. The configuration settings are propagated to Processing Servers and Signaling Servers. When the domain is online, the changes are immediately propagated to the servers using a two-way commit pattern. The configuration is always fetched by the servers when they start.

MBean Hierarchies

Each configurable module has a root MBean. Under this hierarchy, there can be a tree of MBeans all relating to the same module. All MBeans for a specific module belong to the same package.

Each MBean has one parent MBean and zero or more sibling MBeans. Several instances of an MBean can occur under a parent MBean. An instance of an MBean can exist under several different parent MBeans.

Conventions for oracle.axia Configuration MBeans

This section includes information about the conventions used when working with **oracle.axia** configuration MBeans.

Adding and Removing MBean Instances

Instances of **oracle.axia** configuration MBeans can be created and removed using the **add** and **remove** methods of the parent MBean. The naming syntax for these methods is:

- **void add***MBean_name***()**
- **void remove***MBean_name***()** or **void remove***MBean_name***(int** *index***)**

where:

- *MBean_name* is the name of the interface that defines the MBean to be added or removed. When specifying *MBean_name*, omit the suffix **MBean**, which is present in the MBean name.
- *index* is the index number of the MBean instance to remove.

The **remove** method takes the *index* parameter when the parent MBean allows more than one sibling MBean.

For example, the MBean **UnicastAddressesMBean** can have zero or more instances of **UnicastAddressMBean** as siblings. Hence **UnicastAddressesesMBean** defines the methods **void addUnicastAddress()** and **void removeUnicastAddress(int** *index***)**.

When multiple instances of a sibling MBean are allowed, an index is appended to the object name. The index starts at 0 and is incremented automatically as MBeans are added. When an MBean is removed, the indexes of the remaining MBeans are re-calculated to retain an uninterrupted sequence.

If only one instance of an MBean is allowed, an exception is thrown if the **add** method is invoked when there is already a registered instance of that MBean.

An exception is also thrown if the **remove** method is invoked for an MBean when there are no registered instances of that MBean.

MBeans can also be created programmatically by using factory methods. Factory methods for all MBeans for a specific module are provided in a class named **ObjectFactory**. There is one **ObjectFactory** for each MBean hierarchy.

Scope of Configuration MBeans

Configuration settings are applied either to all servers in the domain or to individual servers, as defined by the configuration scope. For the MBeans that can have a scope other than domain, the scope is expressed using the **Target** attribute. The **target** attribute is a string that is set to the name of the server to which the MBean applies. The server name is the name as defined in the domain configuration. Alternatively, if **Target** is empty, the configuration applies to all servers in the domain. When a target is set, the configuration applies to the MBean in which it is set and to all its sibling MBeans.

Read and Write Interfaces

Each **oracle.axia** configuration MBean is defined by two interfaces: one defines **set** (write) operations for the MBean and the other defines **get** (read) operations:

- The interface that defines **get** operations has the suffix **ReadOnlyMBean**. Read-only MBeans are available on each Processing Server and Signaling Server.
- The interface that defines **set** operations has the suffix **MBean** and inherits from the corresponding interface that defines **get** operations, and so exposes the full set of **get** and **set** operations.

  For example, the interface **MulticastAddressMBean** has **MulticastAddressReadOnlyMBean** as a Superinterface. **MulticastAddressMBean** defines all write operations whereas **MulticastAddressReadOnlyMBean** defines the corresponding read operations.

Both interfaces belong to the same package. For example, **MulticastAddressMBean** and **MulticastAddressReadOnlyMBean** both belong to the package **oracle.axia.config.beans.storage.coherence**.

Conventions for com.convergin Configuration MBeans

For information about the conventions used when working with **com.convergin** configuration MBeans, see "Configuration MBeans Overview" in *Oracle Communications Service Broker Configuration Guide*.

Location of Configuration MBean Documentation

Oracle Communications Service Broker Integration Guide includes information about these MBeans:

- (**oracle.axia**) SnmpConfigTypeMBean, in "Monitoring Service Broker with SNMP"

Oracle Communications Service Broker Configuration Guide includes information about these MBeans (these are all **com.convergin** MBeans):

- ChangeCenterMBean, in "Understanding Service Broker Configuration"
- DeploymentMBean, in "Managing Service Broker Interworking Modules"
- OeMBean, in "Configuring Service Broker Orchestration Engine"
- All IM MBeans in the respective chapter on configuring each IM
- All SSU MBeans in the respective chapter on configuring each SSU
- All Supplementary Module (SM) MBeans in the respective chapter on configuring each SM

Oracle Communications Service Broker System Administrator's Guide includes information on these MBeans:

- (**oracle.axia**) DeploymentServiceMBean, in "Upgrading and Patching"
- (**oracle.axia**) DomainServiceMBean, in "Managing Domains"

3

Integrating an External Management System by Using JMX

This chapter gives an overview and examples of how you can use Java Management Extension (JMX) to programmatically access MBeans for the purpose of integrating Oracle Communications Service Broker with an external management system.

This chapter describes accessing the MBeans that are exposed by Processing Servers and Signaling Servers. For a description of accessing MBeans exposed by the Administration Console, see "Configuration MBeans Overview" in *Oracle Communications Service Broker Configuration Guide*.

About Integrating Using JMX

Using JMX, you can access the monitoring MBeans and read-only configuration MBeans that are exposed by the MBean server on each Processing Server and Signaling Server.

Location of MBean Definitions

You need to have access to the Java Archive (JAR) files that define the MBean interfaces and the related classes. The MBean interfaces and classes for a module are provided in the OSGi bundles for the module.

The JARs for the OSGi bundles are located in the directory *Oracle_home***/axia/admin_console/modules**, where *Oracle_home* is the directory in which Service Broker is installed.

The name of the JAR file for **oracle.axia** MBeans is provided with the reference information for the MBean. For example, reference information for **SnmpConfigTypeMBean** is described in Chapter 4, "Monitoring Service Broker with SNMP", which gives the JAR file name, **oracle.axia.snmp-1.0.0.0.jar**.

Include the relevant JARs in your class path.

Connecting to a Processing Server or a Signaling Server

Processing Servers and Signaling Servers are run with SSL enabled by default. Connecting to a server that has SSL enabled requires the following:

- Your administration client needs access to the server certificate and the server needs access to the client certificate. You generate these certificates when you configure the public key infrastructure (PKI). For information about security and setting up a PKI, see "Configuring Security" in *Oracle Communications Service Broker System Administrator's Guide*.
- Your administration client must be configured to support JMX SSL and JMX SSL Client Authentication. For more information, see:

  http://java.sun.com/javase/6/docs/technotes/guides/management/agent.html

You use JMX operations to connect your management system to an MBean server on a Processing Server or a Signaling Server.

You will need the following information:

- Host name or IP address of the Processing Server or Signaling Server.
- IP port that the MBean server uses. The port corresponds to the JMX port defined for the server in the domain configuration.
- JMX registry port for the MBean server. This port corresponds to the JMX registry port defined for the server in the domain configuration.
- Absolute Java Naming and Directory Interface (JNDI) name of the MBean server. This is constructed as a URL:

  service:jmx:rmi://*host***:***port***/jndi/rmi://***host***:***registryPort***/jmxrmi**

Example 3–1 shows how to connect to an MBean server.

Example 3–1 Connecting to a Processing Server or a Signaling Server using JMX

```
final JMXConnector connector;
final MBeanServerConnection mbeanServerConnection;
String serverHost = "localhost";
int jmxPort = 10003;
int jmxRegistryPort = 10103;
String url = "service:jmx:rmi://"+serverHost+":"+jmxPort+"/jndi/rmi://"+
             serverHost+":"+jmxRegistryPort+"/jmxrmi";
JMXServiceURL serviceURL;
final HashMap<String, Object> env = new HashMap<String,Object>();
serviceURL = new JMXServiceURL(url);
connector = JMXConnectorFactory.connect(serviceURL,env);
mbeanServerConnection = connector.getMBeanServerConnection();
```

When an MBean server connection is established, it is used to set and get attributes and invoke operations on specific MBeans.

Invoking Operations

You invoke an operation using **invoke(...)** on the **MBeanServerConnection** object. The operation has the signature:

Object invoke(ObjectName *name***, String** *operationName***, Object[]** *params***, String[]** *signature***)**

where:

- *name* is the object name for the MBean. For example, **oracle:type=oracle.axia.api.management.pac.ProtocolAdapterContainerMBean**.
- *operationName* is the operation you are invoking on the MBean. For example, **getWorkManagerStatus**.
- *params* is a list of the parameters you want to provide to the operation. For example, **Object opParams[] = {"sip"};**
- *signature* is a list of types that corresponds to each parameter. For example, **String opSig[] = {String.class.getName()};**

Example 3–2 shows how to get the value of the attribute **int StartLevel** defined in the interface **oracle.axia.api.management.agent.ManagementAgentMBean**. This attribute defines the start level for the server. The object name is **oracle:type=oracle.axia.api.management.agent.ManagementAgentMBean**.

Example 3–2 Invoking an MBean Operation

```
String objNameStr;
ObjectName objectName;
String attrName;
Object returnValue;
Object opParams[];
String opSig[];

objNameStr=
"oracle:type=oracle.axia.api.management.pac.ProtocolAdapterContainerMBean";
objName = new ObjectName(objNameStr);
opName = "getWorkManagerStatus";
opParams = new Object[] {"sip"};
opSig = new String[] {java.lang.String.class.getName()};
returnValue =  mbeanServerConnection.invoke(objName, opName, opParams, opSig);
```

Getting Attribute Values

You get attributes using the operation **getAttribute(...)** on the **MBeanServerConnection** object. The operation has the signature:

Object getAttribute(ObjectName *name***, String** *attribute***)**

where:

- *name* is the object name for the MBean. For example, **oracle:type=oracle.axia.api.management.agent.ManagementAgentMBean**.
- *attribute* is the name of the attribute you are to get. For example, **StartLevel**.

Example 3–3 shows an example of how to get the value of the attribute **int StartLevel** defined in the interface **oracle.axia.api.management.agent.ManagementAgentMBean**. This attribute defines the start level for the server. The object name is **oracle:type=oracle.axia.api.management.agent.ManagementAgentMBean**.

Example 3–3 Getting an MBean Attribute

```
String objNameStr;
ObjectName objectName;
String attrName;
Object returnValue;
Object opParams[];
```

```
String opSig[];

objNameStr = "oracle:type=oracle.axia.api.management.agent.ManagementAgentMBean";
objName = new ObjectName(objNameStr);
attrName = "StartLevel";
returnValue = mbeanServerConnection.getAttribute(objName, attrName);
```

4

Monitoring Service Broker with SNMP

This chapter describes how to use Simple Network Management Protocol (SNMP) to monitor Oracle Communications Service Broker.

About Service Broker SNMP

You can monitor Oracle Communications Service Broker with enterprise-wide network management systems using the Simple Network Management Protocol (SNMP). The Service Broker supports SNMP Version 1 (SNMPv1), SNMP Version 2 (SNMPv2c), and SNMP Version 3 (SNMPv3).

SNMP management is based on the agent/manager model. The agent resides on the managed resource and provides information to one or more remote managers. In a Service Broker domain monitored by SNMP, an agent runs on each Signaling server and Processing server.

An SNMP agent provides information to managers by responding to queries or by sending unsolicited notifications (traps). SNMP queries can retrieve information on Service Broker activities, such as the number of SIP transactions processed and the length of time a module has been running.

The SNMP agent generates a trap when it detects certain predefined events or conditions on the system. For example, the Service Broker agent can send a trap when the server starts up or when an application error occurs. Traps are sent to any SNMP manager that you configure as a trap destination. In the configuration, you specify the target IP address and listening port of the remote SNMP manager.

By default, the Service Broker SNMP agent sends notifications as SNMPv2 traps. However, it can send SNMPv1 traps and SNMPv3 traps as well. Furthermore, you can define trap-forwarding groups with different SNMP versions, enabling the agent to send traps to multiple destinations in the format appropriate for each.

A management information base (MIB) module defines the properties of the system that are subject to SNMP monitoring. The MIB helps automate the discovery of the properties (or managed objects) by the MIB browser or management system.

Accessing the Service Broker MIB

The Service Broker MIB modules are:

- **ocsb.mib**: Defines managed objects associated with Service Broker runtime and management components.
- **axia.mib**: Defines managed objects associated with the underlying Axia platform.

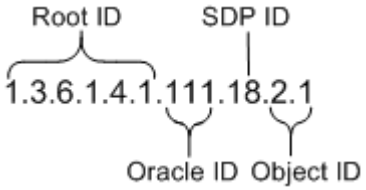
The MIB module files are included in the Oracle Communications Service Broker 5.0 Media Pack.

> **Note:** Service Broker MIB objects are read-only. You cannot modify a Service Broker configuration using SNMP.

About Service Broker SNMP Object Identifiers

Each managed object defined in a MIB has a unique object identifier (OID). An OID consists of a series of dot-delimited numbers, as shown in Figure 4–1.

Figure 4–1 Elements of an OID



All OIDs for Service Broker objects have the same root ID (1.3.6.1.4.1) and Oracle enterprise ID (111). The remaining parts of the OID identify the product group (Service Delivery Platform (SDP) in this case) and the object ID, which may be qualified by an object group.

For example, the OID for the object group relating to statistics on SIP activity is:

sipNetworkChannelStatistics: 1.3.6.1.4.1.111.18.5

A managed object within the **sipNetworkChannelStatistics** group that provides statistics for TCP connections is:

sipNetworkChannelStatisticsTcpConnections: 1.3.6.1.4.1.111.18.5.4

You can use a MIB browser to view the structure and contents of the Service Broker MIB modules, including information on each managed object, such as its OID, syntax, and status.

Service Broker Managed Objects

The Service Broker MIB modules define both queriable objects and traps. Queriable objects provide extensive information on the activities of Service Broker and its components, including those of Interworking Modules, the Orchestration Engine, the Diameter adaptor, and management components.

Traps provide information on events associated with Service Broker. Supported traps are:

- **managedServerRunningTrap**
- **managedServerStoppingTrap**

- **managedServerJoinTrap**
- **managedServerLeaveTrap**
- **bundleStartingTrap**
- **bundleActiveTrap**
- **bundleStoppingTrap**
- **bundleErrorTrap**
- **diameterConnectionUpTrap**
- **diameterConnectionDownTrap**

Configuring SNMP

You configure the Service Broker SNMP service using SNMP configuration MBeans (see "SNMP Configuration MBeans").

You can access the SNMP configuration MBeans using the Scripting Engine or JMX.

The Scripting Engine is a system administration command-line tool that you can use to invoke Service Broker MBeans. You create a script for each configuration MBean that you want to invoke and provide the script file as a parameter when running the Scripting Engine. See "Using Scripts for Configuration and Management" in *Oracle Communications Service Broker System Administrator's Guide* for information about using the Scripting Engine and creating configuration scripts.

To configure the Service Broker SNMP agent, follow these general steps:

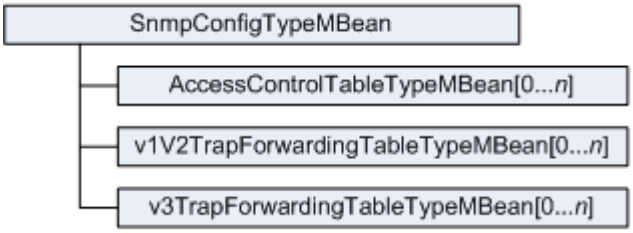
1. For each Processing or Signaling Server, use the **SnmpConfigTypeMBean** to define agent properties, including its listening port, SNMP version, and logging level. At a minimum, you need to enable the agent using this MBean.
2. If desired, specify access control restrictions applicable to SNMP queries using **AccessControlTableTypeMBean**. The MBean includes the community string required for access and IP filtering settings.
3. Specify trap destinations using **v1V2TrapForwardingTableTypeMBean** or **v3TrapForwardingTableTypeMBean**. If desired, add multiple destinations, each with its own security properties or SNMP version.

SNMP Configuration MBeans

This section describes the MBeans and settings you use to configure the SNMP service.

Figure 4–2 shows the hierarchy of the SNMP configuration MBeans. Under the root configuration MBean are MBeans for configuring access control tables, SNMPv1 and SNMPv2 trap destinations, and SNMPv3 trap destinations.

Figure 4–2 SNMP Configuration MBean Hierarchy



SnmpConfigTypeMBean

SnmpConfigTypeMBean is the root MBean for SNMP configuration. This MBean holds the basic configuration settings for the agent.

Factory Method

Created automatically.

JAR File

oracle.axia.snmp-*version*.jar

where *version* is the version number of the JAR file: for example, **1.0.0.0**.

Package

oracle.axia.config.beans.snmp

Object Name

oracle:name=oracle.axia.snmp,name0=SnmpConfig,type=oracle.axia.cm.ConfigurationMBean,version=1.0.0.0

Number of Allowed Occurrences

Maximum: 1

Attributes

enabled

Boolean value that indicates whether the SNMP agent is active. Required. Possible values are:

- true: Enables the SNMP agent.
- false: Disables the SNMP agent.

By default, the agent is disabled.

port

The port number on which the Service Broker SNMP agent listens for queries from managers. Required. Conventionally, SNMP agents listen for requests on port 161. The default value is 8001.

version

The SNMP version to use for the agent. Required. The default version is SNMPv2. Possible values are:

- V1: Sets the SNMP version to SNMPv1.
- V2c: Sets the SNMP version to SNMPv2c, or Community-Based Simple Network Management Protocol version 2.
- V3: Sets the SNMP version toSNMPv3.

loggingLevel

The SNMP agent logging level as an integer. Required. Possible values are from 1 through 6, with the values corresponding to the following logging levels:

- 1: Only fatal events are logged.

- 2: Error-level events.
- 3: Warning-level events.
- 4: Informational-level events.
- 5: Debugging-level events.
- 6: Trace-level events.

Operations

You can add or remove instances of **AccessControlTableMBean**, **v1V2TrapForwardingTableMBean**, and **v3TrapForwardingTableMBean** using the following operations.

addAccessControlTableType

Creates an instance of **AccessControlTableTypeMBean**. The access control table contains authentication settings for SNMP query requests.

removeAccessControlTable

Removes an instance of **AccessControlTableMBean**.

addV1V2TrapForwardingTable

Creates an instance of **v1V2TrapForwardingTableMBean**. The MBean contains settings that control the SNMPv1 and SNMPv2 traps sent by the Service Broker agent.

removeV1V2TrapForwardingTable

Removes an instance of **v1V2TrapForwardingTableMBean**.

addV3TrapForwardingTable

Creates an instance of **v3TrapForwardingTableMBean**. The MBean contains settings that control the SNMPv1 and SNMPv2 traps sent by the Service Broker agent.

removeV3TrapForwardingTable

Removes an instance of **v3TrapForwardingTableMBean**.

AccessControlTableTypeMBean

AccessControlTableTypeMBean specifies access control restrictions applicable to SNMP queries to SNMP agents. The agent can authenticate requests based on the community string provided in the request or by the IP address of the manager. This access control mechanism applies to SNMPv1 and SNMPv2.

Factory Method

SnmpConfigType.addAccessControlTableType()

JAR File

oracle.axia.snmp-*version*.jar

where *version* is the version number of the JAR file: for example, **1.0.0.0**.

Package

oracle.axia.config.beans.snmp

Object Name

oracle:name=oracle.axia.snmp,name0=SnmpConfig,name1=snmpSet[*n*],type=oracle.axia.cm.ConfigurationMBean,version=1.0.0.0

where *n* is an index.

Number of Allowed Occurrences

Minimum: 0; No maximum

Attributes

aclCommunity

The community string required for query access. Required. In SNMP, the community string is used to establish trust between the agent and manager.

aclAccess

The authorization level for SNMP managers who match this community. Required. Possible values are

- 0: No access.
- 1: Read-only access.

Since the Service Broker SNMP MIB defines all objects as read-only, option 2, read-write, is not supported.

aclManagers

The IP address or host name of managers who are allowed to access the agent. Required.

Requests that provide the correct community string but originate from a source IP not specified in this parameter are blocked. Use 0.0.0.0 as the IP address to allow access to any manager who provides a matching community string, or provide a specific IP address. Use semi-colons (;) to separate multiple addresses.

Operations

None

v1V2TrapForwardingTableTypeMBean

v1V2TrapForwardingTableTypeMBean specifies trap forwarding settings for SNMPv1 or SNMPv2 trap destinations. You must configure this MBean for each SNMP manager to which you want to send SNMPv1 or SNMPv2 traps from the SNMP agent.

Factory Method

SnmpConfigType.addV1V2TrapForwardingTableType()

JAR File

oracle.axia.snmp-*version*.jar

where *version* is the version number of the JAR file: for example, **1.0.0.0**.

Package

oracle.axia.config.beans.snmp

Object Name

oracle:name=oracle.axia.snmp,name0=SnmpConfig,name1=snmpSet[*n*],type=oracle.axia.cm.ConfigurationMBean,version=1.0.0.0

where *n* is an index.

Number of Allowed Occurrences

Minimum: 0; No maximum

Attributes

managerHost

The IP address or host name of the SNMP manager to which the agent sends SNMPv1 or SNMPv2 traps. Required.

managerPort

The port number on which the SNMP manager listens for traps. Conventionally, managers listen for traps on port 162. Required.

version

The SNMP protocol version to use for the traps. Required. Possible values are:

- 1: Sets the SNMP version to SNMPv1.
- 2: Sets the SNMP version to SNMPv2c, or Community-Based Simple Network Management Protocol version 2.
- 3: Sets the SNMP version to SNMPv3.

community

The community string for the trap destination manager. The community string is used to establish trust between the agent and manager.

timeout
The notification transmission timeout period, in milliseconds. If the time expires, the agent considers the transmission to have failed and may re-attempt the transmission based on the **retries** value.

retries
The number of attempts that the agent makes to send a notification. If set to 0 or the maximum number or retries has been reached, the notification is not re-attempted and the notification is considered to have failed.

Operations

None

v3TrapForwardingTableTypeMBean

v3TrapForwardingTableTypeMBean specifies destinations for SNMPv3 traps.

SNMPv3 provides enhanced security capabilities over versions 1 or 2 of SNMP. Accordingly, this MBean contains additional security-related settings.

Factory Method

SnmpConfigType.addV3TrapForwardingTableType()

JAR File

oracle.axia.snmp-*version*.jar

where *version* is the version number of the JAR file: for example, **1.0.0.0**.

Package

oracle.axia.config.beans.snmp

Object Name

oracle:name=oracle.axia.snmp,name0=SnmpConfig,name1=snmpSet[*n*],type=oracle.axia.cm.ConfigurationMBean,version=1.0.0.0

where *n* is an index.

Number of Allowed Occurrences

Minimum: 0; No maximum

Attributes

managerHost

The IP address or hostname of the SNMP manager to which the agent sends SNMPv3 traps. Required.

managerPort

The port number on which the SNMP manager listens for traps. Conventionally, managers listen for traps on port 162. Required.

version

The SNMP protocol version to use for the traps. Required. Possible values are:

- 1: Sets the SNMP version to SNMPv1.
- 2: Sets the SNMP version to SNMPv2c, or Community-Based Simple Network Management Protocol version 2.
- 3: Sets the SNMP version to SNMPv3.

community

The community string for the trap destination manager. The community string is used to establish trust between the agent and manager.

userName

The string specifying the user name of the manager user.

userSecModel
An integer value that specifies the manager's user security model. The only value supported by the Service Broker SNMP agent is 3, which specifies USM (User Security Model).

securityLevel
An integer that indicates which security features are applied to the message. You can require authentication and encryption of the trap content. Set the level using one of these options:

- 1: Without authentication and without privacy (noAuthNoPriv).
- 2: With authentication, but without privacy (authNoPriv).
- 3: With authentication and with privacy (authPriv).

userContextName
A string specifying the context of the manager user.

timeout
The notification transmission timeout period, in milliseconds. If the time expires, the agent considers the transmission to have failed, and may re-attempt the transmission based on the **retries** value.

retries
The number of attempts that the agent makes to send a notification. If set to 0 or the maximum number or retries has been reached, the notification is not re-attempted and the notification is considered to have failed.

Operations

None